Technische Universität München
TUM School of Computation, Information and Technology

Robot Systems in Industrial Applications: From Trajectory Optimization and Computer Vision to Grasp Planning

Jianjie Lin

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Daniel Cremers

Prüfer*innen der Dissertation:

1. Prof. Dr.-Ing. habil. Alois C. Knoll
2. Prof. Dr.-Ing. Torsten Kröger

Die Dissertation wurde am 11.04.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 17.04.2023 angenommen.

Abstract

A modern robotic system is a multidisciplinary product, encompassing various aspects from modern engineering methods to environmental perception. It plays a crucial role in the realization of robot automation. The motivation behind this thesis is to demonstrate that autonomous robot tasks can be described and solved through trajectory planning, environment sensing, and grasp planning. Approaches based on teach-in concepts or predefined parameters often lead to long development cycles that are unsuitable for flexible and modular small and medium-sized factory environments. Additionally, the drive to minimize human intervention and achieve automation has prompted me to further enhance critical aspects of robotics technology. In this work, I will develop and upgrade intelligent robotic systems by improving trajectory planning, environmental perception, and grasp planning separately.

The first component I need to improve is trajectory planning, based on an analysis of current robotic systems. Unlike most state-of-the-art approaches, which only consider kinematic constraints and acceleration values or utilize spline-based methods, I utilize the well-known trapezoidal acceleration profile (seven-segment model) as the fundamental model and extend it to a multi-waypoint scenario. The computational complexity of a multi-waypoint trapezoidal acceleration model exponentially increases with the number of waypoints generated by a collision-free motion planning algorithm. To reduce computational complexity, I follow the principle of model predictive control and solve the trajectory optimization problem by decomposing it into a series of subproblems with a set of nonlinear constraints. I solve each subproblem iteratively, significantly reducing the complexity. Furthermore, I extend the standard trajectory planning application by considering the trajectory blending scenario, which aims to reduce trajectory overshooting.

The perception of the environment is a crucial aspect enabling autonomous robotic operations. This thesis primarily focuses on a 3D world capable of semantically describing reality. I propose and develop new algorithms within a deep learning framework to address classical problems in the computer vision domain, such as 3D object classification, pose estimation, and 3D point cloud completion. These algorithms aim to overcome the limitations of other state-of-the-art methods. When it comes to 3D point cloud classification, there exists an inherent problem related to rotation properties. Most deep learning-based approaches require large datasets to mitigate this rotation invariant problem. However, I propose a new approach by introducing the concept of a spherical harmonics rotation-invariant descriptor. This novel approach drastically reduces the training dataset requirement and outperforms other state-of-the-art methods. Pose estimation has been dominated by iterative closest point (ICP) based approaches. However, these algorithms heavily rely on a precise initial guess for the search of corresponding points. Otherwise, the iteration process can easily become stuck in a local minimum or a saddle point. To address this challenge, I introduce Gaussian Process Implicit surfaces (GPIS), which eliminates the need for searching corresponding points. Instead, it considers point registration as a surface-to-surface approach. I formulate the point estimation problem as a constrained manifold optimization problem. Obtaining a complete view of the point cloud representing an object is highly challenging in real-world applications due to limited sensor resolution, occlusion, and camera angles. In this thesis, I propose a transformer-based point cloud completion method, inspired by the manifold idea that assumes points from the same object layer lie on the same manifold layer. This approach significantly aids in recovering incomplete point clouds, greatly assisting grasp planning to find suitable grasp poses.

The final core component of this thesis is grasp planning. RGB-based grasp planning constrains the grasp in the vertical direction, simplifying the pick and place task significantly. However, the nature of 3/4-DOF constraints limits the integration of motion planning, as motion planning algorithms aim to explore all possible directions to generate an optimal motion trajectory. This limitation has motivated researchers to explore a more general approach, known as 6-DOF grasp, which allows for grasping in arbitrary directions. To address the grasp planning problem, this thesis proposes two methods. In industrial applications, objects are often represented as CAD models, which provide valuable geometry information for finding suitable grasp poses. I utilize the Gaussian process implicit surface to describe the object by uniformly sampling the CAD models. I incorporate the Alternating Direction Method of Multipliers (ADMM) and Bayesian optimization techniques to optimize an appropriate grasp pose. Moreover, considering the significant advancements in 6-DOF grasp learning networks, I also explore the application of transformer-based networks for grasping. I introduce my LiePFormer-GraspNet network, which predicts grasps directly from the point cloud input.

In summary, this work makes valuable contributions to the fields of trajectory planning, point cloud-based environment perception, and grasp planning, without being restricted to a specific robot system.

Zusammenfassung

Ein modernes Robotersystem ist ein multidisziplinäres Produkt, das von modernen Ingenieurmethoden bis zur Umweltwahrnehmung reicht. Es ist ein wesentlicher Bestandteil der Realisierung der Roboterautomatisierung. Meine Motivation besteht darin, zu zeigen, dass autonome Roboteraufgaben in Bezug auf Trajektorienplanung, Umgebungserfassung und Greifplanung beschrieben und gelöst werden können. Ansätze, die auf Teach-In-Konzepten oder vordefinierten Parametern basieren, führen zu langen Entwicklungszyklen, die für flexible und modulare kleine und mittelständische Unternehmen ungeeignet sind. Um menschliche Eingriffe zu minimieren und den Zweck der Automatisierung zu erreichen, hat mich dies außerdem dazu veranlasst, alle Aspekte der Robotertechnologie weiter zu verbessern. In dieser Arbeit werde ich das intelligente Robotersystem entwickeln und verbessern, indem ich die Trajektorienplanung, Umgebungswahrnehmung und Greifplanung optimiere.

Die Trajektorienplanung ist die erste Komponente, die ich basierend auf Beobachtungen aktueller Robotersysteme verbessern muss. Im Gegensatz zum modernsten Ansatz, der nur die kinematischen Randbedingungen der Beschleunigung berücksichtigt oder splinebasierte Ansätze verwendet, nutzen wir als Grundmodell das weit verbreitete trapezförmige Beschleunigungsprofil (Sieben-Segment-Modell) und erweitern es für Szenarien mit mehreren Wegpunkten. Die Rechenkomplexität des trapezförmigen Beschleunigungsmodells mit mehreren Wegpunkten steigt exponentiell mit der zunehmenden Anzahl von Wegpunkten, die unter Verwendung eines kollisionsfreien Bewegungsplanungsalgorithmus erzeugt werden. Um die Rechenkomplexität zu reduzieren, folge ich dem Prinzip der modellprädiktiven Steuerung, um das Trajektorienoptimierungsproblem zu lösen, indem ich es in eine Reihe von Teilproblemen zusammen mit einem Satz nichtlinearer Nebenbedingungen zerlege. Wir lösen jedes Teilproblem iterativ, wodurch die Komplexität deutlich reduziert werden kann. Darüber hinaus erweitern wir die Standardanwendung zur Trajektorienplanung um das Trajektorien-Blending-Szenario, das darauf abzielt, das Überschwingen der Trajektorie zu reduzieren und weiterhin zur Kollisionsvermeidung genutzt werden kann.

Die Techniken der Umgebungswahrnehmung ermöglichen eine autonome Robotik. Diese Arbeit konzentriert sich hauptsächlich auf eine 3D-Welt, die die Realität semantisch beschreiben kann. Ich präsentiere neue Algorithmen für die Klassifizierung von 3D-Objekten, die Schätzung von Positionen und die Vervollständigung von 3D-Punktwolken im Bereich der Computer Vision innerhalb eines Deep-Learning-Ansatzes, um die Unzulänglichkeiten aktueller Methoden anzugehen. Aufgrund des inhärenten Problems der Rotationseigenschaft bei der Klassifizierung von 3D-Punktwolken erfordern die meisten modernen Deep-Learning-basierten Ansätze einen großen Datensatz, um das Problem der Rotation zu mildern. Ich schlage eine neue Methodik vor, indem ich das Konzept des rotationsinvarianten Deskriptors der sphärischen Harmonischen einführe, das den Trainingsdatensatz drastisch reduzieren und den Stand der Technik übertreffen kann. Im Bereich der Posenschätzung dominieren iterative Nearest-Point (ICP)-basierte Ansätze. Die Suche nach entsprechenden Punkten im ICP-basierten Algorithmus erfordert jedoch eine sehr gute anfängliche Schätzung. Der Iterationsprozess kann ansonsten leicht in einem lokalen Optimum oder einem Sattelpunkt stecken bleiben. Um dieses Problem zu mildern, verwende ich die Gaussian Process Implicit Surfaces (GPIS), die die

Suche nach entsprechenden Punkten entfernen können und die Punktregistrierung als einen Oberfläche-zu-Oberfläche-Ansatz betrachten. Wir stellen das Punktschätzungsproblem als ein Optimierungsproblem auf einer Beschränkungsmannigfaltigkeit dar. Basierend auf Beobachtungen aus realen Anwendungen ist es aufgrund begrenzter Sensorauflösungen, Okklusion und Kamerawinkeln schwierig, eine vollständige Ansicht einer Punktwolke eines Objekts zu erhalten. Diese Arbeit schlägt eine transformatorbasierte Methode zur Vervollständigung von Punktwolken vor, die die Idee aus der Mannigfaltigkeitstheorie übernimmt, dass alle Punkte aus gleichen Objektschichten derselben Mannigfaltigkeitsschicht angehören. Der vorgeschlagene Ansatz kann die unvollständige Punktwolke wiederherstellen, was die Greifplanung bei der Suche nach einer geeigneten Lösungsgreifpose erheblich unterstützen kann.

Die letzte Kernkomponente dieser Arbeit ist die Greifplanung. RGB-basierte Greifplanung schränkt das Greifen in vertikaler Richtung ein. Diese Art des Greifens kann das Problem der Pick-and-Place-Aufgabe dramatisch vereinfachen. Die Beschränkung auf 3/4-DOF reduziert jedoch das Potenzial zur Integration in eine Bewegungsplanung, da diese Algorithmen versuchen, alle möglichen Richtungen zu untersuchen, um eine gute Bewegungsbahn zu erzeugen. Aufgrund dieser Einschränkung wurden in der Forschung allgemeinere Ansätze mit sechs Freiheitsgraden untersucht, die einen Griff in beliebiger Richtung ermöglichen. Diese Dissertation schlägt zwei Methoden vor, um das Greifplanungsproblem anzugehen. In industriellen Anwendungen werden die meisten Objekte als CAD-Modelle dargestellt. Daher kann ich diese Geometrieinformationen verwenden, um eine Greifpose zu finden. Ich verwende die Gaussian Process Implicit Surface, um das Objekt zu beschreiben, indem ich CAD-Modelle gleichmäßig abtaste und die Alternating Direction Method of Multipliers (ADMM) und Bayes’sche Optimierung integriere, um eine geeignete Greifpose zu optimieren. Darüber hinaus untersuchen wir mit den jüngsten massiven Fortschritten in graphbasierten Lernnetzwerken mit sechs Freiheitsgraden auch die Struktur des transformatorbasierten Netzwerks zum Greifen. Wir schlagen unser LiePFormer-GraspNet-Netzwerk vor, um das Greifen durch direktes Einspeisen der Punktwolke vorherzusagen.

Zusammenfassend liefert diese Arbeit wertvolle Beiträge auf dem Gebiet der Trajektorienplanung, der punktwolkenbasierten Umgebungswahrnehmung und der Greifplanung, die unabhängig von einem spezifischen Robotersystem sind.

Acknowledgement

I wish to begin by extending my deepest gratitude to my esteemed supervisor, Prof. Dr. Alois Knoll, and my mentor, Prof. Dr. Markus Rickert. Their unwavering guidance and the privilege of conducting my research within their esteemed group have been instrumental in shaping the trajectory of this thesis. Prof. Torsten Kröger deserves special acknowledgment as my second advisor, generously investing his time and demonstrating genuine interest in my work.

Heartfelt appreciation is extended to Alexander Perzylo, Dr. Nikhil Somani, Andrej Pandgeric, and all other colleagues and students who have enriched this journey with collaborative excellence.

Special recognition is due to Amy Bücherl and Ute Lomp for their exceptional support and seamless cooperation in managing administrative tasks.

I am sincerely thankful to Prof. Dr. Christian Ott and Prof. Dr. Minjun Kim from the German Aerospace Center for introducing me to the enthralling domain of robotics, sparking a passion for a world brimming with intelligence.

A distinct acknowledgement is reserved for Prof. Dr. Markus Rickert, whose meticulous proofreading and invaluable feedback have significantly elevated the quality of this thesis.

My profound gratitude extends to my beloved wife, Ziwei Jiang, whose unwavering support and encouragement have been a constant wellspring of strength, empowering me to resolutely pursue my dreams. I would also like to express a heartfelt thank you to my son Jensen Yi Lin, whose presence and understanding have brought joy and inspiration to my academic journey.

Lastly, heartfelt appreciation is expressed to my friends and family, with special mention to my parents, for their steadfast support and encouragement throughout the challenging journey of completing this thesis.

Garching, April 04, 2022 *Jianjie Lin*

Contents

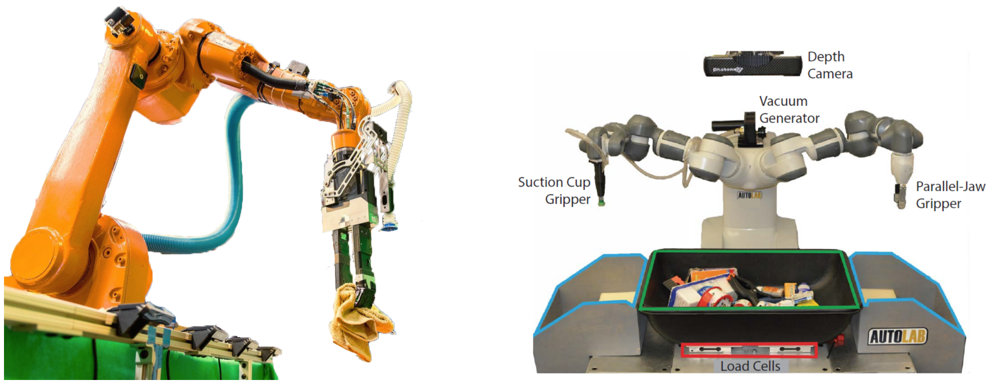
Chapter 1
Introduction

With the progress of society and the gradual development of automation, robots have touched every aspect of our life. The shape of an industrial robot manipulator can be seen everywhere, from factories producing a small industrial parts to the research and development of aircraft. Moreover, the rapid growth of robotic systems has brought a lot of convenience to people and made many difficult, tedious, repetitive tasks simpler and automated. In recent years, robot research has become increasingly powerful and stable with advanced hardware devices, improved sensor accuracy, and the in-depth exploration of the theory and practical application of robots by previous related works. However, there are still many unsolved problems in the area of robotics. Many researchers are trying to combine diverse fields to solve some tricky practical applications. For the same purpose, this thesis aims to develop and improve the algorithms in those fields, from planning, to environment perception, to enable an intelligent robotic system. The algorithms proposed in this work cover many different disciplines, from linear algebra, geometry, optimization algorithms, and software architecture to deep learning. They constitute the critical idea of the whole work proposed in this dissertation.

1.1 Perspectives

Intelligent agents should be capable of perceiving their environment and interacting with objects, such as moving, grasping and placing an object or using more available tools to achieve goals. In the robotic domain, it is formalized as a manipulation problem, and can be interpreted as a multi-step task consisting of sequential actions that involve interactions with objects in the scene. Research on robot grasping and manipulation dates back to 1970, a rime when science fiction classic Westworld staged intelligent human-shaped robots in a fictitious film. This science fiction tells a world in which human guests and robots have no essential differences in behavior and emotional expression, except the robot grasping is not yet perfect. The challenge of robotic grasping is evident. Consider a pick-and-place problem in the warehouse scenario, illustrated in Fig. 1.1: the robot firstly needs to perceive the environment for recognizing and localizing the desired object, then determine a grasp configuration for picking the object. In the next step, it is required to pick up the object in a robust fashion, and move it to a new location considering the kinematic constraints. The whole process needs to ensure no part of the robot collides with the environment. This simple scenario can become extremely challenging due to the infinite variability of real world. For example, if the lighting condition is not good enough for the vision system to detect the objects; if the grasped objects lack the physical information such as weight, size, surface texture, etc; if the path to the goal position is blocked by some obstacles; if other objects occupy the goal position, etc. Those uncertainties and unpredictability in the real world make robust manipulation extremely difficult.

Figure 1.1: Concept: advanced robots working in a warehouse (Image source from PYMNTS)



(a) (b)

Figure 1.2: Amazon Robotics Challenge: (a) is the winner of Amazon Robotics Challenge 2017 and was designed by the MIT MCube Lab and the Princeton Vision and Robotics Group, (b) is proposed with Dex-Net4 and was designed by the Berkeley AUTOLAB.

In the past decade, multiple robotic grasping and manipulation competitions involving robot manipulations of varying degrees of complexity have been held to advance research in the field. Those competitions had several different focuses, such as pure grasping, manipulation, assembly, or even mobile robotic operation. Amazon saw the potential in robotics that is targeted at ensuring automated customer order placement and product delivery tasks. In 2015, it launched the Amazon Robotics Challenge [1], which became known as the Amazon Picking Challenge (Fig. 1.2)s. This challenge aims to bring robotics researchers from industry and academia together to solve as many critical questions as possible about how robots can robustly grasp and handle objects. The automated warehouses from Amazon have already succeeded in removing the walking and searching for objects, but it still remains a challenge

[1]http://pwurman.org/amazonpickingchallenge/2015/

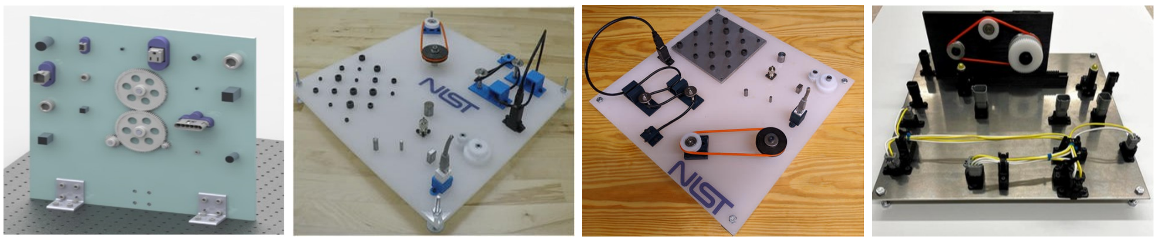


Figure 1.3: The manipulation task board of the Robotic Grasping and Manipulation Competitions at IROS 2017-2021

to pick a cargo automatically. Another ongoing robotic competition is held by IROS [2,3,4,5], shown in Fig. 1.3, which involves multiples tracks, and the difficulty of each task also increased year by year. This also witnessed the progress over the past decade in machine learning, computer vision science. Moreover, the updated hardware has enabled robots to handle a more complex set of objects robustly. An intelligent robot system with the desired outcomes and behaviors needs to be designed from several perspectives, such as perception, planning, and control. The following sections will explain the key ingredients that motivate this work.

1.1.1 Robot motion

Nowadays, human activities in many areas are gradually supported or replaced by robots due to the incredible versatility and flexibility of robots, such as fire search and rescue activities, space exploration, or cargo transporting. Path planning and trajectory planning are the key ingredients of robotics (or more generally the automation domain), which attempts to move a robot from the initial state to the destination position while avoiding collision with obstacles in the environments. Path planning merely generates a series of geometries waypoints without considering any specified time law. Trajectory planning aims to assign a time law to the geometric path under consideration of kinematic constraints (position, velocity, acceleration, jerk, etc.). Therefore, path planning typically precedes trajectory planning. In this work, I focus the efforts on optimizing a trajectory where the waypoints is generated by the off-the-shelf path planning module. The output of the trajectory planner is represented in the form of a sequence of values of position, velocity, acceleration and jerk (the derivative of acceleration), where these kinematic parameters can be interpreted in terms of operational space or joint space. In most cases, the trajectories are planned in the joint space since the robotic control actions are executed on joints. Planning in the operational space requires the kinematic inversion of the transform parameter defined on the end-effector, which imposes more complexity. However, the planned trajectory in joint and operational spaces does not exhibit the same geometric properties due to the nonlinearities introduced from the forward kinematics. For example, a straight-line trajectory in joint space is typically not straight in the operational space. No matter in which space, the laws of motion resulting from the trajectory planning must not generate forces and torques at the joints that are not compatible with the given constraints. In particular, the generated trajectories should have the continuity of joint acceleration to obtain a limited jerk. Such a trajectory can be used to avoid exces-

[2]http://www.rhgm.org/activities/competition_iros2017
[3]https://rpal.cse.usf.edu/competition_iros2019
[4]https://rpal.cse.usf.edu/competition_iros2020
[5]https://rpal.cse.usf.edu/competition_iros2021

sive accelerations of the actuators and vibrations of the mechanical structure and avoid the excitation of the resonance frequencies of the robot. Three different optimality criteria are typically applied when generating a trajectory: minimum execution time, minimum energy, and minimum jerk. Among them, the minimum time trajectory, which can reduce the length of production cycles, attracts a lot of consideration in the robotic literature. However, the complexity of optimizing a time-optimal trajectory is significantly increased with the number of intermediate points and level-of-detail (i.e., trajectories represented by a curve whose derivatives are continuous up to a certain degree). Robot manufactures have already offered dedicated modules to allow direct specification of trajectories using the CAD models or given waypoints. Various algorithms integrated into these dedicated modules can be used to generate a series of offline trajectories, where those algorithms are written using the same set of vendor-specific programming languages in the teach-in method. However, the standard trajectory algorithms offered by the many manufactures consider the constraints only up to the acceleration but neglect the jerk, which may lead to considerable wear in the mechanical structure of the robot. The proposed algorithms can work as complementary to algorithms offered by the manufactures if the real time interface with respect to the position, velocity, etc is provided by those manufactures, such as Universal robot. They are used to generate a trajectory by passing through or blending over waypoints under consideration jerk limitations. I use the trapezoidal acceleration profile, also denoted as the seven-segment profile, as the fundamental trajectory profile, which is widely used in robotic trajectory research [8, 77].

1.1.2 Perceiving the environment

Just like the sensory organs in humans responsible for tasks such as vision, hearing, touch, taste, or smell, robot perception plays a crucial role in enabling a robot to complete tasks in complex environments. Furthermore, robot perception grounds the application of robots in the real world and empowers robots to explore, navigate and manipulate objects based on sensory data. They penetrate every stage of applications, starting from high-level intelligent robot manipulation down to the lower-level closed-loop control. A significant amount of related work has been dedicated to solving the various vision challenges over the past years. Computer perception has also arrived at a new milestone, along with the substantial improvement in 3D acquisition technologies and reduced prices of commercial sensors. However, addressing the amount of information acquired by multiple sensors is extremely difficult due to noisy and redundant sensory data. In addition, the robot can be placed in varying environment, and a less constrained environment introduces more challenges to perceive. The world itself is not unique but ambiguous: some objects are easily confused since they share a similar geometric outline, such as lemon and normaler tennis ball; Some objects can not be easily distinguished in RGB images in some shooting angles, e.g., remote control and cell phone when facing down; some objects are partially obscured by other objects, so their actual shape is unknown; even the detection of the same object at a different posture (position and orientation) in the same world coordinate can fail. All these factors contribute to the difficulty of understanding the world from the perspective of perception.

Classical computer vision topics such as object detection and pose estimation are the foundations of vision-based robot manipulation, which attracts a lot of attention, especially in the scope of deep neural networks. Traditional approaches use mesh surfaces or combinations of primitive shapes to describe three-dimensional objects and deploy hand-crafted features or constraints to interpret the interactions between objects. The representation in the form of primitive shapes, due to their inherent geometric nature, can alleviate the data processing inefficiency problem. Deep learning-based shape analysis and synthesis algorithms are developed mainly for voxel grids, point clouds, and implicit functions and then extract ISO surfaces marching cubes for visualization. I dedicate the efforts in the computer vision domain to recognize and estimate physical instances of objects which are acquired via 3D camera and stored as a raw point cloud. The proposed algorithm for the object detection network is not limited to a standard classification task but also enables rotation-invariant detection, which is lacking in most other point cloud-based networks. Moreover, I propose a novel pose estimation algorithm using Gaussian Process Implicit Surfaces, which enables the utilization of a Lie-algebra manifold optimization and avoids the corresponding search step in the classical iterative closest point algorithm. To facilitate grasp planning, I introduce a shape completion algorithm to mitigate the problem that robots capture only partial point cloud information with a single depth camera.

1.1.3 Robot manipulation and grasping

Robot hands are often inspired by nature, in particular prehensile, multi-fingered appendages situated at primates' upper extremities (arms). The hand is denoted as a grasping organ that exhibits excellent mobility and flexibility in the digits and the whole organ. The ultimate goal of robot hand research is to have human-like dexterity, which can perform versatile and dexterous grasping tasks on arbitrary objects, seamlessly adapt to object properties and the task's requirements, and transfer learned manipulated skills to arbitrary objects. Over the past decades, a huge amount of work has been dedicated to the research on robotic manipulation and grasping, with impressive progress. However, there is still a long way to go before arriving atthe ultimate goal.

Object manipulation relies on reliable path and trajectory planning as well as adequate perceptual capabilities, which is a prerequisite for many industrial applications, such as flexible manufacturing and construction in collaboration with human experts. The structure environment is already challenging due to complexity of the associated high-dimensional state space, such as the objects appearance, position, texture, grasp force, etc. Robotic systems have become very mature in performing repetitive industrial tasks such as pick-and-place operations, car assembly, welding, or spray painting. These applications typically are created using procedures such as the teach-in method, where a series of instructions are formulated with a vendor-specific programming language. The tightly controlled environment simplifies the robotic manipulation problem. However, the world is not a predictable assembly line. The unstructured environment confronted more challenge than the structured environment in that the object properties are not prior knowledge and first need to be obtained using external sensors. Apart from ambiguous and uncertain perceptual information, physical interaction with the outside world faces an inherent uncertainty in how objects react to touch. Some robotic companies improve the manipulation skills via improved hardware designs, such as designing grippers ranging from pincer-like appendages (parallel grippers) to human-like hands (multi-fingered grippers) and equip them with tactile sensors. Rapid advances in machine learning push forward the progress in grasping as well. Therefore, robotics researchers use machine learning or data-driven approaches to empower the robots to identify and grasp objects independently. In this work, two algorithms are proposed for handling the grasping problems. In the first approach, a Bayesian optimization along with the Gaussian Process Implicit Surfaces is utilized to figure out the 6D grasp configuration. In the second approach, a point cloud transformer neural network is proposed to process the objects, which will be further used to predict a 6D grasp configuration.

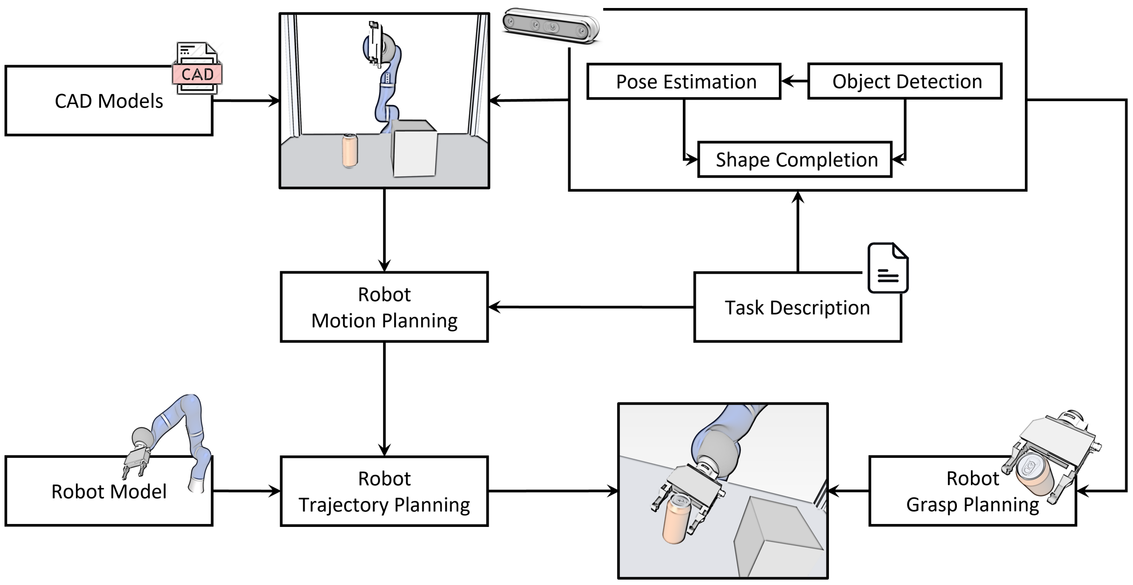


Figure 1.4: Combining the different key ingredients of this work to create a intelligent robotic system architecture

1.2 Architecture for an intelligent robot system

Considering all aspects involved in designing an intelligent robotic system, this work aims to integrate all aforementioned key ingredients to form a final robotic system structure. Recall the scenario in the warehouse as depicted in Fig. 1.1: A robot is assigned a task to grasp a package and place it on a shelf while avoiding any obstacle. I propose and design an intelligent robot system as illustrated in Fig. 1.4, to achieve this warehouse-like transporting task, where the system structure in principle follows the sense-plan-act paradigm. Compared with standard industrial robot applications, such as car-assembly, which only execute pre-instructed offline programs without interacting with human coworkers, an intelligent robot system that respond to dynamic environments requires artificial intelligence support. The task description in the traditional teach-in method requires several additional aspects to accomplish the task, such as specifying the position and orientation of objects or the grasp configuration concerning the object. In contrast, only the kinematic constraints and desired objects are required as input in the task description for an intelligent robot system. The robot model is fundamental for controlling the robots, and it allows the robot to move along the geometric path, that is generated by the path planning module. The geometric path in the joint space or operational space is fed into the trajectory planning module. Then a trajectory is optimized in terms of predefined criteria by obeying the kinematic constraints. Depth cameras, or more general 3D sensors, are used to capture objects to ensure that the intelligent robotic system is capable of reacting dynamically to the environment. Three main computer topics are addressed in this system. A deep learning-based object classification neural network is used to identify the object requested in the task description. The pose estimation algorithm extracts the object from the raw sensory data and estimates its coordinates in the world, which will then fed to the path and grasp planning module. An additional shape completion neural network is introduced in the sense paradigm to enhance grasp planning performance, which aims to enlarge the grasp configuration solution space by completing the partial point cloud to additional geometric information. The grasp planning module is contained in the plan-act-paradigm. The rapid advance in machine learning pushes forward

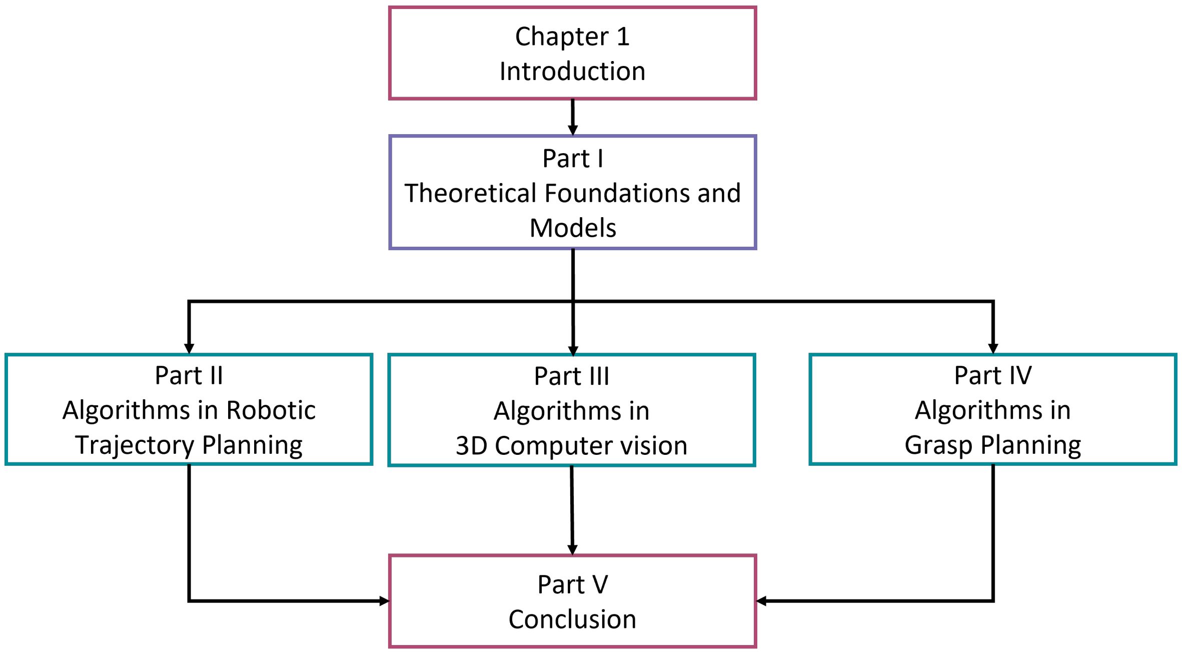

Chapter 1
Introduction
Part I
Theoretical Foundations and Models
Part II
Algorithms in Robotic Trajectory Planning
Part III
Algorithms in 3D Computer vision
Part IV
Algorithms in Grasp Planning
Part V
Conclusion

Figure 1.5: Outline of thesis

the progress in robot manipulation. A data-driven and optimization-based grasp planning module is employed to realize the manipulation task.

1.3 Contributions and Outlines

This work is structured in three parts: Introduction, robotic foundations, and algorithms as demonstrated in Fig. 1.5. Part I explains the essential aspects and mathematical formulations in developing the algorithms in a fully autonomous system. They are the core components in a modular design and provide the foundations for later concepts. Furthermore, I divide the sense-act-plan structure into three parts. In Part II, I develop two algorithms in the trajectory planning domain, and compare them with various state-of-the-art approaches. Their strengths and limitations in the context of time-optimal, straight-line performance and efficiency are discussed and evaluated. The developed and modified trapezoidal acceleration-based motion profiles in this thesis demonstrate their benefits in various challenging industrial motion planning scenarios. Furthermore, one vital component in the context of the sense-plan-act paradigm is depicted in Part III, where three different computer vision algorithms are developed. The first scenario focuses on pose estimation utilizing the concept of Gaussian Process Implicit Surfaces, with which I can describe the grasped object as an implicit surface. The second one demonstrates a point cloud-based rotation invariant algorithm for robust object classification. To further enhance the robotic manipulability, I introduce a point cloud completion approach enabling a full object representation. In Part IV, two different grasp planning algorithms are demonstrated in two intuitive and modular systems for industrial application. The first proposed algorithm utilizes the machine learning approach for optimizing a grasp configuration. In contrast, a transformer-based grasp detection approach is presented for enhancing the grasp capability in the case of an unknown scenario.

Part I

Theoretical Foundations and Models

Chapter 2

Mathematical Preliminaries and Robotics Foundations

Before discussing the topics of trajectory planning, environment perception, and grasp planning, some basic knowledge and principles need to be introduced that will be used throughout the paper. In addition, this section aims to provide enough specific mathematical details to derive algorithms beyond the scope of this thesis using the basic concepts and knowledge examined here.

2.1 Rigid Body Motion

A rigid body [145] is a solid body in which no deformation is conducted, and a rigid motion of an object is a continuous movement that preserves the distance between points regardless of external forces or moments exerted on it at all times. The motion of a single rigid body is considered as the relative motion of a body-fixed coordinate frame B with respect to an inertial frame S, where the inertial frame S by convention is presented as the base frame or world frame in the robotic application. Furthermore, a minimum of six numbers is required to specify the position $\mathbf{p} \in \mathbb{R}^3$ and orientation $\mathbf{R} \in \mathbb{R}^3$ of a rigid body in the three-dimensional Cartesian space (physical space). The translational position $\mathbf{p}$ is defined as composed of a set of three orthonormal axes $\mathbf{o}_x, \mathbf{o}_y, \mathbf{o}_z$ with its respective distance p_x, p_y, p_z along the axes of a corresponding reference frame:

$$\mathbf{p} = p_x \mathbf{o}_x + p_y \mathbf{o}_y + p_z \mathbf{o}_z \tag{2.1}$$

The notation $\mathbf{p}_{sb}$ is used to indicate the translational motion from the origin of the inertial frame S to the origin of reference frame B, and orthogonal matrix $\mathbf{R}_{sb} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3] \in \mathbb{R}^{3\times3}$ is applied to describe the rotation of frame B with respect to the world frame S, where $\mathbf{r}_i$ is the unit vector, and collinear to the i^{th} coordinate direction of frame B. Since the columns of R are mutually orthonormal, it follows

$$\mathbf{r}_i^{\mathrm{T}} \mathbf{r}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{2.2}$$

The determinant of $\mathbf{R}$ is defined as:

$$\det(\mathbf{R}) = \mathbf{r}_1^{\mathrm{T}} (\mathbf{r}_2 \times \mathbf{r}_3) \tag{2.3}$$

In convention, all considered coordinates frame follows the right-hand principle. Therefore, the rotation matrix has the determinant $\det(\mathbf{R}) = +1$. More generally, the special orthogonal group SO(n) is used to describe the set of all such rotation matrices, and it has the following property:

$$\mathrm{SO}(n) = \{\mathbf{R} : \mathbf{R} \in \mathbb{R}^{n\times n}, \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}, \det(\mathbf{R}) = 1\} \tag{2.4}$$

where $n = 3$ with $\mathrm{SO}(3) \in \mathbb{R}^{3\times 3}$ is rotation matrix defined in three-dimensional space.

A group G is defined as a set of elements and an binary operation $\circ$ on two elements by satisfying the following properties:

- **Closure**: If g_1, $g_2 \in G$, then $g_1 \circ g_2 \in G$
- **Associativity**: If g_1, g_2, $g_3 \in G$, then $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$
- **Identity**: There exists an identity element, e, such that $g \circ e = e \circ g = g$ for every $g \in G$
- **Inverse**: For each $g \in G$, there exists a (unique) inverse, $g^{-1} \in G$, such that $g \circ g^{-1} = g^{-1} \circ g = e$

The SO(3) is known as a rotation group under the operation of matrix multiplication. Therefore, I can define the binary operator $\circ$ as the matrix multiplication, and the identity element e is the identity matrix. Furthermore, it can be shown that a rotation matrix represents a rigid body transformation, since it preserves the distance and orientation. However, the SO(3) is not closed under the operation of addition, since adding two rotation matrix can not result in a valid rotation matrix:

$$\mathbf{R}_1, \mathbf{R}_2 \in \mathrm{SO}(3) \nRightarrow \mathbf{R}_1 + \mathbf{R}_2 \in \mathrm{SO}(3) \tag{2.5}$$

Note that, the zero matrix is not a valid rotation matrix: $\mathbf{0} \notin \mathrm{SO}(3)$. I can draw a conclusion that SO(3) is not a vector space of $\mathbb{R}^{3\times 3}$

2.1.1 Euler angles representation for rotation matrix

Rotation matrix [33, 163, 178] has nine elements, which imposes six independent constraints since the columns of R are mutually orthonormal. Thereby, only three elements are necessary to describe a rotation matrix. An essential rotation representation is a rotation about one of the axes of a coordinate system, such as about the x-, y-, or z-axis. And it results in the rotation matrix with $\cos(\theta) = c$ and $\sin(\theta) = s$ in the sense of right-hand rule as

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}, \quad \mathbf{R}_y(\theta) = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}, \quad \mathbf{R}_z(\theta) = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

The other rotation matrix can be obtained from these three using matrix multiplication

$$\mathbf{R} = \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_x(\delta) \tag{2.7}$$

This order of matrix multiplication is so called yaw, pitch, and roll rotation representation. The other order of matrix multiplication such as ZYZ' or ZXZ' are also frequently used in the robotic application. A known Gimbal Lock problem in the Euler angle presentation can result in a loss of a degree of freedom. Besides, this representation of rotations has another disadvantage that small changes in rotation can result in significant angular changes in the proximity of singularities.

2.1.2 Quaternion for rotation matrix

Unit quaternions [29, 85, 112, 158] is an alternative parameterization of rotation that can be used to prevent those problems as mentioned above by introducing a normal unit constraint. Formally, a quaternion $\mathbf{Q} \in \mathcal{H}$ is a 4-tuple written in the form

$$\mathbf{Q} = q_0 + q_1 i + q_2 j + q_3 k,\ q_i \in \mathbb{R}, i = 0, \cdots, 3 \tag{2.8}$$

where $q_0 \in \mathbb{R}$ is the scalar component of $\mathbf{Q}$, and $\mathbf{q} = (q_1, q_2, q_3) \in \mathbb{R}^3$ is the vector component. The symbols i, j, k satisfy the following identities with the quaternion multiplication operator $(\cdot)$:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2.9a}$$
$$i \cdot j = k,\ j \cdot i = -k \tag{2.9b}$$
$$j \cdot k = i,\ k \cdot j = -i \tag{2.9c}$$
$$k \cdot i = j,\ i \cdot k = -j \tag{2.9d}$$

Multiplication of these symbols i, j, k are distributive and associative but not commutative. The magnitude of a quaternion $\mathbf{Q}$ can be described as:

$$\|\mathbf{Q}\|^2 = \mathbf{Q} \cdot \mathbf{Q}^\star = q_0^2 + q_1^2 + q_2^2 + q_3^2 \tag{2.10}$$

where $\mathbf{Q}^\star$ is the conjugate of a quaternion $\mathbf{Q}$ as $(q_0, -\mathbf{q})$. It can be easily derived that the reciprocal of a non-zero quaternion $\mathbf{Q}^{-1}$ is defined as $= \frac{\mathbf{Q}^\star}{\|\mathbf{Q}\|^2}$. The product between two quaternions is a combination of inner product and cross product between two vectors: Let $\mathbf{Q} = (q_0, \mathbf{q})$, $\mathbf{P} = (p_0, \mathbf{p})$ be quaternions, and the their product is defined as

$$\mathbf{Q} \cdot \mathbf{P} = (q_0 p_0 - \mathbf{q}\mathbf{p}, q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p}) \tag{2.11}$$

The unit quaternion is a subset of quaternion, by setting a unit magnitude of $\mathbf{Q}$ as $q_0^2 + \mathbf{q}^2 = 1$. I can further reformulate the quaternion with $\cos(\theta)^2 = q_0^2$, $\sin(\theta)^2 = \mathbf{q}^2$. Therefore, I have the unit quaternion in the form of

$$\mathbf{Q} = \left(\cos(\frac{\theta}{2}), \mathbf{u}\sin(\frac{\theta}{2})\right) \tag{2.12}$$

where $\mathbf{u} \in \mathbb{R}^3$ represents the unit axis of rotation with its corresponding angle $\theta \in \mathbb{R}$. It is straightforward to derive the angle θ and $\mathbf{u}$ for a quaternion $\mathbf{Q} = (q_0, \mathbf{q})$ with

$$\theta = 2\cos^{-1} q_0, \qquad \mathbf{u} = \begin{cases} \frac{\mathbf{q}}{\sin(\theta/2)}, \text{if} \quad \theta \neq 0 \\ 0, \text{otherwise} \end{cases} \tag{2.13}$$

A quaternion rotation is defined as

$$\begin{bmatrix} 0 \\ \mathbf{z}' \end{bmatrix} = \mathbf{Q} \begin{bmatrix} 0 \\ \mathbf{z} \end{bmatrix} \mathbf{Q}^{-1} \tag{2.14}$$

where $\mathbf{z} \in \mathbb{R}^3$ is defined in the global coordinate system, and $\mathbf{z}' \in \mathbb{R}^3$ is the same vector defined in the body fixed coordinates. This unit quaternion rotation can be algebraically manipulated into a matrix rotation as $\mathbf{z}' = \mathbf{R}\mathbf{z}$, therefore, I can derive the rotation matrix in terms of unit quaternion with

$$\mathbf{R}(Q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 + q_3^2 & 2(q_1 q_2 + q_3 q_0) & 2(q_1 q_3 - q_2 q_0) \\ 2(q_1 q_2 - q_3 q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_1 q_0) \\ 2(q_1 q_3 + q_2 q_0) & 2(q_2 q_3 - q_1 q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.15}$$

The advantage of using unit quaternion for presentation a rotation matrix is that the representation of a rotation as a quaternion (4 numbers) is more compact than the representation as an orthogonal matrix (9 numbers). Furthermore, for a given axis and angle, one can directly construct the corresponding quaternion, and conversely, for a given quaternion one can easily read off the axis and the angle.

2.1.3 Lie algebra for rotation matrix

The Lie group $SO(3)$ represents a rotation matrix, and has its corresponding Lie algebra $\{\breve{\ } = \phi^\wedge \in \mathbb{R}^{3\times3} | \phi \in \mathbb{R}^3\}$, which is interpreted as a vector space. The operator $(.)^\wedge$ can further convert this vector ϕ to a skew-symmetric matrix Φ in eqn. (2.16).

$$\Phi = \phi^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3},\ \phi \in \mathbb{R}^3 \tag{2.16}$$

The inverse of the operator $(.)^\wedge$ is defined as $(.)^\vee$ and leads to $\phi = \Phi^\vee$. The exponential map will convert the Lie algebra $\mathfrak{so}(3)$ to the Lie group SO(3) utilizing the matrix exponential formulas: $\mathbf{R} = \mathrm{e}^{\phi^\wedge}$

$$\mathrm{e}^{\mathbf{A}} = \mathbf{I} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \cdots = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{A}^n \tag{2.17}$$

$$\mathbf{R} = \mathrm{e}^{\phi^\wedge} = \sum\nolimits_{n=0}^{\infty} \frac{1}{n!}\left(\phi^\wedge\right)^n = \mathrm{e}^{\theta\mathbf{a}^\wedge} = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{a}\mathbf{a}^T + \sin\theta\mathbf{a}^\wedge$$

where the angle θ is defined as $\|\boldsymbol{\phi}\|$ and the axis $\mathbf{a} = \frac{\boldsymbol{\phi}}{\theta}$ is interpreted as unit-length axis of rotation. The identity $\mathbf{I} \in \mathbb{R}^{3\times3}$ can be rewritten in term of $\mathbf{a}$ with $\mathbf{a}\mathbf{a}^{\mathrm{T}} - \mathbf{a}^\wedge\mathbf{a}^\wedge$. It can be shown that $\mathbf{R} = \mathrm{e}^{(\theta+2\pi m)\mathbf{a}^\wedge}$, which indicates that mutually isomorphic Lie groups have mutually isomorphic Lie algebras, but the converse does not necessarily hold. However, by limiting the angle of rotation as $|\theta| \leq \pi$, each rotation $\mathbf{R}$ in Lie group SO(3) has a unique solution θ in lie algebra $\mathfrak{so}(3)$, and the calculation can be done:

$$\begin{aligned} \mathrm{tr}(\mathbf{R}) &= \mathrm{tr}\left(\cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{a}\mathbf{a}^{\mathrm{T}} + \sin\theta\mathbf{a}^\wedge\right) && \text{(2.18a)} \\ &= \cos\theta \underbrace{\mathrm{tr}(\mathbf{I})}_{3} + (1-\cos\theta)\underbrace{\mathrm{tr}\left(\mathbf{a}\mathbf{a}^{\mathrm{T}}\right)}_{\mathbf{a}^{\mathrm{T}}\mathbf{a}=1} + \sin\theta \underbrace{\mathrm{tr}\left(\mathbf{a}^\wedge\right)}_{0} && \text{(2.18b)} \\ &= 2\cos\theta + 1 && \text{(2.18c)} \end{aligned}$$

It can be solved with

$$\theta = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) + 2\pi m \tag{2.19}$$

and I will pick a value which lies inside $|\theta| \leq \pi$, and the axis is solved based on the factor that $\mathbf{R}\mathbf{a} = \mathbf{a}$, namely $\mathbf{a}$ is the eigenvector of $\mathbf{R}$ with the eigenvalue is equal to 1.

2.1.4 Rigid body transformations

The spatial configuration of a rigid body is considered as a combination of translation $\mathbf{p}$ and rotation matrix $\mathbf{R}$, which can be denoted as special Euclidean group SE(3) :

$$\mathrm{SE}(3) = \{(\mathbf{p}, \mathbf{R}) \,|\, \mathbf{p} \in \mathbb{R}^3, \mathbf{R} \in \mathrm{SO}(3)\} \tag{2.20}$$

In a similar vein to the rotation matrix, a transformation matrix transfers the coordinates of a point from one frame to another. Let $\mathbf{p}_a$, $\mathbf{p}_b \in \mathbb{R}^3$ be the coordinates of a point $\mathbf{p}$ relative to frames A and B, respectively. The geometry relation between $\mathbf{p}_a$ and $\mathbf{p}_b$ can be described as:

$$\mathbf{p}_a = \mathbf{p}_{ab} + \mathbf{R}_{ab}\mathbf{p}_b \tag{2.21}$$

The mapping $g_{ab} = (\mathbf{p}_{ab}, \mathbf{R}_{ab})$ is specified as the configuration of the frame B relative to the frame A. Hence, The equation (2.21) is reformulated in the way of matrix multiplication as

$$\begin{bmatrix} \mathbf{p}_a \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{p}_{ab} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_b \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} \mathbf{p}_b \\ 1 \end{bmatrix} \tag{2.22}$$

where $\mathbf{T}$ is the homogeneous representation of $g_{ab} \in \mathrm{SE}(3)$. The inversion of $\mathbf{T}$ can be derived as

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}_{ab}^{\mathrm{T}} & -\mathbf{R}_{ab}^{\mathrm{T}}\mathrm{T}_{ab} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.23}$$

Similar to the SO(3), SE(3) is also not a vectorspace of $\mathbb{R}^{4\times4}$ [72, 156]. SO(3) and SE(3) are known to be matrix Lie group, which is also a differential manifold with the property that the group operations are smooth. The exponential map will convert the SE(3) to its corresponding lie algebra, denoted as $\mathfrak{se}(3)$, and the Logarithm map will vice versa.

2.2 Robotic Forward Kinematics

The kinematics of a robot manipulator [112, 159] can be schematically represented as a kinematic chain of rigid bodies (links) connected by means of revolute q or prismatic joints. In the case of revolute joints, the joint q describes the angle between two adjacent links while a prismatic joint is represented by the linear displacement between the two adjacent links. Notice that, the angle of revolute joint can also be thought as an element of the set S^1, a unit circle [121].

The joint space $\mathcal{Q}$ of a manipulator is composed of all possible values of the joint variables $\mathbf{q}$ of the robot, which can also be interpreted as configuration space:

$$\mathbf{q} = (q_1, \ldots, q_n)^{\mathrm{T}} \in \mathcal{Q} \tag{2.24}$$

The task of forward kinematic of a robot is to determines the configuration of tool frame, attached to end-effector, by given the joint configuration. The joint twist $\mathcal{E}$ [112] is described as in the case of resolute joint as

$$\mathcal{E} = \begin{bmatrix} -\omega_i \times q_i \\ \omega_i \end{bmatrix} \tag{2.25}$$

where $\omega_i \in \mathbb{R}^3$ is the unit vector aligned to the joint axis and q_i is the arbitrary point on the joint axis. By given the joint configuration $\mathbf{q}$ and the based reference $\mathbf{T}(0)$, I can have the forward kinematics map by using the product of exponentials formula

$$\mathbf{T}(\mathbf{q}) = \mathrm{e}^{\hat{\mathcal{E}}_1 q_1}\mathrm{e}^{\hat{\mathcal{E}}_2 q_2}\ldots\mathrm{e}^{\hat{\mathcal{E}}_n q_n}\mathbf{T}(0) \tag{2.26}$$

The exponential terms $\mathrm{e}^{\hat{\mathcal{E}}_i q_i} \in \mathrm{SE}(3), i \in [1, \ldots, n]$ has the following formula:

$$\mathrm{e}^{\hat{\mathcal{E}}_i q_i} = \begin{cases} \begin{bmatrix} \mathrm{e}^{\hat{\omega}_i q_i} & (\mathbf{I} - \mathrm{e}^{\hat{\omega}_i q_i})(\omega_i \times \mathbf{v}_i) + \omega_i \omega_i^{\mathrm{T}} \mathbf{v}_i q_i) \\ \mathbf{0} & 1 \end{bmatrix}, \text{for } \|\omega_i\| = 1 \\ \\ \begin{bmatrix} \mathbf{I} & \mathbf{v}_i q_i \\ \mathbf{0} & 1 \end{bmatrix}, \text{for } \|\omega_i\| = 0 \end{cases} \tag{2.27}$$

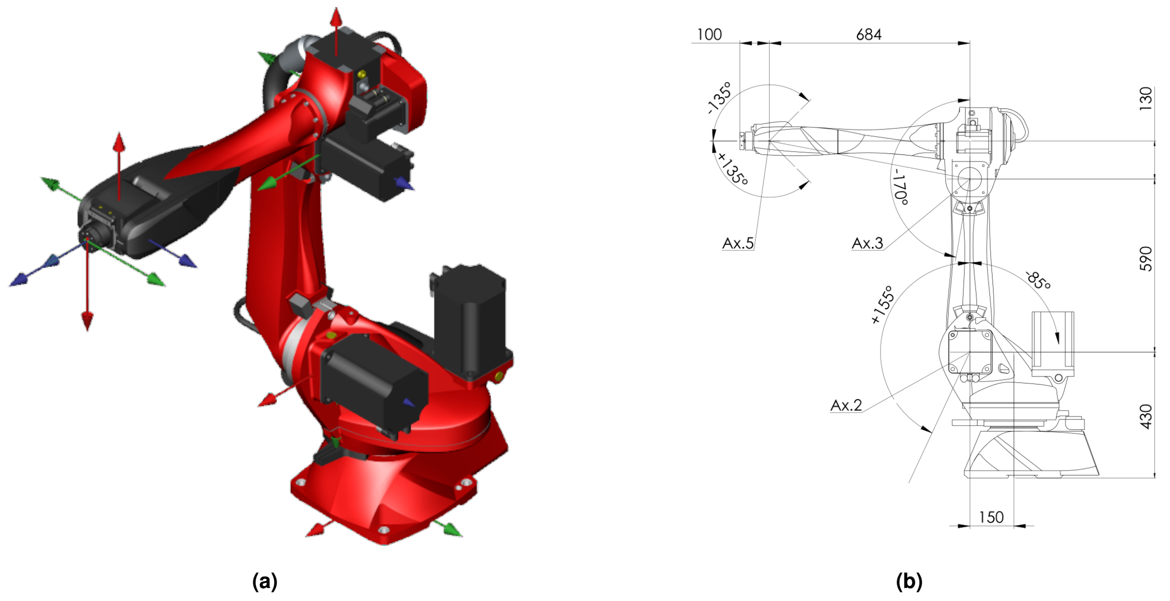


Figure 2.1: (a):A model of an industrial manipulator with six degrees of freedom. The coordinate systems of its links are shown with arrows in red, green, and blue for the corresponding x, y, and z axis [142].(b): Exact measurements of manipulator from CAD data provided by manufacturer. The image shows the distances between the joint axes required for a kinematic description of the robot.(Source from Robotic Library)

where $\mathbf{v}_i$ in the case of pure rotation is defined as $-\omega_i \times q_i$ [104, 112, 121], and the exponential term $e^{\hat{\omega}_i q_i}$ can be computed with the equation (2.18a)

2.2.1 Modeling with Denavit-Hartenberg parameters

By given the based frame S and a tool frame T, which can be arbitrarily attached to the link, the concept of twist shown in the previous section can provide a complete parameterization solution to obtain the forward kinematic, or more specially, the homogenous transformation matrix between these two frames. To simplify the representation of the forward kinematic of robots, a further conversion of parameterization method came into being. A commonly used convention for selecting frames of reference in robotics applications is the Denavit and Hartenberg (D-H) convention which was introduced by Jacques Denavit and Richard S. Hartenberg [32] and is the de facto standard in robotics. This convention presented the first minimal representation for a line with only four parameters compared to the six in a universal transformation. In this convention, each homogeneous transformation $\mathbf{T}_i$ is represented as a product of four basic transformations (pure rotation Rot and pure translation Trans) with

$$\mathbf{T}_{l_{i-1},l_i} = \mathrm{Rot}(z_{i-1}, q_i) \cdot \mathrm{Trans}(z_{i-1}, d_i) \cdot \mathrm{Trans}(x_i, a_i) \cdot \mathrm{Rot}(x_i, \alpha_i) \tag{2.28}$$

$$= \begin{bmatrix} \cos q_i & -\sin q_i & 0 & 0 \\ \sin q_i & \cos q_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_i & -\sin \alpha_i & 0 \\ 0 & \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\begin{array}{ccc|c} \cos q_i & -\sin q_i \cos \alpha_i & \sin q_i \sin \alpha_i & a_i \cos q_i \\ \sin q_i & \cos q_i \cos \alpha_i & -\cos q_i \sin \alpha_i & a_i \sin q_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{p} \\ \hline \mathbf{0} & 1 \end{array}\right] \tag{2.29}$$

where the four quantities q_i, a_i, d_i, α_i are parameters associated with link i and joint i. I consider this transformation as a combination of two axial screw transformations. In some books, the D-H parameterization can also be described as

$$\mathbf{T}_{l_{i-1},l_i} = \mathrm{Trans}(z_{i-1}, d_i) \cdot \mathrm{Rot}(z_{i-1}, q_i) \cdot \mathrm{Trans}(x_i, a_i) \cdot \mathrm{Rot}(x_i, \alpha_i)$$

due to the matrix multiplication of $\mathrm{Rot}(z_{i-1}, q_i) \cdot \mathrm{Trans}(z_{i-1}, d_i)$ is equal to $\mathrm{Trans}(z_{i-1}, d_i) \cdot \mathrm{Rot}(z_{i-1}, q_i)$. These four parameters q_i, a_i, d_i, α_i are denoted as joint angle, link length, link offset, and link twist, respectively. Furthermore, only the joint angle q_i is a variable, and rest of them are constant parameters based on the robot geometric description. As discussion in the book [104], by properly choice of the origin and the coordinate axes, the number of presentation of a homogenous transformation can be reduced to four. The equation (2.28) requires the axis x_i of a frame both intersecting and perpendicular to the axis z_{i-1}. The kinematics of the mechanism can be written as

$$\mathbf{T}_{st} = \mathbf{T}_{l_0,l_1}(q_1) \cdot \mathbf{T}_{l_1,l_2}(q_2) \ldots \mathbf{T}_{l_{n-1},l_n}(q_n)\mathbf{T}_{l_n t} \tag{2.30}$$

This equation (2.30) is a general formula for the forward kinematics map of an open-chain manipulator in terms of the relative transformations between adjacent link frames. An real implementation example is illustrated in the Fig. 2.1.

The relationship between the twist coordinates for the joints of a robot manipulator and the D-H parameter is not simple one-to-one mapping problem. As described in [112], I defined the $\mathcal{E}_{i,i-1}$ is the twist for the i-th link relative to the previous link frame. Then the following formula can be conducted as

$$\mathbf{T}_{l_{i-1},l_i}(q_i) = \mathrm{e}^{\hat{\mathcal{E}}_{i-1,i} q_i} \mathbf{T}_{l_{i-1},l_i}(0) \tag{2.31}$$

Therefore, the forward kinematic map can be described as

$$\mathbf{T}_{st}(\mathbf{q}) = \mathrm{e}^{\hat{\mathcal{E}}_{0,1} q_1} \mathbf{T}_{l_0,l_1}(0) \cdot \mathrm{e}^{\hat{\mathcal{E}}_{1,2} q_2} \mathbf{T}_{l_1,l_2}(0) \ldots \mathrm{e}^{\hat{\mathcal{E}}_{n-1,n} q_n} \mathbf{T}_{l_{n-1},l_n}(0) \tag{2.32}$$

Furthermore, I can rewrite the equation (2.32) as

$$\mathbf{T}_{st}(\mathbf{q}) = \mathrm{e}^{\hat{\mathcal{E}}_{0,1} q_1} \mathbf{T}_{l_0,l_1}(0) \mathrm{e}^{\hat{\mathcal{E}}_{1,2} q_2} \underbrace{\mathbf{T}^{-1}_{l_0,l_1}(0)\mathbf{T}_{l_0,l_2}(0)}_{\mathbf{T}_{l_1,l_2}(0)} \ldots \mathbf{T}_{l_0,l_{n-1}}(0) \mathrm{e}^{\hat{\mathcal{E}}_{n-1,n} q_n} \underbrace{\mathbf{T}^{-1}_{l_0,l_{n-1}}(0)\mathbf{T}_{l_0,l_n}(0)}_{\mathbf{T}_{l_{n-1},l_n}(0)} \tag{2.33}$$

$$= \mathrm{e}^{\hat{\mathcal{E}}_{0,1} q_1} \Big(\mathbf{T}_{l_0,l_1}(0) \mathrm{e}^{\hat{\mathcal{E}}_{1,2} q_2} \mathbf{T}^{-1}_{l_0,l_1}(0)\Big) \ldots \Big(\mathbf{T}_{l_0,l_{n-1}}(0) \mathrm{e}^{\hat{\mathcal{E}}_{n-1,n} q_n} \mathbf{T}^{-1}_{l_0,l_{n-1}}(0)\Big) \mathbf{T}_{l_0,l_n}(0) \tag{2.34}$$

The adjoint transformation

$$\mathrm{Ad}_{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & \hat{\mathbf{p}}\mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{2.35}$$

is a 6×6 matrix, which transforms twists from one coordinate frame to another. By using this adjoint transformation, I have the following formula I define adjoint transformation as

$$\mathbf{T}\mathrm{e}^{\hat{\mathcal{E}} q}\mathbf{T}^{-1} = \mathrm{e}^{\widehat{\mathrm{Ad}_{\mathbf{T}}\mathcal{E}} q} \tag{2.36}$$

Therefore, I can reformulate the equation (2.32) as

$$\mathbf{T}_{st}(\mathbf{q}) = \mathrm{e}^{\hat{\mathcal{E}}_{0,1} q_1} \mathrm{e}^{\widehat{\mathrm{Ad}_{\mathbf{T}_{l_0,l_1}(0)}\mathcal{E}_{1,2}} q_2} \ldots \mathrm{e}^{\widehat{\mathrm{Ad}_{\mathbf{T}_{l_0,l_{n-1}}(0)}\mathcal{E}_{n-1,n}} q_n} \mathbf{T}_{l_0,l_n}(0) \tag{2.37}$$

By comparison between equation (2.26) and (2.37), I can immediately obtain

$$\mathcal{E}_i = \mathrm{Ad}_{\mathbf{T}_{l_0,l_i}(0)} \mathcal{E}_{i-1,i} \tag{2.38}$$

This formula verifies that the twist $\mathcal{E}_i$ is the joint twist for the i-th joint in its reference configuration and written relative to the base coordinate frame. Furthermore, if a manipulator is parameterized with D-H parameter, I can use the equation (2.31), (2.38) to compute the twist $\mathcal{E}_i$.

2.2.2 Robotic kinematic constraints

The kinematic constraints are imposed in the trajectory generation or grasp planning in terms of joint position $\mathbf{q}$, velocity $\dot{\mathbf{q}}$, acceleration $\ddot{\mathbf{q}}$, and jerk $\dddot{\mathbf{q}}$, which can be mathematically modeled as inequality constraints:

$$\mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max} \tag{2.39a}$$
$$\dot{\mathbf{q}}_{\min} \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}_{\max} \tag{2.39b}$$
$$\ddot{\mathbf{q}}_{\min} \leq \ddot{\mathbf{q}} \leq \ddot{\mathbf{q}}_{\max} \tag{2.39c}$$
$$\dddot{\mathbf{q}}_{\min} \leq \dddot{\mathbf{q}} \leq \dddot{\mathbf{q}}_{\max} \tag{2.39d}$$

In addition to the kinematic constraints, the manipulability [115, 189, 190] describes the ability of a robot which can reach or change the position freely in the workspace. The measurement of the manipulability can be characterized with the Jacobian of the manipulator $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{m\times n}$, where $m \leq n$ indicates the dimension of an workspace, and the number of robot joint $\mathbf{q}$ is denoted as n. The manipulability conditions is imposed as:

$$\max \operatorname{rank}(\mathbf{q}) = m \tag{2.40}$$

The satisfied condition of (2.40) implies that the degree of the redundancy of this manipulator is $n-m$. In the case of unsatisfied condition of (2.40), especially in some joint value $\mathbf{q}^\star$, the rank of the Jacobian is less than dimension of the workspace, and I can conclude that the manipulator falls in a singular state $\det(\mathbf{J}(\mathbf{q}^\star))$. Therefore, the quantitative measure of manipulability is defined

$$w = \sqrt{\det(\mathbf{J}(\mathbf{q})\mathbf{J}^{\mathrm{T}}(\mathbf{q}))} \tag{2.41}$$

The physical interpretation of the manipulability measure is to specify the volume of an ellipsoid, which can be computed with the singular value decomposition [74]. The scale value w can be further used to find an optimal posture [190].

2.3 Trajectory planning

A classical problem in robot motion planning aims at determining a motion profile that allows a robot to follow a given trajectory within a certain accuracy in the shortest possible time. Moreover, in most industrial applications, besides the requirement of time-optimal, the quality of a trajectory in terms of straight-line movement, jerk-limited is also vital for extending the life cycle of a robot and avoiding collision. Two trajectory types are typically applied in the real application: interpolation and approximation. In the case of interpolation, the trajectory will exactly pass through all the given waypoints, while in the approximation, the trajectory will blend the waypoints with a prescribed tolerance. I can't simply say which method is better because each has its application area. This section will introduce some basic interpolation trajectory generation models by giving the waypoints, which are created by the motion planning algorithm.

2.3.1 Implicit and parametric form

Commonly, two methods are chosen to present a curve in geometric modeling: implicit equations and parametric functions. The implicit form describes a curve lying in the two coordinate plane (e.g xy-plane) with the

$$\mathrm{f}(x, y) = 0 \tag{2.42}$$

This equation encodes the implicit relationship between these two coordinate variables, which can be considered as the set of zeros of a function of two variables. The implicit representations [154] facilitate the computation of intersection points by evaluating whether a point lies on a specific curve. In contrast to the implicit representation, parametric form separately consider each coordinate on a curve as an explicit function of an independent parameter

$$\mathbf{C}(u) = \big(x(u),\, y(u)\big) \tag{2.43}$$

where $\mathbf{C}(u)$ is a vector-valued function of the independent variable u. It is universally recognized that the parametric form is convenient to generate points along a curve. In the sense of a trajectory generation, the variable u can be interpreted as a time variable, and the $\mathbf{C}(u)$ is considered as the path traced out by a particle in terms of time. It is difficult to maintain that one formulation is superior to the other one. Moreover, the conversion between implicit and parametric equation is possible in some special cases. In the reminder of this section, I will mainly focus on the formulation in terms of parametric form. A trajectory of the particle is represented by the parameterized curve $\mathbf{p}(t) = \big(p_x(t),\, p_y(t),\, p_z(t)\big)$.

2.3.2 Trajectory with polynomial expression

The trajectory can be straightforwardly parameterized with a polynomial function

$$p(t) = a_0 + a_1 t^1 + a_2 t^2 + \ldots + a_n t^n \tag{2.44}$$

by given the coefficient a_i. The boundary conditions are used to determine the coefficient a_i. The smoothness of a trajectory is dependent on the polynomial degree n: For example, in the case of linear trajectory, which has highest degree of 1, I can have a constant velocity trajectory, and in the case of parabolic trajectory, which has the highest degree of 2, I can create a constant acceleration trajectory.

In general, the boundary conditions on the point-to-point trajectory in a one dimensional case entail initial and final position, velocity, acceleration, jerk conditions. Mathematically, I can specify these boundary conditions:

$$k!a_k + (k+1)!a_{k+1}t_j + \ldots + \frac{n!}{(n-k)!}a_n t_j^{n-k} = p^{(k)}\big(t_j\big) \tag{2.45}$$

where $p^{(k)}(t_j)$ is the k-th time derivative of the polynomial $p(t)$ at a given instant t_j. For a $n+1$ conditions, I can rewrite the (2.45) in the way of matrix multiplication:

$$\mathbf{Ma} = \mathbf{b} \tag{2.46}$$

where $\mathbf{a} \in \mathbb{R}^{n+1}$ collects all unknown coefficients with the corresponding derivative coefficients $\mathbf{M} \in \mathbb{R}^{(n+1)\times(n+1)}$. The vector $\mathbf{b} \in \mathbb{R}^{n+1}$ collects the boundary conditions for determining the polynomial coefficients. Therefore, it is straightforward to compute the $\mathbf{a}$ by multiplying the inverse the matrix $\mathbf{M}$ with the $\mathbf{b}$.

There are typically two approaches for the case of multiple waypoints: The first one is to use a high degree polynomial function to fit all values at once, which, however, can easily result in a Runge's phenomenon. Runge's phenomenon [146] is a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of a high degree over a set of equispaced interpolation points. Spline interpolation is an alternative approach for the multiple waypoints that can avoid the problem of Runge's phenomenon. It fits low-degree polynomials to small subsets of the values: for example, fitting nine cubic polynomials between each of the pairs of ten points, instead of fitting a single degree-ten polynomial to all of them. A flexible strip is then bent across each of these weights, resulting in a pleasingly smooth curve. The weights are the polynomial coefficients, which bend the line to pass through each data point without any erratic behavior or breaks in continuity.

2.3.3 Trajectory with cubic spline

The cubic spline interpolation [106] is a special case for spline interpolation, which gives an interpolating polynomial that is smoother and has a minor error than some other interpolating polynomials. The problem of cubic spline can be formulated as follows: Given a set of n+1 data points, denoted as $(t_i, p_i), i \in [1, n+1]$, where $t_{min} = t_0 < t_1 \ldots < t_{n-1} < t_n = t_{max}$, a cubic spline function $S(t_i)$ is composed of n polynomial functions:

$$\mathrm{S}(t) = \begin{cases} \mathrm{C}_1(t),\ t_{min} \le t \le t_1 \\ \vdots \\ \mathrm{C}_i(t),\ t_{i-1} \le t \le t_i \\ \vdots \\ \mathrm{C}_n(t),\ t_n \le t \le t_{max} \end{cases} \tag{2.47}$$

where $\mathrm{C}_i(t)$ is a third degree polynomial, defined by $a_i(t-t_i)^3 + b_i(t-t_i)^2 + c_i(t-t_i) + d_i, i \in [0, n]$ with $a_i \neq 0$. The first and second derivatives of n cubic functions are vital for the interpolation process.

$$\mathrm{C}_i^{'}(t) = 3a_i(t-t_i)^2 + 2b_i(t-t_i) + c_i \tag{2.48a}$$
$$\mathrm{C}_i^{''}(t) = 6a_i(t-t_i) + 2b_i \tag{2.48b}$$

The cubic spline $S(t)$ is determined by determining each cubic polynomial coefficients. Besides, for generating a smooth trajectory, I need to impose the constraints at the connection points.

$$\mathrm{C}_i(t_{i-1}) = x_{i-1}, \text{and}, C_i(t_i) = x_i,\ i \in [1, n] \tag{2.49a}$$
$$\mathrm{C}_i^{'}(t_i) = C_{i+1}^{'}(t_i),\ i \in [1, n-1] \tag{2.49b}$$
$$\mathrm{C}_i^{''}(t_i) = C_{i+1}^{''}(t_i),\ i \in [1, n-1] \tag{2.49c}$$

The equation(2.49a) imposes the constraints that the piecewise function $\mathrm{S}(t)$ will interpolate all data points. The equation (2.49a) (2.49b) and (2.49c) imply that the spline function will be continuous on the interval $[t_0, t_n]$. Built on those constraints, I have $n+n+(n-1)+(n-1) = 4n-2$ conditions, and the equation is rewritten in terms of acceleration $M_i = \mathrm{C}_i^{''}(t_i)$:

$$\underbrace{\begin{bmatrix} 1 & 4 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 4 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \\ M_n \end{bmatrix}}_{\mathbf{M}} = \underbrace{\frac{6}{T_i^2} \begin{bmatrix} x_1 - 2x_2 + x_3 \\ x_2 - 2x_3 + x_4 \\ x_3 - 2x_4 + x_5 \\ \vdots \\ x_{n-4} - 2x_{n-3} + x_{n-2} \\ x_{n-3} - 2x_{n-2} + x_{n-1} \\ x_{n-2} - 2x_{n-1} + x_n \end{bmatrix}}_{\mathbf{B}} \tag{2.50}$$

where $T_i = t_i - t_{i-1}$. Furthermore, the relationship between the M_i and the cubic polynomial coefficients a_i are formulated as following:

$$a_i = \frac{M_{i+1} - M_i}{6T_i} \tag{2.51a}$$

$$b_i = \frac{M_i}{2} \tag{2.51b}$$

$$c_i = \frac{x_{i+1} - x_i}{T_i} - \frac{M_{i+1} + 2M_i}{6} T_i \tag{2.51c}$$

$$d_i = x_i \tag{2.51d}$$

The coefficient matrix **A** has the size of $(n-2) \times n$, which is an under-determined matrix. For finding a unique cubic spline function, two additional boundary conditions are required to complete the computation. There are three common types of boundary conditions:

1. Given the initial and final velocity v_0, v_n:

$$\mathrm{C}_1^{'}(t_0) = c_1,\ \mathrm{C}_n^{'}(t_n) = c_n \tag{2.52}$$

2. Given the initial and final acceleration a_0, a_n:

$$\mathrm{C}_1^{''}(t_0) = M_1 = 2b_1,\ \mathrm{C}_n^{''}(t_n) = M_n = 2b_n \tag{2.53}$$

3. The spline function is a periodic function by imposing the constraints of the initial and end position, velocity, acceleration:

$$\mathrm{C}_1(t_0) = \mathrm{C}_n(t_n) \tag{2.54a}$$

$$\mathrm{C}_1^{'}(t_0) = \mathrm{C}_n^{'}(t_n) \tag{2.54b}$$

$$\mathrm{C}_1^{''}(t_0) = \mathrm{C}_n^{''}(t_n) \tag{2.54c}$$

Based on the equation (2.51a)-(2.51d), the additional boundary equations can be integrated to the equation (2.50) for determining a unique cubic spline function.

The aforementioned method considers the data points t_i, x_i as a preconditions, where the way-points x_i can be either manually predefined based on some specific applications or generated by using the motion planning algorithm. And the time instants t_i is used to perform the interpolation process of the points x_i, and it can be chosen in different ways with different result. In the most common techniques, I use a unitary interval by setting

$$t_0 = 0,\ t_n = 1 \tag{2.55}$$

and I define the distribution of intermediate time instants as

$$t_{i+1} = t_i + \frac{d_i}{d}, \text{with} d = \sum_{i=0}^{n-1} d_i \tag{2.56}$$

Different choice of d_i leads to different properties of cubic spline:

- By choosing $d_i = \frac{1}{n-1}$, equally spaced point, I can have a highest speed trajectory profile
- By choosing $d_i = |x_{i+1} - x_i|^{1/2}$, I can reduce the acceleration

However, the aforementioned approaches do not consider the physical constraints such as kinematic constraints, which is vital for a robotic application. Therefore, for considering the kinematic constraints, it is convenient to turn the problem into an optimization problem aiming at minimizing the whole travel time by limiting the maximum kinematic constraints [21, 94]:

$$\min \quad T = \sum_{i}^{n} T_i = t_n - t_0 \tag{2.57a}$$

$$\text{s.t:} \quad |C_i^{'}(t)| < v_{\max} \tag{2.57b}$$

$$|C_i^{''}(t)| < a_{\max} \tag{2.57c}$$

$$|C_i^{(3)}(t)| < j_{\max} \tag{2.57d}$$

The maximum velocity value $|C_i^{'}(t)|$ can exist at the time knot t_i, t_{i+1} or at the $\tilde{t}_i \in [t_i, t_{i+1}]$, where $C_i^{''}(\tilde{t}_i) = 0$, therefore, the maximum velocity constraint can be formulated as

$$|C_i^{'}(t)| < C_{i,\max}^{'}(t) = \max\{|C_i^{'}(t_i)|, |C_i^{'}(t_{i+1})|, |C_i^{'}(\tilde{t}_i)|\} < v_{\max} \tag{2.58}$$

The acceleration constraints exists only at the time knots t_i or t_{i+1}, since it is linear function.

$$|C_i^{''}(t)| < C_{i,\max}^{''}(t) = \max\{|C_i^{''}(t_i)|, |C_i^{''}(t_{i+1})|\} < a_{\max} \tag{2.59}$$

The jerk constraint is denote as $C_{i,\max}^{(3)}(t) = \frac{C_i^{''}(t_{i+1}) - C_i^{''}(t_i)}{T_{i+1}}$ This is a nonlinear optimum problem with a linear objective function, solvable with classical techniques of operational research. It is also possible solve the problem in a iterative way by scaling the time interval for satisfy the constraints [94], which however can only find a local optimum solution, since the modification occurs only in the local segment. Let

$$\lambda_v = \max_i \lambda_{v,i}, \quad \text{with} \quad \lambda_{v,i} = \max_{t_i < t < t_{i+1}} \frac{C_{i,\max}^{'}(t)}{v_{\max}} \tag{2.60a}$$

$$\lambda_a = \max_i \lambda_{a,i}, \quad \text{with} \quad \lambda_{a,i} = \max_{t_i < t < t_{i+1}} \frac{C_{i,\max}^{''}(t)}{a_{\max}} \tag{2.60b}$$

$$\lambda_j = \max_i \lambda_{j,i}, \quad \text{with} \quad \lambda_{j,i} = \max_{t_i < t < t_{i+1}} \frac{C_{i,\max}^{(3)}(t)}{j_{\max}} \tag{2.60c}$$

There have two ways for scaling the cubic spline. In the first method, I can scale the whole time interval with the same scale factor λ

$$\tilde{T}_i = \lambda T_i, \quad \text{with,} \quad \lambda = \max\{1, \lambda_v, \sqrt{\lambda_a}, \sqrt[3]{\lambda_j}\} \tag{2.61}$$

to arrive the maximum speed or the maximum acceleration. For obtaining a minimum time, I need to optimize each segment separately with

$$\tilde{T}_i = \lambda_i T_i, \quad \text{with,} \quad \lambda_i = \max\{1, \lambda_{v,i}, \sqrt{\lambda_{a,i}}, \sqrt[3]{\lambda_{j,i}}\} \tag{2.62}$$

Due to the scaling of the time interval, it can result in a discontinuous at the joints between two contiguous tracts. It is necessary to iteratively recompute the spline coefficients with $\tilde{T}_i$ using the equation (2.50), until no variation between $\tilde{T}_i$ and T_i exits, more specifically $\lambda_i = 1, i \in [0, n]$. The initial value of time distribution T_i can be computed based on equation (2.56),

2.3.4 Trajectory with B-Spline

The cubic spline limits the highest derivative of a trajectory. A higher degree of splines is required to ensure a continuous jerk or even higher-order derivative such as snap. Instead of using piece-wise higher degree polynomials, which is described as a power-based form of a curve, it is preferable to express the spline function in the way of B-form [31, 130], also called B-spline. The power-based form of the curve is numerically not stable and prone to round-off error if the coefficients vary significantly in magnitude. Besides, the algorithms for processing trajectory lacks geometry interpolation. Therefore it is challenging to design the shape by changing the coefficients of a polynomial.

Basis-spline [31, 41, 130] curves (commonly called B-spline curves) extends the concept of the Bezier curves, which has the following form:

$$\mathrm{S}(u) = \sum_{i=0}^{m} \mathbf{P}_i \, \mathrm{f}_{i,p}(u), \quad \text{with} \quad u_{\min} \le u \le u_{\max} \tag{2.63}$$

where $\mathbf{P}_i$ is indicated as the control points which form the control polygon and $\mathrm{f}_{i,p}$ describes the basis function with the degree of p with the knot vector u, where the knot vector is defined as a vector of $[u_0, u_1, \dots, u_{n_{knot}}]$ with $u_i \le u_{i+1}$. The knot span is defined as the difference between two consecutive knots: $u_{i+1} - u_i$. The constraint imposing on the basic function $f_{i,p}$ in terms of the de Boor [14] recursion formula is formulated as

$$\mathrm{f}_{i,0}(u) = \begin{cases} 1, & u_i < u < u_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.64a}$$

$$\mathrm{f}_{i,p}(u) = \omega_j^p(u) f_{i,p-1}(u) + \big(1 - \omega_{i+1}^p(u)\big) f_{i+1,p-1}(u), \quad p > 0 \tag{2.64b}$$

where the weights ω_i is defined as

$$\omega_{i,p}(u) = \begin{cases} \frac{u-u_i}{u_{i+p}-u_i}, u_{i+1} \neq u_i \\ 0, \text{otherwise} \end{cases} \tag{2.64c}$$

Furthermore, the basic function is normalized between the first and last knot

$$\sum_{j}^{m} \mathrm{f}_{i,p}(u) = 1, \forall u \in [u_0, u_{n_{\text{knot}}}] \tag{2.65}$$

From the aforementioned equations, it can be concluded that each basic function $\mathrm{f}_{i,p}$ is nonzero only on a limited number of sub-intervals $[u_i, u_{i+p+1}]$, instead of entire domain. This is the local support feature, therefore, the changing of $\mathbf{P}_i$ affects the shape of curve only on the sub-intervals. The intuitive geometry interpretation behind the control points with a normalized basic function is to blend the curve to lie within the convex hull of its control points. It can be mathematically formulated as

$$S(u) \in \text{ConvexHull}\big(\mathbf{P}_{i-p}, \dots, \mathbf{P}_i\big), u \in [u_i, u_{i+1}] \tag{2.66}$$

Furthermore, the derivative of basic function can be mathematically formulated in following:

$$\mathrm{f}'_{i,p} = \frac{p}{u_{i+p} - u_j} \mathrm{f}_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} \mathrm{f}_{i+1,p-1}(u) \tag{2.67}$$

Furthermore, I repeat the differentiation of (2.67) to the k-th derivative of $\mathrm{f}_{i,p}(u)$

$$\mathrm{f}^{(k)}_{i,p} = \frac{p}{u_{i+p} - u_i} \mathrm{f}^{(k-1)}_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} \mathrm{f}^{(k-1)}_{i+1,p-1}(u) \tag{2.68}$$

where $\mathrm{f}_{i,p-1}^{(k-1)}$ and $\mathrm{f}_{i+1,p-1}^{(k-1)}(u)$ can be rewritten in terms of $\mathrm{f}_{i,p-k},\dots,\mathrm{f}_{i+k,p-k}$. Therefore, the generalization of (2.68) can be described as:

$$\mathrm{f}_{i,p}^{k} = \frac{p!}{(p-k)!}\sum_{j=0}^{m} a_{k,j}\,\mathrm{f}_{i+j,p-k}, \quad \text{with} \quad k \le q \tag{2.69a}$$

with

$$a_{0,0} = 1, \qquad a_{k,0} = \frac{a_{k-1,0}}{u_{i+p-k+1}-u_i} \tag{2.69b}$$

$$a_{k,j} = \frac{a_{k-1,j}-a_{k-1,j-1}}{u_{i+p+j-k+1}-u_{i+j}}, \quad j = 1\dots,k-1, \qquad a_{k,k} = \frac{-a_{k-1,k-1}}{u_{i+p+1}-u_{i+k}} \tag{2.69c}$$

where the denominators of $a_{k,j}$ is nonzero, otherwise $a_{k,j}$ is set to zero.

In sense of geometry, the knot vector determines where and how the control points affect the shape of curves. The number of knot vector is chosen based on the number of control points m plus curve order $p+1$:

$$n_{\text{knot}} = m + p + 1 \tag{2.70}$$

Typically there have three different choice to determine the knot vectors: uniform, open-uniform and non-uniform. In generally, I can describe the distribution of a knot vector in the way of (2.71):

$$(\underbrace{u_0,\dots,u_{l-1}}_{l\text{ knots}},\ \underbrace{u_l,\dots,u_{n_{\text{knot}}-l-1}}_{n_{\text{knot}}-2l\text{ internal knots}},\ \underbrace{u_{n_{\text{knot}}-l},\dots,u_{n_{\text{knot}}}}_{l\text{ knots}}) \tag{2.71}$$

The multiplicity of a vector, denoted as L, is defined as the number of coinciding internal knots, therefore if all the internal knots are distinct, it implies that $L = 1$.

- The uniform distribution of a knot vector means the knot span is identical, and $L = 1$.
- The open-uniform B-spline, also denoted as clamped B-spline, requires the l-th equal knots at each begin and end, which can be formally described as $u_i = u_0$ for $0 \le i < l$, $u_{i+1} - u_i =$ constant for $l \le i \le n_{\text{knot}} - l - 1$ and $u_i = u_{n_{\text{knot}}}$ for $n_{\text{knot}} - l - 1 < i < n_{\text{knot}}$. The distribution of a knot vector in (2.71) by setting $l = p+1$ is the case of multiplicity $L = p+1$.
- The non-uniform distribution of a knot vector has only one constraint: $u_i \le u_{i+1}$. The non-uniform rational B-spline, also called NURBS, is one example of non-uniform B-spline, which is commonly used in computer graphics for representing curves and surfaces, which shows a great flexibility and precision for processing both analytic and modeled shapes.

Recall that the B-spline is used to interpolate a trajectory by given the points q_i with desired degree p

$$S(u_k) = q_k, \quad k = 0,\dots,n \tag{2.72}$$

Therefore, to compute the control point $\mathbf{P}_i, i \in [0,m]$, I can stack the $n+1$ equations to build a linear system with

$$q_k = \left[\mathrm{f}_{0,p}(u_k), \mathrm{f}_{1,p}(u_k), \dots, \mathrm{f}_{m,p}(u_k)\right]\begin{bmatrix}\mathbf{P}_0\\ \mathbf{P}_1\\ \vdots\\ \mathbf{P}_m\end{bmatrix}, k = 0,\dots,n \tag{2.73}$$

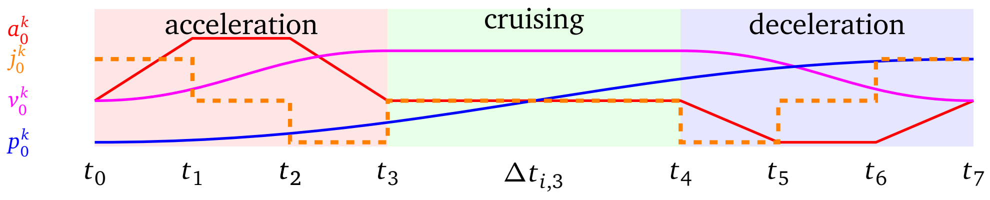


Figure 2.2: The seven-segment motion profiles at axis k between the waypoint i and $i+1$. The profile is split into three phases, where $t_{i,0}$ to $t_{i,3}$ is the acceleration phase, $\Delta t_{i,3}$ is the cruising phase, and $t_{i,4}$ to $t_{i+1,0} = t_{i,7}$ is the deceleration phase.

The relationship between the number of control points m and the waypoints n can be formulated as

$$m = \begin{cases} n+p-1, & \text{if } p \text{ is odd} \\ n+p, & \text{if } p \text{ is even} \end{cases} \tag{2.74}$$

It is straightforward to see that the $n+1$ equations can not be used to determine $m+1$ unknown control points by given the knot vectors. To generate a unique trajectory solution, p or $p+1$ additional constraints have to be imposed. Similar to cubic spline, apart from the constraints via points, the velocity $S(u_k)^{(1)}$ and acceleration $S(u_k)^{(2)}$ at the initial u_0 and final points u_n are assisted to determine the coefficients.

$$S(u_k)^{(i)} = \left[\mathrm{f}_{0,p}^{(i)}(u_k), \mathrm{f}_{1,p}^{(i)}(u_k), \ldots, \mathrm{f}_{m,p}^{(i)}(u_k) \right] \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_m \end{bmatrix} \tag{2.75}$$

It is worth noticing that in the case of the number of free parameters less than the desired conditions, additional knots can be added in any position of knot vector. By preserving the shape of curve, additional control points are need to be inserted according to the equation (2.70). In the sense of trajectory generation, I can consider the knot spans $\Delta t_i = u_{i+1} - u_i$ as traveling time between two adjacent joints, and using the same optimization formulation in (2.57a).

2.3.5 Trajectory with trapezoidal acceleration profile

The trapezoidal acceleration profile, also called the seven-segment trajectory, comprises seven different tracts with a constant jerk, described as a polynomial function with a degree of 3. The prominent distinction between the trapezoidal acceleration profile and cubic spline is that the cubic spline can be considered as one segment profile with a constant jerk. The illustration (Fig. 2.2) consists three distinguished phases: acceleration ($T_a = t_3 - t_0$), cruising ($T_v = t_4 - t_3$) and deceleration ($T_d = t_7 - t_4$) phase. With the trapezoidal acceleration profile, it is desired to generate a trajectory that makes it possible to reach the maximum value of jerk, acceleration and velocity, and result in a minimum traveling time T between two points (q_1, q_0).

Point-To-Point trajectory in one dimension

For sake of simplicity, I assume that $q_1 > q_0$, and a desired displacement $h = q_1 - q_0$ with initial and final velocity (v_0, v_1) and acceleration ($a_1 = a_0 = 0$). It is crucial to verify if a trajectory can be performed by considering the condition of displacement and various velocities. There are total four situations to verify if a trajectory can be calculated by checking the following condition

$$(v_{\text{lim}} - v_0)j_{\max} < a_{\max}^2 \tag{2.76}$$

$$(v_{\text{lim}} - v_1)j_{\max} < a_{\max}^2 \tag{2.77}$$

In the case of $v_{\text{lim}} = v_{\max}$, if the condition (2.76) holds, it implies that the acceleration can not arrive the maximum value $a_{\max}$, as a consequence, the t_1 is determined as

$$t_1 = \sqrt{\frac{(v_{\text{lim}} - v_0)}{j_{\max}}}, \quad T_a = 2t_1, \quad t_2 = t_1 \tag{2.78}$$

otherwise

$$t_1 = \frac{a_{\max}}{j_{\max}}, \quad T_a = t_1 + \frac{v_{\text{lim}} - v_0}{a_{\max}} \tag{2.79}$$

Following the same strategy, the deceleration segment $t_5 - t_4$ can be computed by replacing the velocity v_0 in the (2.78) or (2.79) with the final velocity value v_1 by checking the condition in (2.77). In the end, I can determine the time interval in the cruising phase: $T_v t_4 - t_3$ with the following formula:

$$T_v = \frac{q_1 - q_0}{v_{\text{lim}}} - \frac{T_a}{2}(1 + \frac{v_0}{v_{\text{lim}}}) - \frac{T_d}{2}(1 + \frac{v_1}{v_{\text{lim}}}) \tag{2.80}$$

The precondition $v_{\text{lim}} = v_{\max}$ will be violated if the T_v is less than zero. It implies that the maximum velocity can not be reached by given conditions. Therefore, I need to reduce the v_{lim} that is smaller than the maximum velocity, and neglect the cursing phase with $T_v = 0$ in the meantime. I firstly assume that the maximum acceleration can be arrived in this case, therefore, it is straightforward to compute the time interval with

$$t_1 = t_5 - t_4 = T_j = \frac{a_{\max}}{j_{\max}} \tag{2.81a}$$

$$T_a = \frac{C_1 - 2v_0 + \sqrt{\Delta}}{2a_{\max}} \tag{2.81b}$$

$$T_d = \frac{C_1 - 2v_1 + \sqrt{\Delta}}{2a_{\max}} \tag{2.81c}$$

where

$$C_1 = \frac{a_{\max}^2}{j_{\max}} \tag{2.81d}$$

$$\sqrt{\Delta} = C_1^2 + 2(v_1^2 + v_0^2) + a_{\max}\left(4(q_1 - q_0) - 2\frac{a_{\max}}{j_{\max}}(v_0 + v_1)\right) \tag{2.81e}$$

To verify if the assumption of arriving the maximum acceleration is satisfied, the condition of $T_a > 2T_j$ and $T_d > 2T_j$ should be evaluated. The violated state implies that the maximum acceleration can not be achieved. It requires to scaling the maximum acceleration as $a_{\text{lim}} = \lambda a_{\max}$ with $0 < \lambda < 1$, which can result in a non-optimal but acceptable solution. The choice of scaling factor λ is needed to recursive adjust until the conditions $T_a > 2T_j$ and $T_d > 2T_j$ both are satisfied. There still exist two extreme cases $T_a < 0$ or $T_d < 0$, which means no acceleration phase or no deceleration phase, respectively. The non-acceleration $T_a = 0$ and non-cruising $T_v = 0$ phase trajectory requires an necessary condition $v_0 > v_1$, and it generates a trajectory with

$$T_d = 2\frac{q_1 - q_0}{v_1 + v_0} \tag{2.82a}$$

$$t_5 - t_4 = \frac{j_{\max}(q_1 - q_0) - \sqrt{j_{\max}\left(j_{\max}(q_1 - q_0)^2 + (v_1 + v_0)^2(v_1 - v_0)\right)}}{j_{\max}(v_1 + v_0)}. \tag{2.82b}$$

Furthermore, in the same way, I can generate a trajectory with non-deceleration and non-cruising phase

$$T_a = 2\frac{q_1 - q_0}{v_1 + v_0} \tag{2.83a}$$

$$t_1 - t_0 = \frac{j_{\max}(q_1 - q_0) - \sqrt{j_{\max}\left(j_{\max}(q_1 - q_0)^2 - (v_1 + v_0)^2(v_1 - v_0)\right)}}{j_{\max}(v_1 + v_0)}. \tag{2.83b}$$

The aforementioned approach is based on the precondition of $q_1 > q_0$. To reuse the those equations, I can formulate the conditions by introducing the sign function as $\sigma = \text{sign}(q_1 - q_0)$:

$$\hat{q}_0 = \sigma q_0, \quad \hat{q}_1 = \sigma q_1, \quad \hat{v}_0 = \sigma v_0, \quad \hat{v}_1 = \sigma v_1$$

The resultant trajectory is need to be transformed by multiplying with the sign factor σ.

Multi-waypoints trajectory in multi-dimensions

The above approach is well-established for generating point-to-point trajectories, but it does not directly extend to the multiple waypoint case. This model's unique advantages motivate us to explore the broader applications of this model further. Multiple waypoint trajectories are more demanding in practical applications. For example, I must ensure that the motion of each joint dimension is synchronized at each waypoint during a task. For this reason, I propose an optimization-based solution in [100] for more complex and practical applications. Compared to previous work [56, 77, 102] that synchronized motion at each waypoint only, I impose a more stringent requirement that the motion of each dimension is synchronized at each segment. This requirement brings at least two benefits to the robot. First, by sharing the three motion phases of each dimension, the trajectory becomes smoother. In particular, by sharing the same constant velocity phase, the robot manipulator can perform tasks requiring high precision and stability in this phase, such as gluing and painting. Secondly, since the motion of each dimension has the same time period, the search space in the optimization is reduced, thus increasing the efficiency

I use $(p_0^k, p_1^k, ..., p_{n-1}^k)$ to denote waypoints in the kth dimension ($k < m$), where m is the number of degrees of freedom, n is the total number of waypoints, and p_i^k denotes the position of waypoint i in the kth dimension. For paths in configuration space, p_i^k presents the joint position. In the case of paths in Cartesian space, this refers to the Cartesian position. The trajectory between any two consecutive waypoints is modeled as a trapezoidal acceleration or a seven-segment acceleration profile.

I consider a trajectory between p_i^k and p_{i+1}^k. A typical trajectory profile is shown in Fig. 2.2, where the motion time is divided into seven segments or three phases. The acceleration phase from $t_{i,0}$ to $t_{i,3}$ has an increasing velocity. It is followed by the constant velocity phase from $t_{i,3}$ to $t_{i,4}$. The velocity is decreasing in the final deceleration phase from $t_{i,4}$ to $t_{i,7} = t_{i+1,0}$. The jerk profile has a fixed value of zero in the three time segments $(t_{i,1}, t_{i,2})$, $(t_{i,3}, t_{i,4})$, and $(t_{i,5}, t_{i,6})$ due to a constant acceleration in these segments. I denote the acceleration, velocity and position at $t_{i,h}$ as $a_{i,h}$, $v_{i,h}$, and $p_{i,h}$, respectively. The jerk in the segment $(t_{i,h-1}, t_{i,h})$ is labeled as $j_{i,h}$ with $h \in [0, \ldots, 6]$. Then, for the time $t \in (t_{i,h}, t_{i,h+1})$, the time segment can be defined as $\Delta t = t - t_{i,h}$. The acceleration, velocity, and position profiles can be derived from the previous segment $(t_{i,h-1}, t_{i,h})$ as:

$$j^k_{i,h+1}(t) = j^k_{i,h+1}\,, \tag{2.84a}$$

$$a^k_{i,h+1}(t) = a^k_{i,h} + j^k_{i,h+1}\Delta t\,, \tag{2.84b}$$

$$v^k_{i,h+1}(t) = v^k_{i,h} + a^k_{i,h}\Delta t + \frac{1}{2} j_{i,h+1}\Delta t^2\,, \tag{2.84c}$$

$$p^k_{i,h+1}(t) = p^k_{i,h} + v^k_{i,h}\Delta t + \frac{1}{2} a^k_{i,h}\Delta t^2 + \frac{1}{6} j_{i,h+1}\Delta t^3\,. \tag{2.84d}$$

As mentioned in section 2.2.2, the motion should not violate the kinematic constraints of the robot in any segment. Note that the acceleration is a monotonic function that varies with time and is piecewise smooth. Therefore, in order to guarantee the kinematic constraints within a segment, I only need to ensure that the kinematics at both ends of this segment satisfy the constraints. With the kinematic constraints considered, the motion time of the robot manipulator through all waypoints can be minimized in terms of time $t_{i,h}$ and stiffness $j_{i,h}$, which can be expressed as follows.

$$p^k(t_{i,7}) = p^k(t_{i+1,0}) = p^k_{i+1} \tag{2.85a}$$

$$v^k(t_{0,0}) = v^k(t_{n,0}) = 0 \tag{2.85b}$$

$$a^k(t_{0,0}) = a^k(t_{n,0}) = a^k(t_{i,3}) = 0 \tag{2.85c}$$

$$j^k_{i,h} = 0\,, \qquad \forall h \in [1, 3, 5] \tag{2.85d}$$

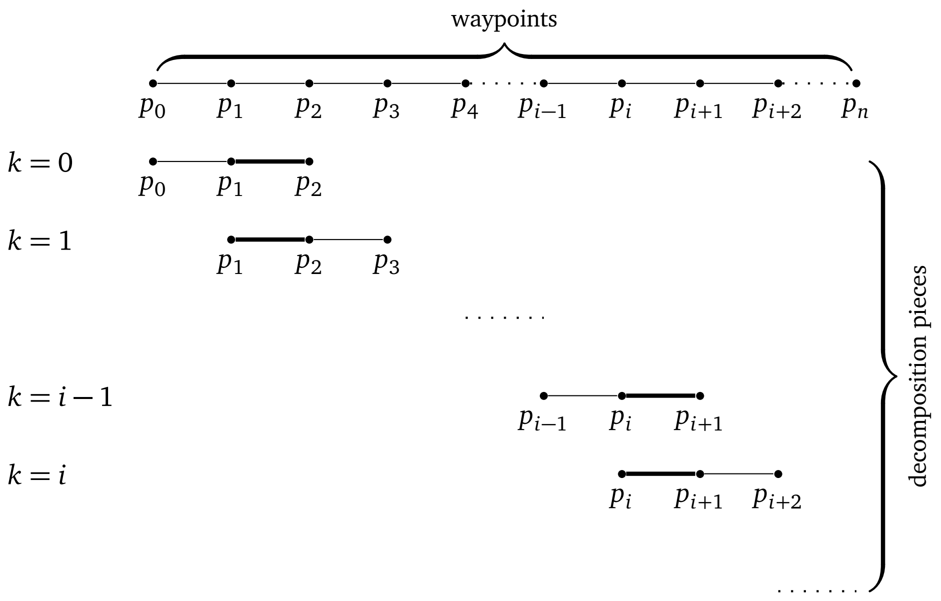
$$|a^k(t_{i,h})| \le a^k_{\max}\,, \qquad \forall h \in [0, \ldots, 6] \tag{2.85e}$$

$$|v^k(t_{i,h})| \le v^k_{\max}\,, \qquad \forall h \in [0, \ldots, 6] \tag{2.85f}$$

$$|j^k_{i,h}| \le j^k_{\max}\,, \qquad \forall h \in [0, 2, 4, 6] \tag{2.85g}$$

At the initial and final points, i.e., at $t_{0,0}$ and $t_{n,0}$, the acceleration and velocity are equal to zero. Setting $a^k(t_{i,3}) = 0$ guarantees that the velocity will remain constant during the constant velocity phase. In (2.85), since $v^k(t_{0,0})$ and $a^k(t_{0,0})$ are equal to zero and p^k_1 is known, the whole trajectory can be generated based on $t_{i,h}$ and $j^k_{i,h}$. As $j^k_{i,h}$ is equal to zero $\forall h \in [1, 3, 5]$, the unknown variables are $t_{i,h}$ $\forall i \in [0, \ldots, n-1]$ and $\forall h \in [0, \ldots, 6]$, as well as $j^k_{i,h}$ $\forall i \in [0, \ldots, n-1]$ and $\forall h \in [0, 2, 4, 6]$. Therefore, the total number of unknown variables is $7(n-1)$ for the time variables and $4(n-1)$ for the jerk variables.

Optimization Problem Formulation: The trajectory generation can be formulated as a nonlinear and nonconvex constrained optimization problem that can be solved using a sequential quadratic programming algorithm. The algorithm decomposes the nonconvex problem into a sequential convex problem. I use NLOpt's SLSQP [76] solver [64] for this purpose. With the high dimensionality of the configuration space and the large number of waypoints, the complexity of the optimization problem and optimization space increases significantly. A good initial estimate of the trajectory can significantly affect the convergence of the optimization procedure. I address this problem using the concept of model predictive control (MPC). The objective of the constrained optimization problem is to optimize the entire trajectory in time by using a trapezoidal acceleration model. The optimization parameters X for this problem include the time and jerk values for each trajectory segment. The acceleration, velocity and position can be derived automatically by (2.84) using these values. For generating a time-optimal trajectory, I formulate the objective function in terms of $\Delta t_{i,h} = t_{i,h} - t_{i,h-1}$ for



waypoints
p_0 p_1 p_2 p_3 p_4 p_{i-1} p_i p_{i+1} p_{i+2} p_n
$k=0$
$k=1$
$k=i-1$
$k=i$
decomposition pieces

Figure 2.3: The waypoints have been decomposed into $n-2$ batches. Each batch consists of three waypoints, where the former and last two waypoints (*bold lines*) are respectively overlapped with their former and subsequent batches.

every value of h and i:

$$f(X) = \sum_{i=0}^{i=n-1} \sum_{h=0}^{h=6} \Delta t_{i,h} \tag{2.86a}$$

$$\text{where} \quad X = (\{\mathbf{t}_0, \ldots, \mathbf{t}_i, \ldots, \mathbf{t}_{n-1}\}, \{\mathbf{j}_0, \ldots, \mathbf{j}_i, \ldots, \mathbf{j}_h\}), \tag{2.86b}$$

$$\mathbf{t}_i = \{\Delta t_{i,0}, \ldots, \Delta t_{i,6}\}, \tag{2.86c}$$

$$\mathbf{j}_i = \{\{j^0_{i,0}, j^0_{i,2}, j^0_{i,4}, j^0_{i,6}\}, \ldots, \{j^{m-1}_{i,0}, j^{m-1}_{i,2}, j^{m-1}_{i,4}, j^{m-1}_{i,6}\}\}. \tag{2.86d}$$

Besides the objective function, the constraints also play an important role in solving this problem. Roughly, the constraints comprise nonlinear inequality and equality constraints as well as lower and upper bounds.

$$\underset{X}{\text{minimize}} \quad f(X) \tag{2.87a}$$

$$\text{subject to} \quad \text{lb} \le X \le \text{ub} \tag{2.87b}$$

$$c(X) \le 0 \tag{2.87c}$$

$$\text{ceq}(X) = 0 \tag{2.87d}$$

The multidimensional trapezoid model consists of $7n + 4nm$ optimization variables, a total of $14mn$ nonlinear inequality constraints, and $mn + nm$ nonlinear equality constraints.

Model predictive control inspired optimization approach: The optimization problem has high computational complexity due to a large number of degrees of freedom and waypoints. It is not realistic to optimize all waypoints within one optimization step. However, the waypoints can be divided into consecutive batches. The remaining problem is then to connect all the individual batches. Directly connecting all batches will result in a zero velocity at all connecting points. In order to avoid this situation, I adopt the idea behind Model Predictive Control (MPC) approaches [167]. As shown in Fig. 2.3, I first need to set a receding horizon. I use a value of two for this, therefore only the next two waypoints p_i and p_{i+1} will be taken into account at waypoint p_{i-1} and the batch contains the waypoint set $\{p_{i-1}, p_i, p_{i+1}\}$. The

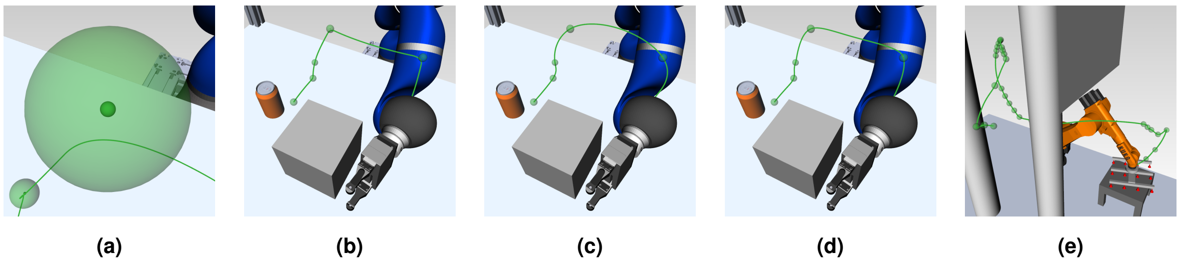


Figure 2.4: Trajectories for different interpolation models on path planning use cases for (a)–(d) a Kuka LWR next to a table with a parallel gripper and (e) a Kuka KR60-3 next to a wall and three columns with a vacuum gripper. In (a)-(b) is the result of Linear parabolic, and in (c),the cubic spline is used, and (d)-e applies the trapezoidal acceleration profile

batch including these three waypoints is used as an input for the optimization solver, but only the result between p_{i-1} and p_i will be used and the result between p_i and p_{i+1} will be discarded. However, I can still use this result as initial guess for the next optimization step, since the next batch will contain the waypoints $\{p_i, p_{i+1}, p_{i+2}\}$. Hence, every batch shares two waypoints with its former and subsequent batch. In this way, the trajectory profile between two overlapping waypoints can be predicted by optimizing the former batch and updated by optimizing the subsequent batch. As shown in Fig. 2.3, the whole set of waypoints has been successively decomposed into $n-h$ batches, where h is the receding horizon.

Experimental evaluations: I evaluate the performance of the proposed optimization approaches and two state-of-the-art approaches: Linear Parabolic interpolation [8] and Cubic Splines [31]. The Linear Parabolic model divides each interpolation step into three parts: two parabolic blends with the previous and next trajectory segments and a linear interpolation with constant velocity in the middle. I test these approaches on two path planning examples and present a detailed analysis of their performance and properties. For both the Cubic Spline and Linear Parabolic models, I ensured that the acceleration and velocity limits of each joint are satisfied by scaling the timescales of the trajectory segments. For the Trapezoid Acceleration model, this is handled intrinsically by my optimization routine.

I show how I fulfill two key requirements of path planning algorithms, i.e., reaching waypoints exactly and maintaining near-linear trajectories in the configuration space between successive waypoints. Fig. 2.4b–2.4d shows a path planning example for the Kuka LWR consisting of seven waypoints indicated by green spheres. The interpolated trajectories are shown by green lines. The trajectory calculated by the Cubic Spline model (Fig. 2.4c) deviates significantly from a straight line interpolation between successive waypoints. The trajectory from the Linear Parabolic method (Fig. 2.4b) follows a near-linear interpolation but is clearly non-smooth. My method (Fig. 2.4d) satisfies both requirements. Cubic Spline and Trapezoidal Acceleration methods are guaranteed to pass through all waypoints exactly. Linear Parabolic interpolation blends around the waypoints and fails to hit inner waypoint as demonstrated in Fig. 2.4a. The percentage of the trajectory segment that is blended influences this deviation from the waypoints. I set this to 20% for my experiments, resulting in deviations of 0.0°, 2.3°, 1.3°, 3.0°, 4.5°, 2.2°, and 0.0° for the seven waypoints.

2.4 Grasp planning

This section will show a number of the fundamental models of grasp analysis and their grasp quality criteria. In addition, several off-the-shelf simulation platforms related to grasp planning will be briefly introduced.

Similar to the robot arm, the gripper (robot hand) can be considered as a series of kinematic chain, that is composed of a palm that serves as the common base for any number of fingers, each with any number of joints. A grasp is defined as a set of contacts on the object's surface. This thesis considers only the precision grasp, where only the fingertips contact with the models and manipulator constraints are neglected. The grasp model can be mathematically described, which aims at predicting the interaction behavior between the hand and object with various conditions arising from grasping. The grasp maintenance is the most desirable behavior by handling the unknown disturbing forces and moments acting on the object/workpiece. Therefore, planning a good/optimal grasp is vital in the robotic applications that require to firmly hold an object and resist the external force [43, 116].

I define the finger joints with $\mathbf{q} = [q_1, \dots, q_{n_q}] \in \mathbb{R}^{n_q}$, where n_q is the number of joints, and $\mathbf{q}$ are denoted as the vector of joint displacement. The torques in revolute joints or forces in prismatic joints are described as $\tau = [\tau_1, \dots, \tau_{n_q}] \in \mathbb{R}^{n_q}$, which arises from interaction at the contacts between the object and hand. The wrench is defined as 6-dimensional vector by concatenation the force vector and torque vector as

$$\mathbf{w}_i = [\mathbf{f}_i^{\mathrm{T}}, \tau_i]^{\mathrm{T}} \tag{2.88}$$

where τ_i is computed as $\lambda(c_i - r) \times \mathbf{f}_i$, acting on the object at the contact c_i and its reference point $\mathbf{r}$. The center of mass is typically chosen as reference point, to give it a physical meaning.

2.4.1 Grasp closure properties

In general, a final grasp is capable of resisting the disturbance so that it can prevent the grasp from sliding the force in the contact i and its corresponding wrenches w_i, and any disturbing forces or wrenches w_{ext}. Therefore, the whole system should have lied in the equilibrium state [15]:

$$\sum_{i=0}^{i=N} w_i + w_{\text{ext}} = 0 \tag{2.89}$$

Various quality criteria can be used to judge a given grasp configuration. Form and force closure are two typical characterizations of grasp restraint, which can enable the grasp equilibrium regardless of the direction of the counter force. A grasp is considered as form-closure if the held object can not move, even infinitesimally, when the hand grasps an object with its locked finger joint and fixed palm in place. From the geometrical interpretation, form closure is a set of mechanical constraints imposed on a rigid body to disable the rigid body motion in any direction, and the object motion is completely independent of friction. Therefore, a form-closed grasp is desired when the works pieces are constrained independent of frictional properties. Under the same condition, if the applied wrenches at the contacts can balance any disturbing wrench w_{ext}, the grasp is called force-closure. Therefore I can also conclude that form closure grasp is also a kind of force closure grasps.

Form closure

The form closure can be precisely described by introducing the implicit distance function $\psi_i(u, q)$ [4]

$$\psi_i(\mathbf{u}, \mathbf{q}) \begin{cases} = 0, & \text{if the contact on the surface} \\ > 0, & \text{if contact breaks} \\ < 0, & \text{if penetration occurs} \end{cases} \quad , \forall i \in [1, \dots, n_c] \tag{2.90}$$

where $\mathbf{u}$ and $\mathbf{q}$ are the object and robot hand configuration, respectively. Mathematically speaking, a grasp $\mathbf{u}, \mathbf{q}$ satisfies the condition of form closure, if and only if the following requirements is fulfilled.

$$\psi_i(\mathbf{u} + \mathrm{d}u, \mathbf{q}) \geq 0 \implies \mathrm{d}u = 0 \tag{2.91}$$

Force closure

Force closure is a relaxation of form closure by allowing the friction force to balance the object wrench. It will reduce the number of contact points utilizing friction analysis. For example, seven contact points are required for grasping a three-dimensional object with six degrees of freedom under the form closure criteria. But only two contact points are needed to satisfy the force closure condition if the gripper is modeled as soft fingers, and three contact points are sufficient if the gripper is modeled as hard fingers. Therefore, roughly speaking, the main distinction between the form and closure is whether the friction force is used to help the grasp to hold an object. The Coulomb friction is the most commonly used friction model, which is an approximated model for computing the dry friction

$$\mathcal{F}_f = \left\{(F_i, f_{io}) \mid F_i \leq \mu_i F_{i,n}\right\} \tag{2.92a}$$
$$F_i = \|f_i - (f_i \cdot n_i) f_i\| \tag{2.92b}$$
$$F_{i,n} = f_i \cdot n_i \tag{2.92c}$$

where $\mathcal{F}_f \in \mathbb{R}^3$ is the coulomb friction cone. The force $F_{i,n}$ is the normal force exerted by each surface at the contact point c_i and contact normal n_i with the corresponding friction coefficients μ_i. In generally, the friction cones is simplified as polyhedral cones. One step closed to force closure is defined as frictional form closure with

$$\left.\begin{array}{l} \mathbf{G}\lambda = -g \\ \lambda \in \mathcal{F} \end{array}\right\} \forall g \in \mathbb{R}^{n_v} \tag{2.93a}$$

Where the $\mathbf{G}$ is the Grasp matrix, and the composite friction cone $\mathcal{F}$ defined as

$$\begin{aligned} \mathcal{F} &= \mathcal{F}_1 \times \cdots \times \mathcal{F}_{n_c} \\ &= \{f_i \in \mathbb{R}^m \mid f_i \in \mathcal{F}_i; i = 1, \ldots, n_c\} \end{aligned} \tag{2.93b}$$

The force closure is stricter than the frictional form closure with the requirements of the full rank of grasp matrix rank($\mathbf{G}$), and the intersection of nullspace of grasp matrix $\mathcal{N}(\mathbf{G})$ and nullspace of hand Jacobian $\mathcal{N}(\mathbf{J})$ is None.

2.4.2 Grasp wrench space

However, the force closure is the only minimal quality requirement for a grasp. The equation (2.89) is the primary goal for a grasp using the wrench to compensate arbitrary disturbances or balance a special set of disturbances. Combining the equation (2.88) and (2.92), the Cone Wrench space (CWS) [16] at the contact point c_i has the following formula:

$$\mathrm{CWS}_{\mathrm{c_i}} = \left\{\mathrm{w_i} \mid \mathrm{w_i} = \begin{pmatrix} \mathrm{f_i} \\ \lambda(\mathbf{c_i} - \mathbf{r}) \times \mathrm{f_i} \end{pmatrix} \wedge \|\mathrm{f_i} - (\mathbf{f_i} \cdot \mathbf{n_i})\,\mathbf{n_i}\| \leq -\mu(\mathbf{f_i} \cdot \mathbf{n_i}) \wedge \|\mathrm{f_i}\| \leq 1\right\} \tag{2.94}$$

Let's denote the Grasp Wrench Space (GWS) [16, 107, 109] as a set of all wrenches applied to the object via the k grasp contacts

$$\mathcal{W} = \left\{w \mid w = \sum_{i=1}^{k} w_i = \sum_{i=1}^{k} M_i f_i \wedge w_i \in \mathrm{CWS}_{\mathrm{c_i}}, \quad f_i \in \mathcal{F}_i, \quad i = 1, \ldots, k\right\} \tag{2.95}$$

where $\mathbf{M} = [M_1, \dots, M_k]$ is the grasp map, which maps the local contact reference frame to an object-defined global reference frame. The friction cone is commonly approximated as a pyramid with m edges, besides, any force $f_i \in \mathcal{F}_i$ can be written as positive linear combination of unitary force f_{ij} along the pyramid edges, denoted as primitive forces:

$$f_i = \sum_j^m \alpha_{ij} f_{ij}, \alpha_{ij} \geq 0 \tag{2.96}$$

The constraints of $\sum_j^m \alpha_{ij} \leq 1$ imposes the overall magnitude. The friction force can be integrated into (2.95) leading to the final formulation as

$$\mathcal{W} = \left\{ w \mid w = \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_{i,j} w_{i,j},\ w_{i,j} = \begin{bmatrix} f_{i,j} \\ \lambda\big((c_i - r) \times f_{i,j}\big) \end{bmatrix}, \sum_{j=1}^{m} \alpha_{i,j} \leq 1,\ \alpha_{i,j} \geq 0 \right\} \tag{2.97}$$

where $w_{i,j}$ is interpreted as primitive wrenches. However, the computation of GWS (2.95) can be pretty cumbersome in practice since it is challenging to find a linear combination of finger force to compensate a disturbance wrench. The alternative is using the grasp wrench hull $\tilde{\mathcal{W}}$, which can be computed very efficiently

$$\tilde{\mathcal{W}} = \left\{ w \mid w = \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_{i,j} w_{i,j}, \quad w_{i,j} = \begin{bmatrix} f_{i,j} \\ \lambda\big((c_i - r) \times f_{i,j}\big) \end{bmatrix}, \quad \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_{i,j} = 1, \alpha_{i,j} \geq 0 \right\} \tag{2.98}$$

The definition of a convex hull of a point set $\mathbf{A} = \{a_1, \dots, a_n\}$ is the smallest convex object containing $\mathbf{A}$, which can be expressed as a convex combination of $\mathbf{A}$ with

$$x = \sum_i^n \lambda_i a_i, \sum_i^n \lambda_i = 1, \lambda_i \geq 0 \tag{2.99}$$

Therefore, the constraints $\sum_{i=1}^k \sum_{j=1}^m \alpha_{i,j} = 1$ is imposed to create a convex hull of wrenches w_{ij}, resulting in a grasp wrench hull $\tilde{\mathcal{W}}$. In addition, the property $\tilde{\mathcal{W}} \subseteq \mathcal{W}$ holds by the definition. I can rewrite (2.98) using the Minkowski sum $\bigoplus$ of primitive wrenches $w_{i,j}$ for the convex hull operation, which gives an efficient way to compute the grasp wrench hull

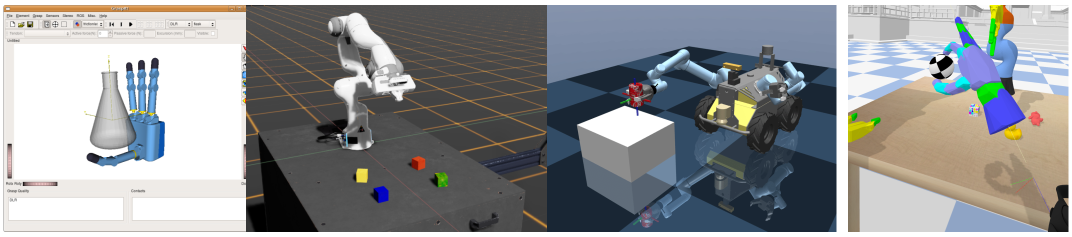
$$\tilde{\mathcal{W}}_{L_\infty} = \text{ConvexHull}\left(\bigoplus_{i=1}^{k} \{\mathbf{w}_{i,1} \dots \mathbf{w}_{i,k}\} \right) \tag{2.100}$$

where $\tilde{\mathcal{W}}_{L_\infty}$ describes the convex hull of a finite set of elements in $\mathbb{R}^6$.

2.4.3 Grasp quality

The definition and computation of form closure, force closure, and the grasp wrench space imply that grasp configuration for an object is not unique. Therefore, a grasp quality metric is necessary to rank different grasp configurations. Ferrari-Canny metric [43] is the most commonly used criteria for evaluation. It defines the grasp quality measure as the largest perturbation wrench that the grasp can resist in any direction. Mathematically speaking, the grasp quality metrics is formulated as

$$Q_{L_\infty} = \inf_{\mathbf{x} \in \partial \tilde{\mathcal{W}}_{L_\infty}} \|\mathbf{x}\|_2 \tag{2.101}$$



(a) (b) (c) (d)

Figure 2.5: A list of state of the art simulator: (a) is the Graspit! simulator, (b) is the ISSAC simulator, (c) is the mujoco simulator, and (d) is the pybullet simulator.

where $\partial \tilde{\mathcal{W}}_{L_\infty}$ describes the boundary of $\tilde{\mathcal{W}}_{L_\infty}$, and quality can be geometrically interpreted as the radius $\mathbf{x}$ of the largest ball centered at the origin of the grasp wrench space inside the grasp wrench hull. Physically, the optimal quality aims at minimizing the maximum force exerted by any of the fingers to resist an arbitrary external wrench. This grasp quality is widely used for the grasp synthesis. An alternative for obtaining the grasp quality is to sum all the force exerted by the fingers to compensate the external wrench by reformulating the Minkowski sum $\bigoplus$ in (2.100) with the union operator $\bigcup$

$$\tilde{\mathcal{W}}_{L_1} = \text{ConvexHull}\left(\bigcup_{i=1}^{k}\{\mathbf{w}_{i,1}\ldots\mathbf{w}_{i,k}\}\right) \tag{2.102a}$$

$$Q_{L_1} = \inf_{\mathbf{x}\in\partial\tilde{\mathcal{W}}_{L_1}} \|\mathbf{x}\|_2 \tag{2.102b}$$

The two qualities mentioned above depend on the reference point r, while the center of mass is typically chosen as default and coherent with the system. However, in some cases, the center of mass is difficult to be determined accurately. An alternative method is used to the convex hull volume, which is independent of the reference point.

$$Q_{\text{volume}} = \text{Volume}(\tilde{\mathcal{W}}) \tag{2.103}$$

Hence the quality is independent of the reference point, but the drawback of this quality is that it can not indicate if a grasp shows poor performance in a specific direction to resist the external wrench. There are many other research [16, 101, 162] to mitigate the problem arising from the reference-based or volume-based quality approach. The grasp quality metric is applied to guide the grasp planning process by iteratively evaluating the quality. Therefore besides the accuracy of computing the grasp quality, efficiency also plays a vital role in accelerating the computation of the quality metric. The estimation of the convex hull requires a number of resources. The de-facto standard approach uses the QuickHull algorithm [5], which is proven to be efficient and widespread used in the industry. The procedure of computing a quality is to construct the convex hull of grasp wrench space in six dimensions, and based on the (2.101), the maximum radius of the ball is calculated. It is straightforward to observe that a whole convex hull is computed in each iteration in this procedure, which is not necessary. To accelerate the computation, only the closest face of a convex hull to the center is sufficient for getting the radius. Many robot platforms have implemented the function of grasp quality metric, such as Graspit! [108], which provides several robot hands and methods for grasp selection and simulation. ISSAC simulator [93] and Mujoco [172] and bullet simulator [27] are states of the art physical engines and are attracted a lot of attention in the past years, which can be exploited to simulate the physical interaction between the object and robot hands and evaluate the grasp maintenance property.

Part II

Algorithms in Robotic Trajectory Planning

Chapter 3

Jerk-Limited Trajectories with Blending for Robot Motion Planning

Chapter Summary

This chapter extends the previous work [100] by considering the trajectory blending scenario. I presented two different approaches to generate a time local-optimal and jerk-limited trajectory with blends for a robot manipulator under consideration of kinematic constraints. The first approach generates a trajectory with blends based on the trapezoidal acceleration model by formulating the problem as a nonlinear constraint and a non-convex optimization problem. The resultant trajectory is locally optimal and approximates straight-line movement while satisfying the robot manipulator's constraints. The second approach is a combination of a trapezoidal acceleration model with a 7-degree polynomial to form a path with blends. It can be efficiently computed given the specified blending parameters. The same approach is extended to Cartesian space. Furthermore, a quaternion interpolation with a high degree polynomial under consideration of angular kinematics is introduced. Multiple practical scenarios and trajectories are tested and evaluated against other state-of-the-art approaches.

This chapter is a slightly modified version of peer-reviewed conference paper ©2021 IEEE. Reprinted, with permission, from

- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "Parameterizable and Jerk-Limited Trajectories with Blending for Robot Motion Planning and Spherical Cartesian Waypoints," IEEE International Conference on Robotics and Automation (ICRA) 2021

The images created, algorithms designed, data from experiments and text written by me in this publication will be directly referenced in this chapter. The original version is referred to 178

Contributions

I took a leading role in the writing and revising of the manuscript in this article. I have made the following significant personal contributions to the formulation, implementation, and evaluation of the algorithms in this paper: describing the trajectory planning problem as a nonlinear optimization problem, proposing and designing a model-based predicate control based optimization procedure to solve the optimization complexity problem, being the leading developer in the implementation of the algorithms, and being responsible for the evaluation of the experiments.

3.1 Introduction

Trajectory generation is a fundamental topic in the robotics community that deals with the calculation of a time-optimal, smooth, jerk-limited, and accurate motion for a well-defined task. Manually programming and optimizing paths for complex robot systems is no longer viable when it comes to flexible production with small lot sizes and multiple robot manipulators, a common use case in small and medium-sized enterprises [126]. In order to quickly adapt to new processes, new paths have to be generated automatically by modern path planning algorithms that are able to calculate complex motions for multiple manipulators in a narrow space. As cycle times should be as short as possible, globally optimal path planning algorithms [157] present a considerable advantage over classical algorithms that are followed by a local optimization step. In order to avoid stopping at every waypoint in a path, supporting various forms of blending is a desired property in a trajectory generation algorithm to further increase the performance of a robot system. Kinodynamic path planning algorithms with velocity information however are proven to be PSPACE hard [34] and therefore lead to a large increase in computation time. Industrial robot controllers and open-source implementations commonly support blending via linear parabolic motions and cubic spline interpolation. Trajectories based on linear parabolic blending, that only limit the acceleration, suffer from infinite jerk around the blend waypoints [100]. Although cubic spline interpolation can improve the smoothness of a path by limiting the jerk, it can result in a more significant deviation of the straight-line movement. This is especially important when calculating trajectories for position-based solutions in robot path planning. Trajectory generation without explicit error bounds in the path deviation can lead to undesired behavior. Deviating too far from the collision-free solution path can result in collisions. Based on these observations, I present two different approaches to generate a trajectory for following multiple waypoints. They support an explicit upper bound in deviation and are jerk-limited around the blended waypoints. In my previous work [100], I consider the situation of performing an accurate motion for a robot manipulator by forcing the trajectory to precisely pass through all waypoints, which are either manually specified or generated via a path planning algorithm. In this work, I extend this to a more general application by considering blending around the waypoints. In contrast to most state-of-the-art blending algorithms, the jerk limitation is followed throughout the trajectory. In the same way as Haschke et al. [56] and Kröger et al. [77], the trapezoidal acceleration profile is used to generate the trajectory between two consecutive waypoints. As stated in [100], the trapezoidal acceleration profile increases the optimization complexity while considering phase synchronization. In comparison to my previous work [100], I relaxed the objective function by introducing two additional weights to control the distribution of acceleration, deceleration, and cruising phases, which reduces the optimization complexity and shows a better performance from the perspective of straight-line movement. I continue to utilize the principle of model predictive control [100] for optimizing all waypoints by decomposing them into many consecutive waypoint batches and bridging each two adjacent batches with an overlapping waypoint. In addition to the optimization approach, I present another new approach that combines the trapezoidal acceleration model with a high-degree polynomial to perform a blending trajectory in joint and Cartesian space. Notably, quaternion interpolation is integrated and extended to a high degree polynomial, which considers the angular jerk and results in a smooth quaternion trajectory.

3.2 Related Work

Generating time-optimal and smooth trajectories has been studied for decades in the robotics community. The proposed trajectory planning techniques are roughly divided into two categories: online and offline planning.

Online real-time trajectory generation is mainly used to deal with unforeseen events and a dynamic and fast modification of the planned trajectory. Macfarlane et al. [102] proposed a jerk-bounded fifth-order polynomial with parabolic blends between two waypoints. Haschke et al.[56] presented an online trajectory planner by considering arbitrary initial kinematics and stopping at each waypoint. Kröger et al. [77] extended the online planner in a more general approach, which can handle arbitrary start and goal states. Lange et al. [84] proposed a path-accurate and jerk-limited online trajectory generation in configuration space. However, this cannot be easily extended to multiple waypoints and it is also not possible to blend the trajectory around the target.

Offline trajectory planning is suitable for a well-defined task, such as assembly or welding applications. The standard trajectory generator utilizes polynomials based on splines such as cubic splines or polynomials of higher degrees to provide a jerk-bound smooth trajectory. B-splines and their extension method [151] are also widely used to generate smooth trajectories. Although polynomial-based and B-spline-based algorithms can generate a smooth trajectory, they cannot fully explore the robot's capabilities and show a significant deviation from a straight line. Pham et al. [128] proposed a new approach based on reachability analysis for the time-optimal path parameterization (TOPP) problem. Similarly, Nagy and Vajk [114] applied a linear programming-based (LP) solver to tackle TOPP. Furthermore, Barnett et al. [7] introduced a bisection algorithm (BA) by extending the dynamic programming approaches to generate a trajectory. However, those algorithms are expensive to perform a trajectory with blends. Kunz et al. [81] proposed a path-following algorithm by adding circular blends that consider the acceleration bounds in joint space. Dantam et al. [30] presented spherical, parabolic blends by using the SLERP function, where no interpolation in a Cartesian pose is considered. These algorithms however do not take jerk limitation into account.

3.3 Problem Formulation

The goal is to find a time-optimal, jerk-limited, and smooth trajectory that blends an intermediate waypoint without violating kinematic constraints. Furthermore, it is required to minimize the deviation to a straight line in either joint or Cartesian space. The trapezoidal acceleration-based trajectory model [100], also called seven-segment model, has the capability to generate a smooth and jerk-limited trajectory. At segment $h \in [0, \cdots, 6]$, the kinematics are formulated as (2.84) at the waypoint i in the axis k. The parameter $\Delta t_{i,h}$ is the time difference, defined as $t_{i,h+1} - t_{i,h}$. For a phase synchronization [77] trajectory, position, velocity, acceleration, and jerk in each axis at the same segment should be synchronized. The time evolution of a position is interpreted by a third-order polynomial, which can increase the smoothness of trajectories by bounding the jerk. In this paper, I present two approaches: In the first approach, I extend my previous work, which enforces a precise pass through all waypoints, denoted as TrajOpt-Pass-Joint (TOPJ), to generate a blending trajectory by formulating it as a nonlinear constraint optimization problem, indicated as TrajOpt-Blend-Joint (TOBJ). In the second approach, I combine the trapezoidal acceleration model with a high-dimensional polynomial (7-degree) to generate a blending trajectory both in joint space TrajPoly-Blend-Joint (TPBJ) and Cartesian space TrajPoly-Blend-Cart (TPBC).

3.3.1 Blending by Optimization in Joint Space (TOBJ)

A blending trajectory is formulated as a nonlinear constraint optimization problem by applying a nonlinear optimization solver (SQP) [118].

Objective Function

The purpose of the objective function f is to find a trajectory that is optimal in time and moves as linearly as possible in joint space as

$$\begin{aligned} f = &\sum_{i=1}^{i=n}\Big(\lambda_3\Big(\big((\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\lambda_1\big)^2 + \big(\Delta t_{i,\text{cruis}}\lambda_2\big)^2\Big) \\ &+ \lambda_4 \sum_{k=1}^{k=m} (v_{i,3}^k)^2 \exp\Big(-\frac{\Delta t_{i,\text{cruis}}^2}{2\sigma^2}\Big)\Big), \end{aligned} \tag{3.1}$$

where n is the number of waypoints, m are the degrees of freedom of a robot. The time interval of the acceleration phase is indicated as $\Delta t_{i,\text{acc}}$, the cruising phase as $\Delta t_{i,\text{cruis}}$, and the deceleration phase as $\Delta t_{i,\text{dec}}$. The weight values λ_1 and λ_2 are used to control the distribution of the acceleration/deceleration and cruising phase. The choice of $\big((\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\lambda_1\big)^2 + (\Delta t_{i,\text{cruis}}\lambda_2)^2$ has advantages over the formulation $\big((\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\lambda_1 + \Delta t_{i,\text{cruis}}\lambda_2\big)^2$, which avoids the product of $(\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\Delta t_{i,\text{cruis}}$, so that acceleration/deceleration phase and cruising phase cannot affect each other. On top of this, λ_3 is used to minimize the whole trajectory time. In this formulation, λ_3 and λ_1/λ_2 conflict with each other. λ_1/λ_2 is used to achieve a straight-line motion, while minimizing the time with λ_3 requires a longer acceleration/deceleration phase that can lead to an overshooting trajectory. In [100], the straight-line deviation bound is added in the objective function. In this work, I relax this constraint by emphasizing a straight-line in joint space. Furthermore, the overshooting is observed with

$$\Delta t_{i,\text{cruis}} = \big((p_{i,\text{target}}^k - p_{i-1,0}^k) - \big(\Delta p_{i,\text{acc}}^k + \Delta p_{i,\text{dec}}^k\big)\big)/v_{i,3}^k\,, \tag{3.2}$$

where $\Delta p_{i,\text{acc}}^k = \sum_{h=0}^{h=2} p_{i,h}^k(t_{i,h}, v_{i,0}^k, a_{i,0}^k)$ and $\Delta p_{i,\text{dec}}^k = \sum_{h=4}^{h=7} p_{i,h}^k(t_{i,h}, v_{i,4}^k, a_{i,4}^k, v_{i,7}^k, a_{i,7}^k)$. If the time $\Delta t_{i,\text{cruis}}$ is negative, the reached position overshoots the target. To eliminate this undesired behavior, the cruising phase should be omitted. Since $\Delta p_{i,\text{acc}}^k + \Delta p_{i,\text{dec}}^k$ has a longer distance than $p_{i,\text{target}}^k - p_{i-1,0}^k$, I can reduce the cruising velocity $v_{i,3}^k$ to shorten $\Delta p_{i,\text{acc}}^k$. Besides, the trapezoidal end velocity $v_{i,7}$ influences the position $\Delta p_{i,\text{dec}}^k$, which can be automatically tuned by the optimization solver. I add a regular term $(v_{i,3}^k)^2 \exp(-\frac{\Delta t_{i,\text{cruis}}^2}{2\sigma^2})$ in the objective function with σ, which controls the decay rate. It can be shown that at $\Delta t_{i,\text{cruis}} \approx 0$, the regular term $(v_{i,3}^k)^2 \exp(-\frac{\Delta t_{i,\text{cruis}}^2}{2\sigma^2})$ is approximated as $(v_{i,3}^k)^2$, and the velocity is reduced by minimizing the objective function. In the case of $\Delta t_{i,\text{cruis}} > 0$, the Gaussian value $\exp(-\frac{\Delta t_{i,\text{cruis}}^2}{2\sigma^2})$ is exponentially decayed to zero, which has no effect on the cruising phase.

Kinematic Constraints

Instead of passing through the waypoints, I define a blending bound between two consecutive line segments. Furthermore, I set a non-zero velocity and acceleration for each waypoint, except for the first and last waypoint. The constraints are described as

$$p^k(t_{i,h_1}) = p^k_{i+1,\mathrm{bl,start}},\, h_1 \in [4,\ldots,6], \tag{3.3a}$$
$$p^k(t_{i+1,h_2}) = p^k_{i+1,\mathrm{bl,end}}\,, h_2 \in [0,\ldots,3], \tag{3.3b}$$
$$v^k(t_{0,0}) = v^k(t_{n,0}) = 0,\, \Delta t_{i,h} \geq 0\,, \tag{3.3c}$$
$$a^k(t_{0,0}) = a^k(t_{n,0}) = a^k(t_{i,3}) = 0\,, \tag{3.3d}$$
$$|j^k(t_{i,h})| \leq j^k_{\max}\,, |a^k(t_{i,h})| \leq a^k_{\max}\,, \forall\, h \in [0,\ldots,6] \tag{3.3e}$$
$$|v^k(t_{i,h})| \leq v^k_{\max}\,, |p^k(t_{i,h})| \leq p^k_{\max}\,, \forall\, h \in [0,\ldots,6] \tag{3.3f}$$
$$p^k_{i,\mathrm{bl,lower}} \leq p^k_{i+1,\mathrm{bl}} \leq p^k_{i+1,\mathrm{bl,upper}} \tag{3.3g}$$

where $p^k_{i+1,\mathrm{bl,start}}$ is the start blending segment position at waypoint $i+1$ in axis k and $p^k_{i+1,\mathrm{bl,end}}$ is the end blending segment position at waypoint $i+1$ in axis k. The variable $p^k_{i,\mathrm{bl,lower}}$, $p^k_{i,\mathrm{bl,upper}}$ is a lower and upper blending bound, respectively. The corresponding optimized blending waypoint at $i+1$ in axis k is indicated as $p^k_{i+1,\mathrm{bl}}$. The variable h_1 and h_2 are predefined values that depend on the blending percentage. One relaxation of the jerk $\boldsymbol{j}$ constraint [56] is made by allowing double acceleration or deceleration phases: sign($\boldsymbol{j}$) is no longer strictly defined as $[\pm,0,\mp,0,\mp,0,\pm]$ but changed to $\mathrm{sign}(\boldsymbol{j}) = [\pm,0,\pm,0,\pm,0,\pm]$. This relaxation allows reaching the next waypoint without slowing down.

Blending Bound Constraints

The blending constraint $\boldsymbol{p}_{i,\mathrm{bl,con}}$ is computed as $\boldsymbol{p}_{i+1} + \hat{\boldsymbol{y}} r \eta$, where $\hat{\boldsymbol{y}}$ is defined as $\frac{\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1}{\|\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1\|}$ with

$$\hat{\mathbf{y}}_1 = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|} \tag{3.4a}$$
$$\hat{\mathbf{y}}_2 = \frac{\mathbf{p}_{i+2} - \mathbf{p}_{i+1}}{\|\mathbf{p}_{i+2} - \mathbf{p}_{i+1}\|} \tag{3.4b}$$

Additionally, $r = l_i/(\tan\alpha_{i+1}/2)$ with $\alpha_{i+1} = \arccos(\hat{\boldsymbol{y}}_1^T \hat{\boldsymbol{y}}_2)$ and

$$l_i = \min\{\frac{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|}{2}, \frac{\|\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+1}\|}{2}, \frac{\delta \sin(\alpha_{i+1}/2)}{(1-\cos(\alpha_{i+1}/2))}\} \tag{3.5}$$

where δ is the predefined blending distance from q_{i+1}. η is the percentage value for controlling the blending bound. The deviation $\epsilon_{\mathrm{bl}} = \|\boldsymbol{p}_{i,\mathrm{bl,con}} - \boldsymbol{p}(t_{i,7})\|$ is bound. Utilizing the blending constraints, I have

$$p^k_{i,\mathrm{bl,lower}} = \min\{p^k_{i,\mathrm{bl,con}}, p^k(t_{i,7})\} \tag{3.6a}$$
$$p^k_{i,\mathrm{bl,upper}} = \max\{p^k_{i,\mathrm{bl,con}}, p^k(t_{i,7})\} \tag{3.6b}$$

Model predictive based optimization Procedure

As shown in my previous work [100], the complexity of optimizing a trapezoidal acceleration profile trajectory depends on the degrees of freedom and the number of waypoints. I still apply the Model Predictive Control (MPC)-based optimization approach and divide all waypoints into small batches by considering a horizon value and individually optimize each

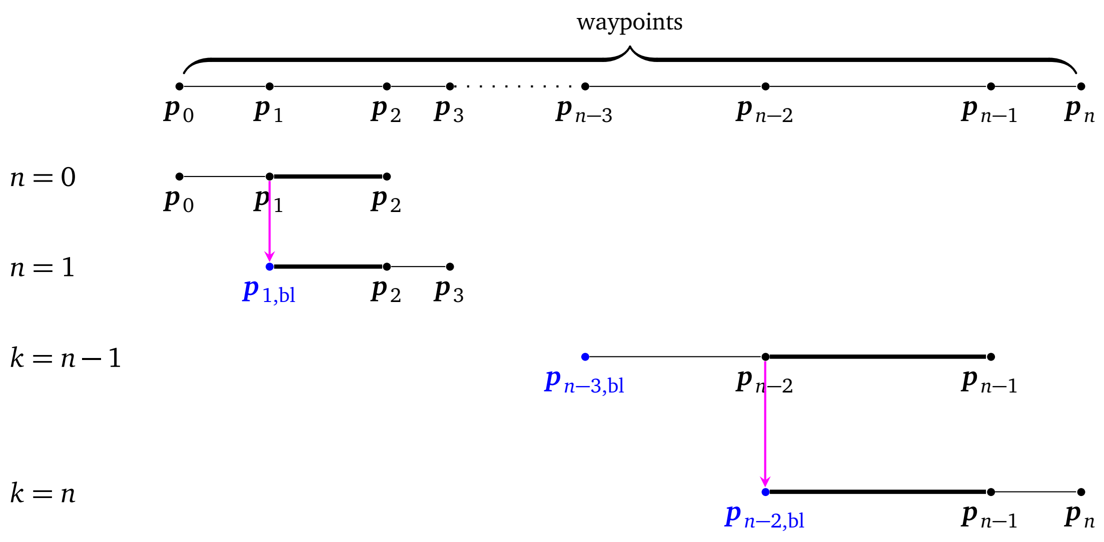


Figure 3.1: The waypoints are decomposed into $n-H$ batches, where H is the horizon step. Each batch consists of $H+1$ waypoints, where the former and last two waypoints (***bold lines***) are respectively overlapping with their former and subsequent batches. $p_{1,\mathrm{bl}}$ is the optimized blending position.

batch. As a result, the computation complexity for each batch is fixed and depends on the MPC horizon step H. In this work, I use $H=2$. Only the solution of the first two waypoints $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$ is kept and the kinematic information at waypoint $i+1$ works as the initial condition for the next batch optimization. For a high-dimensional non-convex and nonlinear constraint, a good initial guess is essential for an optimization solver that controls the gradient search direction to a good local minimum or even to a global minimum. I use the same procedure for find the initial guess as in my previous work [100]. In addition, I add a small Gaussian noise perturbation for the initial point if the solver cannot converge and repeat the optimization procedure until successful convergence to a predefined tolerance value.

3.3.2 Blending with Polynomial in Joint Space (TPBJ)

The second approach combines the trapezoidal with a high-dimensional polynomial to form a blending path. The blending segment is described as a high-degree polynomial under consideration of initial $f_0 = (p_0, v_0, a_0, j_0)$ and final $f_1 = (p_1, v_1, a_1, j_1)$ kinematic constraints. These require a total of eight coefficients, therefore I utilize a 7-degree polynomial function in one dimension:

$$f(t) = b_7 t^7 + b_6 t^6 + \cdots + b_2 t^2 + b_1 t + b_0. \tag{3.7}$$

The coefficients b_0 - b_3 can be computed using f_0 with $b_0 = p_0$, $b_1 = v_0$, $b_2 = \frac{a_0}{2}$ and $b_3 = \frac{j_0}{6}$. The coefficients b_4 to b_7 depending on f_1 and polynomial time t can be described as

$$\underbrace{\begin{pmatrix} t_1^7 & t_1^6 & t_1^5 & t_1^4 \\ 7t_1^6 & 6t_1^5 & 5t_1^4 & 4t_1^3 \\ 42t_1^5 & 30t_1^4 & 20t_1^3 & 12t_1^2 \\ 210t_1^4 & 120t_1^3 & 60t_1^2 & 24t_1 \end{pmatrix}}_{\mathbf{A}_1} \underbrace{\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \end{pmatrix}}_{\boldsymbol{x}_1} = \underbrace{\begin{pmatrix} p_1 \\ v_1 \\ a_1 \\ j_1 \end{pmatrix}}_{\boldsymbol{y}_1} - \underbrace{\begin{pmatrix} 1 & t_1^1 & t_1^2 & t_1^3 \\ 0 & 1 & 2t_1 & 3t_1^2 \\ 0 & 0 & 2 & 6t_1 \\ 0 & 0 & 0 & 6 \end{pmatrix}}_{\mathbf{A}_2} \underbrace{\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}}_{\boldsymbol{y}_2}. \tag{3.8}$$

Due to this, $\boldsymbol{x}_1(t_1, f_0, f_1) = \mathbf{A}_1^{-1}(\boldsymbol{y}_1 - \mathbf{A}_2 \boldsymbol{y}_2)$ with time matrices A_1, A_2. The polynomial is simplified as

$$f(t, f_0, f_1) = [t^7, t^6, t^5, t^4]\boldsymbol{x}_1(t_1, f_0, f_1) + h_2(t, f_0), \tag{3.9}$$

where $h_2(t, f_0)$ is described as $\frac{1}{6}j_0t^3 + \frac{1}{2}a_0t^2 + v_0t + p_0$. Firstly, I assume that the initial f_0 and final f_1 kinematic constraints are available. Therefore, $f(t)$ depends only on the time t. To find a polynomial blending trajectory that satisfies all kinematic constraints, I need to verify the extreme point of the polynomial by computing the root of its derivative. For example, the extreme point of jerk can be found at the position where the first derivative of the jerk (snap) is equal to zero. In addition, a n-degree polynomial has at most n real roots. The constraints can be mathematically formulated as

$$|f^{(n-1)}(\lambda(f^{(n)}))\chi_\lambda(\lambda(f^{(n)}))| \leq k_{\max}, \tag{3.10}$$

where $f^{(n)}$ is n-th derivative of f with $n \in [1, 4]$ and the corresponding constraints $k_{\max} \in [p_{\max}, v_{\max}, a_{\max}, j_{\max}]$. The root-finding function $\lambda(f^{(n)})$ for a given polynomial is used to find the extreme value position for $f^{(n-1)}$. $\chi_A(x)$ is the indicator function of A and will be set to one if $x \in [0, t]$, otherwise to zero. Note that if the $\chi_A(x)$ function is not derivable, the gradient-based optimization solver will diverge. To find a time-optimal polynomial trajectory that satisfies initial/final conditions and lies within the kinematic constraints, I iteratively check the constraints (3.10) by adding a small delta to $t = t + \Delta t$, where in my case Δt is set to 0.001. Furthermore, to obtain the initial f_0 and final f_1 kinematics, I compute a Point to Point (P2P) trapezoidal acceleration profile movement between each two waypoints, which can be in joint space or Cartesian space, with zero initial and end conditions, and the computed traveling time is indicated as $t_{i->i+1}$. After that, I predefine a blending percentage η to set a start blending time $t_{b_{\text{start}}} = (1-\eta)t_{i->i+1}$ and end blending time $t_{b_{\text{end}}} = \eta t_{i+1->i+2}$ for each two consecutive trajectories. By querying the trapezoidal model at $t_{b_{\text{start}}}$ and $t_{b_{\text{end}}}$, I obtain the kinematics f_0 and f_1.

3.3.3 Blending with Polynomial in Cartesian Space (TPBC)

Utilizing the same principle, I extend the algorithm to the Cartesian space, which is widely used in industrial applications. The benefits of planning trajectory in Cartesian space lies in that I can exactly generate a straight-line trajectory in Cartesian space which can not be achieved in the joint space since the forward kinematic is a nonlinear function. The blending trajectory in Cartesian space requires separate blending for position and orientation. Interpolation of the Cartesian space is executed in the same way as described in Section 3.3.2. For the orientation part, which has to consider a spherical interpolation, I apply the formulation:

$$\boldsymbol{h}(t) = \boldsymbol{h}_i \Delta \boldsymbol{h}(t) = \boldsymbol{h}_i \begin{pmatrix} \boldsymbol{u}(t)\sin(\frac{\theta(t)}{2}) \\ \cos(\frac{\theta(t)}{2}), \end{pmatrix} \tag{3.11}$$

where $\boldsymbol{h}_i$ is the initial quaternion, and $\Delta\boldsymbol{h}(t)$ transforms the quaternion from $\boldsymbol{h}_i$ to $\boldsymbol{h}(t)$. The Eigen axis between two quaternions is defined as $\boldsymbol{u}(t) = \boldsymbol{\theta}(t)/\,||\boldsymbol{\theta}(t)|| \in \mathcal{R}^3$ with a rotation angle $\theta(t) = ||\boldsymbol{\theta}(t)||$. Therefore, the quaternion interpolation depends only on $\boldsymbol{\theta}(t) \in \mathcal{R}^3$.

The time evolution function $\boldsymbol{\theta}(t)$ can be described as polynomial, such as third-order polynomial.

$$\boldsymbol{\theta}(t) = \boldsymbol{a}_1^3(x-1)^3 + \boldsymbol{a}_2^3 x(x-1)^2 + \boldsymbol{a}_3^3 x^2(x-1) + \boldsymbol{a}_4^3 x^3, \tag{3.12}$$

It can also be easily converted to the standard polynomial formulation in (3.7) by using constant matrix coefficients. To consider the angular velocity $\boldsymbol{\omega}$, angular acceleration $\dot{\boldsymbol{\omega}}$, and angular jerk $\ddot{\boldsymbol{\omega}}$ at the start and end state for the quaternion blending, the time evolution function $\boldsymbol{\theta}(t)$ is described as:

$$\boldsymbol{\theta}(t) = \boldsymbol{a}_1^7(x-1)^7 + \boldsymbol{a}_2^7 x(x-1)^6 + \cdots + \boldsymbol{a}_8^7 x^7 \tag{3.13}$$

with $x = \frac{t-t_0}{t_f-t_0} \in [0,1]$. Its roots and its derivative are computed at point $x = 0$ and $x = 1$. The derivatives of $\boldsymbol{\theta}$ can be defined as

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{a}_1^6(x-1)^6 + \boldsymbol{a}_2^6 x(x-1)^5 + \cdots + \boldsymbol{a}_7^6 x^6 \tag{3.14}$$

with the parameter $\Delta t \boldsymbol{a}_i^{n-1} = (n-i+1)\boldsymbol{a}_i^n + i\boldsymbol{a}_{i+1}^n$. The angular rate vector $\boldsymbol{\omega}$ can be computed as

$$\begin{aligned}\dot{\boldsymbol{h}} &= \frac{1}{2}\boldsymbol{h}\boldsymbol{\omega} \\ \boldsymbol{\omega} &= 2\boldsymbol{h}^{-1}\dot{\boldsymbol{h}} = 2\Delta\boldsymbol{h}\Delta\dot{\boldsymbol{h}}\end{aligned} \tag{3.15}$$

where $\Delta\dot{\boldsymbol{h}}$ can be derived as

$$\Delta\dot{\boldsymbol{h}} = \begin{pmatrix} \dot{\boldsymbol{u}}(t)\sin(\frac{\theta(t)}{2}) + \frac{1}{2}\boldsymbol{u}(t)\cos(\frac{\theta(t)}{2})\dot{\theta}(t) \\ -\frac{1}{2}\sin(\frac{\theta(t)}{2})\dot{\theta}(t) \end{pmatrix} \tag{3.16}$$

To simplify the formal, I will neglect the t in the following equations. and

$$\begin{aligned}\dot{\theta} &= \frac{d\|\boldsymbol{\theta}\|}{dt} = \frac{\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}}{\theta} = \boldsymbol{u}^T\dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{u}}(t) &= \boldsymbol{w} \times \boldsymbol{u} = (\frac{\boldsymbol{\theta}\times\dot{\boldsymbol{\theta}}}{\theta^2}) \times \boldsymbol{u} = (\frac{\boldsymbol{u}\times\dot{\boldsymbol{\theta}}}{\theta}) \times \boldsymbol{u} = -\boldsymbol{u} \times (\frac{\boldsymbol{u}\times\dot{\boldsymbol{\theta}}}{\theta})\end{aligned} \tag{3.17}$$

Formal of angular velocity $\boldsymbol{\omega}$

Replacing the $\Delta\dot{\boldsymbol{h}}$ in the equation (3.15), I can derive the relationship between $\boldsymbol{\omega} \in \mathcal{R}^3$ and $\dot{\boldsymbol{\theta}} \in \mathcal{R}^3$ as

$$\boldsymbol{\omega} = \boldsymbol{u}\dot{\theta} - \boldsymbol{w} + \sin(\theta)\boldsymbol{w} \times \boldsymbol{u} + \cos(\theta)\boldsymbol{w} \tag{3.18}$$

where $\boldsymbol{w} = \frac{(\boldsymbol{u}\times\dot{\boldsymbol{\theta}})}{\theta} \in \mathcal{R}^3$ and $\dot{\theta} = \boldsymbol{u}^T\dot{\boldsymbol{\theta}}$ is a scalar value. I further simplify $\boldsymbol{\omega}$ with the skew-symmetric matrix $(\cdot)^\times$ as

$$\begin{aligned}\boldsymbol{\omega} &= (\boldsymbol{u}\boldsymbol{u}^T - \frac{(1-\cos(\theta))}{\theta}\boldsymbol{u}^\times)\dot{\boldsymbol{\theta}} + \underbrace{\frac{\sin(\theta)}{\theta}\boldsymbol{u}\times\dot{\boldsymbol{\theta}}\times\boldsymbol{u}}_{\frac{\sin(\theta)}{\theta}(\dot{\boldsymbol{\theta}}(\boldsymbol{u}^T\boldsymbol{u})-\boldsymbol{u}(\boldsymbol{u}^T\dot{\boldsymbol{\theta}}))} \\ &= (\boldsymbol{u}\boldsymbol{u}^T + \frac{\sin(\theta)}{\theta}(\mathbf{I}-\boldsymbol{u}\boldsymbol{u}^T) - \frac{(1-\cos(\theta))}{\theta}\boldsymbol{u}^\times)\dot{\boldsymbol{\theta}} \\ &= \mathbf{A}_{\boldsymbol{\omega},1}\dot{\boldsymbol{\theta}}\end{aligned} \tag{3.19}$$

Formal of angular acceleration $\dot{\boldsymbol{\omega}}$

In the same way, I can compute the angular acceleration $\dot{\boldsymbol{\omega}}$ and angular jerk $\ddot{\boldsymbol{\omega}}$. Furthermore, the angular acceleration $\dot{\boldsymbol{\omega}}$ is formulated as

$$\begin{aligned}\dot{\boldsymbol{\omega}} &= \dot{\boldsymbol{u}}\dot{\theta} + \boldsymbol{u}\ddot{\theta} + \cos\theta\dot{\theta}\boldsymbol{w} \times \boldsymbol{u} + \sin\theta\dot{\boldsymbol{w}} \times \boldsymbol{u} + \sin\theta\boldsymbol{w} \times \dot{\boldsymbol{u}} - \sin\theta\dot{\theta}\boldsymbol{w} - (1-\cos\theta)\dot{\boldsymbol{w}} \\ &= \boldsymbol{u}\ddot{\theta} + \sin\theta\dot{\boldsymbol{w}}^\times\boldsymbol{u} - (1-\cos\theta)\dot{\boldsymbol{w}} + \boldsymbol{w}^\times\boldsymbol{u}\dot{\theta} + \underbrace{(\cos\theta\dot{\theta}\boldsymbol{w} \times \boldsymbol{u} + \sin\theta\boldsymbol{w} \times \dot{\boldsymbol{u}} - \sin\theta\dot{\theta}\boldsymbol{w})}_{H_1}\end{aligned} \tag{3.20}$$

Based on the equation (3.18), the term H_1 can be further simplified as

$$\begin{aligned}\boldsymbol{\omega} \times (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) &= (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) \times (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) + (\sin\theta\boldsymbol{w} \times \boldsymbol{u} + \cos\theta\boldsymbol{w}) \times (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) \\ &= 0 + (\sin\theta\boldsymbol{w} \times \boldsymbol{u}) \times (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) + (\cos\theta\boldsymbol{w}) \times (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) \\ &= \sin\theta\boldsymbol{w} \times \boldsymbol{u} \times \boldsymbol{u}\dot{\theta} - \sin\theta\boldsymbol{w} \times \boldsymbol{u} \times \boldsymbol{w} + \cos\theta\dot{\theta}\boldsymbol{w} \times \boldsymbol{u} \\ &= \sin\theta\boldsymbol{w} \times \boldsymbol{u} \times \boldsymbol{u}\dot{\theta} + \sin\theta\boldsymbol{w} \times \dot{\boldsymbol{u}} + \cos\theta\dot{\theta}\boldsymbol{w} \times \boldsymbol{u}\end{aligned} \tag{3.21}$$

Based on the vector triple product $(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = \mathbf{b}(\mathbf{c} \cdot \mathbf{a}) - \mathbf{a}(\mathbf{b} \cdot \mathbf{c})$. It leads to

$$\boldsymbol{w} \times \boldsymbol{u} \times \boldsymbol{u} = \mathbf{u}(\mathbf{u} \cdot \mathbf{w}) - \mathbf{w}(\mathbf{u} \cdot \mathbf{u}) = \mathbf{u}(\mathbf{u} \cdot \mathbf{w}) - \mathbf{w} = \mathbf{u}(\mathbf{u} \cdot \frac{(\boldsymbol{u} \times \dot{\boldsymbol{\theta}})}{\theta}) - \mathbf{w}. \tag{3.22}$$

Furthermore, scalar triple product of three vectors is defined as $\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}) = \mathbf{c}\cdot(\mathbf{a}\times\mathbf{b})$. Therefore the equation of $\mathbf{u} \cdot \frac{(\boldsymbol{u}\times\dot{\boldsymbol{\theta}})}{\theta} = \frac{\dot{\boldsymbol{\theta}}}{\theta} \cdot (\boldsymbol{u} \times \boldsymbol{u})$ is equal to zero. As a result, $\boldsymbol{\omega} \times (\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) = (\cos\theta\dot{\theta}\boldsymbol{w} \times \boldsymbol{u} + \sin\theta\boldsymbol{w} \times \dot{\boldsymbol{u}} - \sin\theta\dot{\theta}\boldsymbol{w}) = H_1$. The equation (3.20) can be simplified as

$$\dot{\boldsymbol{\omega}} = \boldsymbol{u}\ddot{\theta} + \sin(\theta)\dot{\boldsymbol{w}}^{\times}\boldsymbol{u} - (1-\cos\theta)\dot{\boldsymbol{w}} + \boldsymbol{w}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{\omega}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) \tag{3.23a}$$

with

$$\dot{\boldsymbol{w}} = (\boldsymbol{\theta}^{\times}\ddot{\boldsymbol{\theta}} - 2\boldsymbol{\theta}^{T}\dot{\boldsymbol{\theta}}\boldsymbol{w})/\theta^{2} \tag{3.23b}$$

$$\ddot{\theta} = (\boldsymbol{w}^{\times}\boldsymbol{u})^{T}\dot{\boldsymbol{\theta}} + \boldsymbol{u}^{T}\ddot{\boldsymbol{\theta}} \tag{3.23c}$$

By replacing the $\dot{\boldsymbol{w}}$ and $\ddot{\theta}$ in the $\dot{\boldsymbol{\omega}}$, the angular acceleration is derived as

$$\begin{aligned}\dot{\boldsymbol{\omega}} = &\underbrace{(\boldsymbol{u}\boldsymbol{u}^{T} - \frac{(1-\cos\theta)\boldsymbol{\theta}^{\times}}{\theta^{2}})\ddot{\boldsymbol{\theta}}}_{\mathbf{A}_{\dot{\boldsymbol{\omega}},1,1}} + \underbrace{\frac{\sin\theta\boldsymbol{\theta}\times\ddot{\boldsymbol{\theta}}\times\boldsymbol{u}}{\theta^{2}}}_{\frac{\sin\theta\boldsymbol{u}\times\ddot{\boldsymbol{\theta}}\times\boldsymbol{u}}{\theta} = \frac{\sin(\theta)}{\theta}(\ddot{\boldsymbol{\theta}}(\boldsymbol{u}^{T}\boldsymbol{u}) - \boldsymbol{u}(\boldsymbol{u}^{T}\ddot{\boldsymbol{\theta}})) = \mathbf{A}_{\dot{\boldsymbol{\omega}},1,2}} \\ &+ \underbrace{\boldsymbol{u}(\boldsymbol{w}^{\times}\boldsymbol{u})^{T}\dot{\boldsymbol{\theta}} + \frac{2\sin\theta\boldsymbol{u}^{\times}\boldsymbol{\theta}^{T}\dot{\boldsymbol{\theta}}\boldsymbol{w}}{\theta^{2}} + \frac{2(1-\cos\theta)\boldsymbol{\theta}^{T}\dot{\boldsymbol{\theta}}\boldsymbol{w}}{\theta^{2}} + \boldsymbol{w}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{\omega}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w})}_{\mathbf{A}_{\dot{\boldsymbol{\omega}},2}}\end{aligned} \tag{3.24}$$

By replacing $\boldsymbol{u} = \frac{\boldsymbol{\theta}}{\theta}$, the parameter $\mathbf{A}_{\dot{\boldsymbol{\omega}},1,1}\ddot{\boldsymbol{\theta}} + \mathbf{A}_{\dot{\boldsymbol{\omega}},1,2}$ can be simplified as

$$\mathbf{A}_{\dot{\boldsymbol{\omega}},1,1}\ddot{\boldsymbol{\theta}} + \mathbf{A}_{\dot{\boldsymbol{\omega}},1,2} = (\boldsymbol{u}\boldsymbol{u}^{T} + \frac{\sin(\theta)}{\theta}(\mathbf{I} - \boldsymbol{u}\boldsymbol{u}^{T}) - \frac{(1-\cos(\theta))}{\theta}\boldsymbol{u}^{\times})\ddot{\boldsymbol{\theta}} = \mathbf{A}_{\dot{\boldsymbol{\omega}},1}\ddot{\boldsymbol{\theta}} \tag{3.25}$$

and $\mathbf{A}_{\dot{\boldsymbol{\omega}},2}$ is simplified as using $\boldsymbol{u} = \frac{\boldsymbol{\theta}}{\theta}$, $\dot{\theta} = \boldsymbol{u}^{T}\dot{\boldsymbol{\theta}}$

$$\begin{aligned}\mathbf{A}_{\dot{\boldsymbol{\omega}},2} &= \boldsymbol{u}\dot{\boldsymbol{u}}^{T}\dot{\boldsymbol{\theta}} + \frac{2\sin\theta\boldsymbol{u}^{\times}\boldsymbol{u}^{T}\dot{\boldsymbol{\theta}}\boldsymbol{w}}{\theta} + \frac{2(1-\cos\theta)\boldsymbol{u}^{T}\dot{\boldsymbol{\theta}}\boldsymbol{w}}{\theta} + \boldsymbol{w}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{\omega}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) \\ &= \boldsymbol{u}\dot{\boldsymbol{u}}^{T}\dot{\boldsymbol{\theta}} + \frac{2\sin\theta\dot{\theta}\boldsymbol{u}^{\times}\boldsymbol{w}}{\theta} + \frac{2(1-\cos\theta)\dot{\theta}\boldsymbol{w}}{\theta} + \boldsymbol{w}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{\omega}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) \\ &= \dot{\boldsymbol{u}}\dot{\theta} + \frac{2\sin\theta\dot{\theta}\boldsymbol{u}^{\times}\boldsymbol{w}}{\theta} + \frac{2(1-\cos\theta)\dot{\theta}\boldsymbol{w}}{\theta} + \boldsymbol{w}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{\omega}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w})\end{aligned} \tag{3.26}$$

The angular acceleration $\dot{\boldsymbol{\omega}}$ can be reformulated as

$$\dot{\boldsymbol{\omega}} = \mathbf{A}_{\dot{\boldsymbol{\omega}},1}\ddot{\boldsymbol{\theta}} + \mathbf{A}_{\dot{\boldsymbol{\omega}},2} \tag{3.27}$$

Formal of angular Jerk $\ddot{\boldsymbol{\omega}}$

The angular jerk is defined as derivative of $\dot{\boldsymbol{\omega}}$

$$\begin{aligned}\ddot{\boldsymbol{\omega}} &= \dot{\boldsymbol{u}}\ddot{\theta} + \boldsymbol{u}\dddot{\theta} + \cos\theta\dot{\theta}\dot{\boldsymbol{w}}^{\times}\boldsymbol{u} + \sin(\theta)\ddot{\boldsymbol{w}}^{\times}\boldsymbol{u} + \sin(\theta)\dot{\boldsymbol{w}}^{\times}\dot{\boldsymbol{u}} - \sin\theta\dot{\theta}\dot{\boldsymbol{w}} - (1-\cos\theta)\ddot{\boldsymbol{w}} \\ &\quad + \dot{\boldsymbol{w}}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{w}^{\times}\dot{\boldsymbol{u}}\dot{\theta} + \boldsymbol{w}^{\times}\boldsymbol{u}\ddot{\theta} + \dot{\boldsymbol{\omega}}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) + \boldsymbol{\omega}^{\times}(\dot{\boldsymbol{u}}\dot{\theta} + \boldsymbol{u}\ddot{\theta} - \dot{\boldsymbol{w}}) \\ &= \dot{\boldsymbol{u}}\ddot{\theta} + \boldsymbol{u}\dddot{\theta} + \sin\theta\ddot{\boldsymbol{w}}^{\times}\boldsymbol{u} - (1-\cos\theta)\ddot{\boldsymbol{w}} + \boldsymbol{A}_{\ddot{\boldsymbol{w}},a} + \boldsymbol{A}_{\ddot{\boldsymbol{w}},b}\end{aligned} \tag{3.28a}$$

with

$$\dddot{\theta} = (\dot{\boldsymbol{w}}^{\times}\boldsymbol{u} + \boldsymbol{w}^{\times}\dot{\boldsymbol{u}})^T\dot{\boldsymbol{\theta}} + (\boldsymbol{w}^{\times}\boldsymbol{u})^T\ddot{\boldsymbol{\theta}} + \dot{\boldsymbol{u}}^T\ddot{\boldsymbol{\theta}} + \boldsymbol{u}^T\dddot{\boldsymbol{\theta}} = \dddot{\theta}_1 + \boldsymbol{u}^T\dddot{\boldsymbol{\theta}} \tag{3.28b}$$

$$\boldsymbol{A}_{\ddot{\boldsymbol{w}},a} = \dot{\boldsymbol{w}}^{\times}\boldsymbol{u}\dot{\theta} + \boldsymbol{w}^{\times}\dot{\boldsymbol{u}}\dot{\theta} + \boldsymbol{w}^{\times}\boldsymbol{u}\ddot{\theta} + \dot{\boldsymbol{\omega}}^{\times}(\boldsymbol{u}\dot{\theta} - \boldsymbol{w}) + \boldsymbol{\omega}^{\times}(\dot{\boldsymbol{u}}\dot{\theta} + \boldsymbol{u}\ddot{\theta} - \dot{\boldsymbol{w}}) \tag{3.28c}$$

$$\boldsymbol{A}_{\ddot{\boldsymbol{w}},b} = \dot{\theta}\cos\theta\,\dot{\boldsymbol{w}}^{\times}\boldsymbol{u} + \sin\theta\,\dot{\boldsymbol{w}}^{\times}\dot{\boldsymbol{u}} - \sin\theta\,\dot{\theta}\,\dot{\boldsymbol{w}} \tag{3.28d}$$

$$\begin{aligned}
\ddot{\boldsymbol{w}} &= \frac{\theta^2(\dot{\boldsymbol{\theta}}^{\times}\ddot{\boldsymbol{\theta}} + \boldsymbol{\theta}^{\times}\dddot{\boldsymbol{\theta}} - 2\dot{\boldsymbol{\theta}}^T\dot{\boldsymbol{\theta}}\boldsymbol{w} - 2\boldsymbol{\theta}^T\ddot{\boldsymbol{\theta}}\boldsymbol{w} - 2\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}\dot{\boldsymbol{w}}) - 2(\boldsymbol{\theta}^{\times}\ddot{\boldsymbol{\theta}} - 2\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}\boldsymbol{w})\theta\dot{\theta}}{\theta^4} \\
&= \frac{(\dot{\boldsymbol{\theta}}^{\times}\ddot{\boldsymbol{\theta}} + \boldsymbol{\theta}^{\times}\dddot{\boldsymbol{\theta}} - 2\dot{\boldsymbol{\theta}}^T\dot{\boldsymbol{\theta}}\boldsymbol{w} - 2\boldsymbol{\theta}^T\ddot{\boldsymbol{\theta}}\boldsymbol{w} - 2\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}\dot{\boldsymbol{w}})}{\theta^2} - \frac{2(\boldsymbol{\theta}^{\times}\ddot{\boldsymbol{\theta}} - 2\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}\boldsymbol{w})\dot{\theta}}{\theta^3} \\
&= \frac{(\dot{\boldsymbol{\theta}}^{\times}\ddot{\boldsymbol{\theta}} - 2\dot{\boldsymbol{\theta}}^T\dot{\boldsymbol{\theta}}\boldsymbol{w} - 2\boldsymbol{\theta}^T\ddot{\boldsymbol{\theta}}\boldsymbol{w} - 2\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}\dot{\boldsymbol{w}})}{\theta^2} - \frac{2(\boldsymbol{\theta}^{\times}\ddot{\boldsymbol{\theta}} - 2\boldsymbol{\theta}^T\dot{\boldsymbol{\theta}}\boldsymbol{w})\dot{\theta}}{\theta^3} + \frac{\boldsymbol{\theta}^{\times}\dddot{\boldsymbol{\theta}}}{\theta^2} \\
&= \boldsymbol{A}_{\ddot{\boldsymbol{w}},c} + \frac{\boldsymbol{\theta}^{\times}\dddot{\boldsymbol{\theta}}}{\theta^2}
\end{aligned} \tag{3.28e}$$

Similarly, I can formulate the angular jerk $\ddot{\boldsymbol{\omega}}$ as

$$\ddot{\boldsymbol{\omega}} = \mathbf{A}_{\ddot{\boldsymbol{\omega}},1}\dddot{\boldsymbol{\theta}} + \mathbf{A}_{\ddot{\boldsymbol{\omega}},2} \tag{3.29a}$$

where

$$\mathbf{A}_{\ddot{\boldsymbol{\omega}},1} = \mathbf{A}_{\boldsymbol{\omega},1} \tag{3.29b}$$

$$\mathbf{A}_{\ddot{\boldsymbol{\omega}},2} = \dot{\boldsymbol{u}}\ddot{\theta} + \boldsymbol{u}\dddot{\theta}_1 - (\sin\theta\,\boldsymbol{u}^{\times} + (1-\cos\theta))\boldsymbol{A}_{\ddot{\boldsymbol{w}},c} + \boldsymbol{A}_{\ddot{\boldsymbol{w}},a} + \boldsymbol{A}_{\ddot{\boldsymbol{w}},b} \tag{3.29c}$$

In summary, the quaternion trajectory blending is modeled as

$$\begin{aligned}
\boldsymbol{\theta}(t_0) &= \mathbf{0}, & \boldsymbol{\theta}(t_f) &= \boldsymbol{u}\theta \\
\dot{\boldsymbol{\theta}}(t_0) &= \mathbf{A}_{\boldsymbol{\omega},1}^{-1}\boldsymbol{\omega}_0, & \dot{\boldsymbol{\theta}}(t_f) &= \mathbf{A}_{\boldsymbol{\omega},1}^{-1}\boldsymbol{\omega}_1 \\
\ddot{\boldsymbol{\theta}}(t_0) &= \mathbf{A}_{\dot{\boldsymbol{\omega}},1}^{-1}(\dot{\boldsymbol{\omega}}_0 - \mathbf{A}_{\dot{\boldsymbol{\omega}},2}), & \ddot{\boldsymbol{\theta}}(t_f) &= \mathbf{A}_{\dot{\boldsymbol{\omega}},1}^{-1}(\dot{\boldsymbol{\omega}}_1 - \mathbf{A}_{\dot{\boldsymbol{\omega}},2}) \\
\dddot{\boldsymbol{\theta}}(t_0) &= \mathbf{A}_{\ddot{\boldsymbol{\omega}},1}^{-1}(\ddot{\boldsymbol{\omega}}_0 - \mathbf{A}_{\ddot{\boldsymbol{\omega}},2}), & \dddot{\boldsymbol{\theta}}(t_f) &= \mathbf{A}_{\ddot{\boldsymbol{\omega}},1}^{-1}(\ddot{\boldsymbol{\omega}}_1 - \mathbf{A}_{\ddot{\boldsymbol{\omega}},2}),
\end{aligned}$$

where the initial and end conditions are obtained by querying the quaternion trapezoidal acceleration profile. This process is similar to Section 3.3.2. I iteratively check the quaternion kinematics constraints in the same way as (3.10). I combine Cartesian position and quaternion interpolation to form a trajectory with blends.

3.4 Experimental Evaluation

I compare the performance of the presented approaches in this paper with the work by Kunz et al. [81] (TO-BAV), Pham et al. [128] (TOPP-RA) and my previous work [100]. The work in [81] generates a trajectory with a designed circular blend under consideration of velocity and acceleration bounds and the work [128] used the reachability-analysis (RA) to solve the time optimal path parameterization (TOPP) problem. My previous work [100] generates a trajectory by forcing a precise pass through all desired waypoints without stopping. For the evaluation of the motion planning scene, the collision-free paths in the examples are computed using the Robotics Library [142], which is also used for kinematic calculations and simulation. In the evaluation, I set the weights in [100] and TOBJ as $\lambda_1 = 1.2$, $\lambda_2 = 1$, $\lambda_3 = 5000$, and $\lambda_4 = 10$. All evaluations were performed on a laptop with a 2.6 GHz Intel Core i7-6700HQ and 16 GB of RAM.

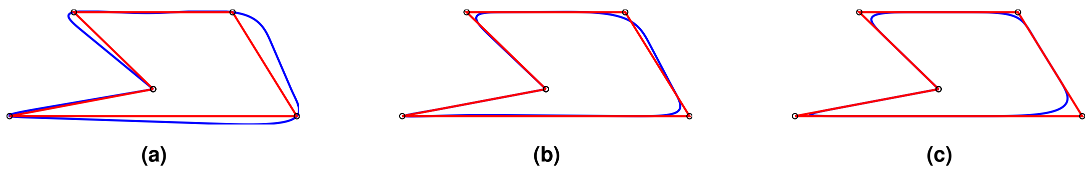


Figure 3.2: Comparison of joint space trajectories for the ISO cube scenario. The plots show the first and second DOF of different algorithms. The straight line reference in joint space is shown in *red*, the calculated joint trajectory in *blue* with a TOPJ [100], b TOBJ, c TPBJ.

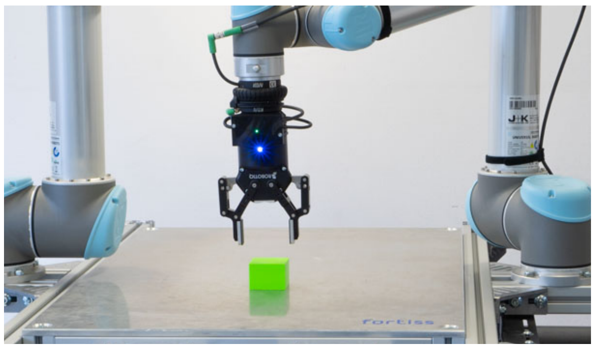


Figure 3.3: Industrial robot workcell with UR5 manipulators.

3.4.1 Evaluation of deviation from straight-line in joint space

For the first evaluation, I consider a subset of the well-known ISO 9283 [161] cube industrial benchmark as shown in Fig. 3.2. Here, the robot's end effector has to follow a number of waypoints that are part of a two-dimensional rectangle inside a three-dimensional cube while applying blending. The corresponding results are shown in Fig. 3.2. The previous work TOPJ [100] in Fig. 3.2a completed the ISO cube task within 6.26 s by passing through all waypoints and the generated trajectory has to deviate from the straight-line movement to avoid stopping at each waypoint. For improving the quality of the trajectory, TOBJ presented in this work extends the TOPJ by blending around the target position, which results in a better straight-line movement and completed the ISO cube task with a shorter time of 6.1 s due to a shortened path length. The polynomial based algorithm (TPBJ) can further reduce the completion time to 5.94 s with a bigger blending circle radius.

3.4.2 Evaluation of the algorithm in a real robot workcell

I evaluate my algorithms on a Universal Robots UR5 robot manipulator as shown in Fig. 3.3 by looking at the actual velocity and current measured by the UR5 controller with an update rate of 125 Hz over the native Real-Time Data Exchange protocol. I compare my approaches against the algorithm developed by TO-BAV [81]. The evaluation results are illustrated in Fig. 3.4. The maximum velocity and acceleration values are identical for all trajectory generators. The only difference is that [81] does not consider any jerk limitation. The remaining algorithms limit the maximum jerk to a value of five times faster than the maximum velocity. I evaluate the computational complexity of each algorithm by running the experiment 40 times. TO-BAV [81] takes 0.263(50) s to generate a blending trajectory. The proposed algorithm TrajPoly-Blend-Cart shows a similar computation time of 0.265(50) s. TPBJ is more efficient with a computation time of 0.16(2) s and is 1.56x faster than TO-BAV. In comparison to these three algorithms, the optimization-based approaches due to the non-convex and non-linear constraints optimization formulation—have a computation time of 2.52(50) s in each

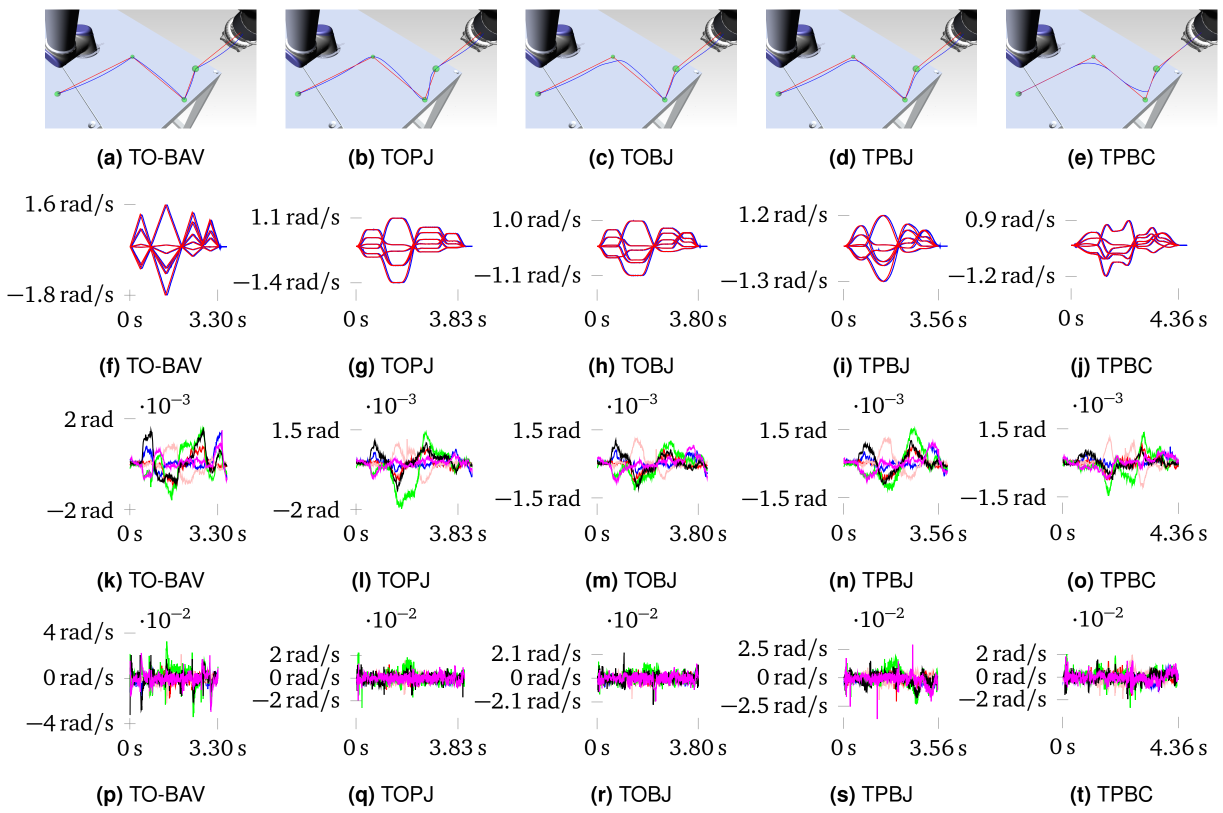
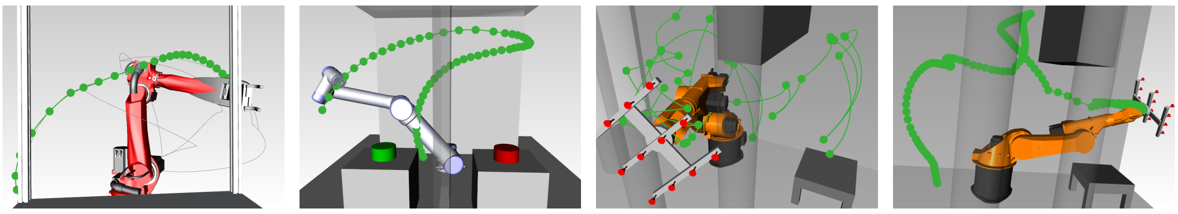


Figure 3.4: A comparison of different trajectory profiles for the UR5 example. The individual plots show a–e position, f–j velocity with the controller's target (red) and actual (blue) value, and p–t corresponding velocity differences.

batch optimization, with 86 DOF = 516 inequality constraints and $(2H+3)$ DOF = 42 equality constraints in one MPC-horizon step with $H = 2$. The computed trajectories are visualized in a 3D environment [142] as shown in figs. 3.4a to 3.4e. The blending directly implemented in Cartesian space follows a straight line. The other four algorithms have a similar trajectory performance. It needs be pointed out that the blending mode in [100] is different from the other algorithms, as it can pass precisely through all desired waypoints without stopping. From the perspective of velocity performance, the result from [81] indicates a large gap between desired and real velocity which occurs at each turning point, shown in Fig. 3.4p, as the robot controller is not able immediately to execute a trajectory that contains an infinite jerk. The robot further shows higher peaks in the measured motor current as demonstrated in Fig. 3.4k. If the maximum specified acceleration is further increased, the additional burden on the motors may lead to hardware issues and a reduced lifetime. In contrast to this, the algorithm presented in this work considers the jerk limitation. The robot controller can follow the desired waypoints continuously and demonstrates reduced motor current values. Figures 3.4g and 3.4h exhibit a clear period of cruising phase in comparison to the other algorithms, as the objective function in the optimization step emphasizes a longer constant velocity phase over the acceleration and deceleration phases. This behavior is desirable in certain industrial robot task such as welding and gluing, as the cruising segments show a smoother overall behavior compared to the other segments.

Table 3.1: Benchmark results of path planning scenarios. The best results are highlighted in bold and smaller values are better. The maximum blending deviation δ is set to 0.1. The jerk limitation used by TPBJ is set to 100x and 500x of the maximal velocity constraints, denoted as TPBJ1 and TPBJ5, respectively. TOBJ is set to 100x. N_{point} describes the number of waypoint, t_{comp} is the computation time, and t_{tra} is the traveling time. L_{alg} is the traveling length, L_{stra} is the base straight-line length, and I present the percentage.

	Scenario 1 ($N_{\text{point}} = 42$)					Scenario 2 ($N_{\text{point}} = 55$)					Scenario 3 ($N_{\text{point}} = 42$)					Scenario 4 ($N_{\text{point}} = 181$)				
Alg.	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5
t_{comp}	0.23	0.28	54.15	0.15	**0.088**	0.28	**0.186**	65.7	0.65	0.319	0.32	0.45	149.52	0.41	**0.25**	1.68	0.54	231.68	0.70	**0.41**
t_{tra}	4.08	**3.759**	7.09	7.02	5.95	3.67	**3.40**	6.35	6.58	5.88	25.20	**23.95**	26.17	26.32	25.48	11.75	**10.55**	26.48	26.44	18.91
$L_{\text{alg}}/L_{\text{stra}}$	0.999	1.0027	**0.997**	1.0001	0.999	0.999	1.003	0.998	1.0001	**0.998**	0.998	1.174	**0.997**	1.001	**0.997**	**0.996**	1.001	1.001	1.001	0.997



(a) Scenario 1 **(b)** Scenario 2 **(c)** Scenario 3 **(d)** Scenario 4

Figure 3.5: Benchmarks on different motion planning scenarios. a a Comau Racer 7-1.4 moves to a workstation with a parallel gripper. b a UR5 moving between two walls. c–d a Kuka KR60-3 next to a wall and 3 columns with a vacuum gripper.

3.4.3 Comparison between TO-BAV, TOPP-RA and TPBJ

Since the trajectory generation from [81, 128] does not consider jerk constraints, the acceleration immediately jumps to the peak value. Therefore, the velocity achieves extreme values in a short time and increases the burden on the robot motor and leads to undesired behaviors when following such a trajectory. The main benefit of my algorithm is that I can explicitly consider the jerk when generating a much smoother trajectory. This smoothness naturally comes at the cost of a longer execution time due to the additional jerk constraints. By varying the maximum value for the jerk constraints, I can tune the trajectory profile to be slightly longer with high smoothness or shorter and thus closer in runtime to [81]. In this section, I compare the algorithm of [81, 128] and my polynomial-based one, as using an extreme jerk value in the optimization-based one may not converge since the gradient value in jerk direction is not at the same order of magnitude with other gradient values in the gradient vector. The results are shown in Figs. 3.6a to 3.6o. The first column shows the results from TO-BAV [81], where the velocity arrives at the peak value very quickly with execution time 2.2577 s. The second column illustrates the results of TOPP-RA [128], which is forced precisely to pass through desired waypoints with the traveling time 2.88 s. The other columns show the results of the polynomial-based algorithm with increasing jerk limits, starting from 5, to 100, and finally 10 000 times the maximum velocity value. The respective trajectory travel time reduces from a value of 3.577 s to 2.28 s and my algorithm gradually approaches the profile of [81], while still limiting the jerk. From the perspective of computation time, TO-BAV takes 0.256 s to generate the trajectory and TOPP-RA requires slightly less time with 0.244 s to obtain the result. My algorithm is more efficient than these two algorithms with 0.165 s. I plot the first and third DOF path without loss of generality, shown in the first row. TOPP-RA produces a trajectory shown in Fig. 3.6b which has a noticeably bigger straight-line deviation in joint space than other two algorithms. The bigger straight-line deviation can lead to a collision, which is not desirable for the industrial application. From the perspective of velocity and acceleration performance, the velocity profile from TOPP-RA shown in figs. 3.6g and 3.6l is less smooth than TO-BAV and TPBJ and exhibits several vibration points.

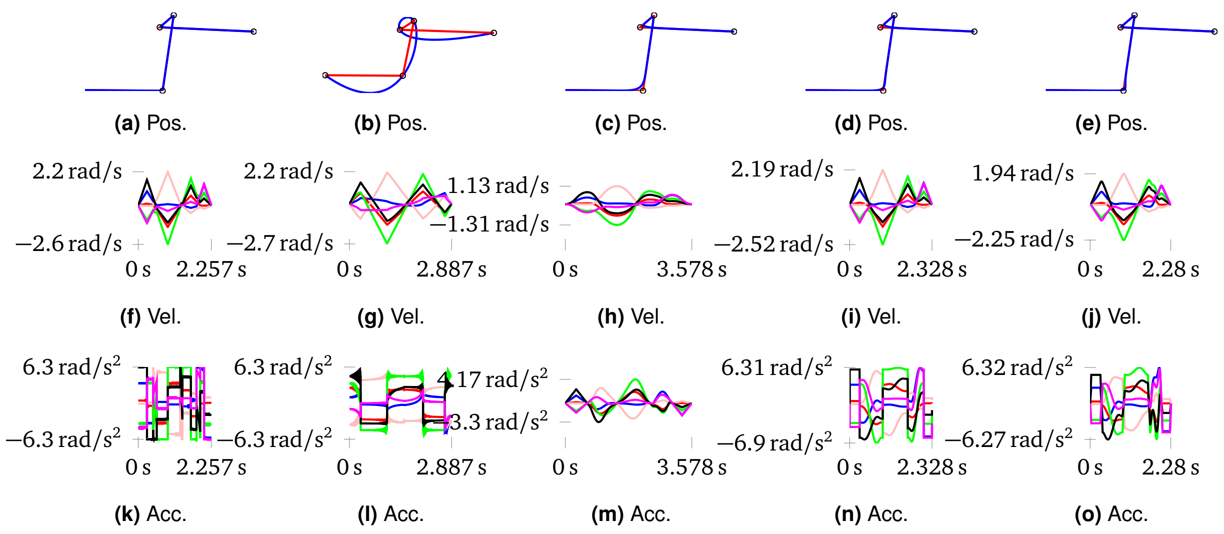
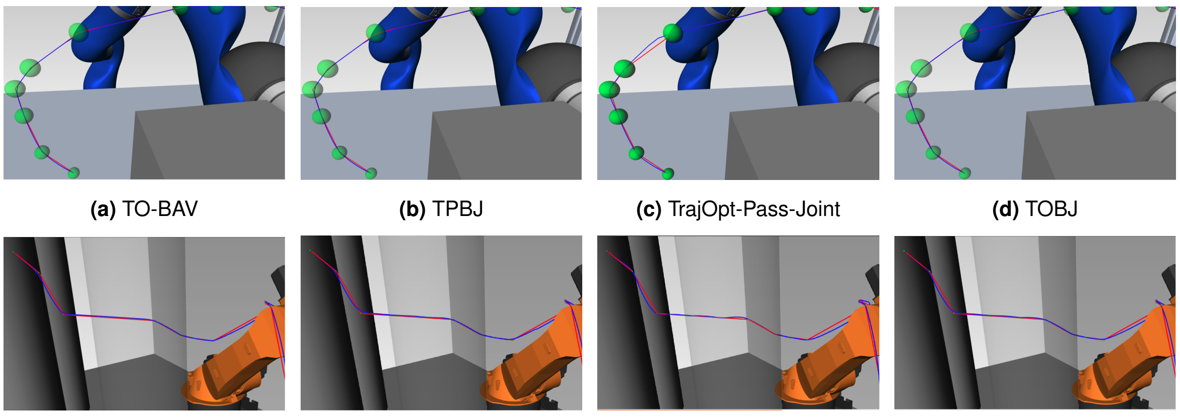


Figure 3.6: Comparison of results from TO-BAV [81] (the first column), TOPP-RA [128](the second column) and TPBJ(from the third column to the last column) with increasing jerk constraints. a–e show the plot of first and third DOF with generated trajectory (blue) and target straight-line path (red). f–j is the velocity profile. k–o is the acceleration profile. In my approach, TPBJ gradually increases jerk constraints from $j_{\max} = \{5, 100, 10000\}v_{\max}$.

3.4.4 Evaluation of computation complexity in motion planning scenarios

I evaluate the algorithms for generating trajectories in different path planning scenarios with an increasing number of waypoints (from 42 to 181) and different point distribution characteristics, as illustrated in Fig. 3.5. The paths are generated by using Robotic library [142], and the resultant path is optimized with an advanced optimizer and given as a sequence of waypoints in joint space. I compare my approaches against TO-BAV, which is currently a standard trajectory generator in the MoveIt! framework, and TOPP-RA. I summarize the results in Table 3.1. Regarding computation time, TPBJ is faster in most scenarios apart from scenario 2. In comparison to TOBJ and TPBJ under the same jerk limitation (100x of maximal velocity constraints), they show a similar performance. However, if the jerk limitation is bigger than 100x, TOBJ has a convergence problem, because jerk and time have huge differences in numerical magnitude. In comparison to TO-BAV and TPBJ with jerk limitation set to 100x and 500x of the maximum velocity constraint, I can conclude that with a higher jerk limitation, TPBJ can significantly reduce the traveling time. This conclusion can be drawn from subsection 3.4.3 as well. The trajectory optimizer TOPP-RA without jerk limitation generates a trajectory with minimal traveling time. From the perspective of straight-line deviation, TOPP-RA generated a trajectory with significant overshooting in scenario 3 with 117.4% path length with respect to a straight-line movement, since TOPP-RA utilizes cubic spline interpolation for path parameterization. The drawbacks of using cubic spline interpolation are shown in [100]. The Cartesian space blending is not evaluated in motion planning scenes since motion planning implemented in configuration space did not consider the singularity problem, which can lead to a problem by inverse kinematics. The generated solutions from a path planner are given as a sequence of waypoints in joint space. The final results are illustrated in Fig. 3.7. For following the linearly interpolated solution path, TO-BAV of [81], as well as TOBJ and TPBJ, presented here show similar performances. TrajOpt-Pass-Joint from [100] has a small deviation due to its property of precisely following through the given waypoints and adhering to a specified tolerance between them. From the perspective of time, the trajectory generated by TO-BAV arrives at the goal in the shortest time, however without considering jerk limitations. From the perspective of computation time, in the scenario



(a) TO-BAV **(b)** TPBJ **(c)** TrajOpt-Pass-Joint **(d)** TOBJ

(e) TO-BAV **(f)** TPBJ **(g)** TrajOpt-Pass-Joint **(h)** TOBJ

Figure 3.7: Evaluation of trajectories in different motion planning scenarios, a–b a Kuka LWR next to a table with a parallel gripper with 15 waypoints, e–f a Kuka KR60-3 next to a wall and three columns with a vacuum gripper with 28 waypoints

a–b,TPBJ is 3x faster than the TO-BAV, and in the scenario e–f is 2.76x faster. By comparing TrajOpt-Pass-Joint and TOBJ, I can see that under the premise of no collision, the blending options can provide a better trajectory performance from the perspective of straight-line movement.

3.5 Conclusion

In this work, I have extended my previous work by including the capability to blend around the target position. I have presented two different approaches to finding a time-optimal and jerk-limited trajectory. In the first approach, the algorithm follows the same principle as in my previous work by using a bridged optimization procedure, which reduces the computational complexity to a linear complexity with respect to the number of waypoints and degrees of freedom. In contrast to my previous work, I redesigned the objective function and blending constraints to achieve a better straight-line movement in joint space and allow the trajectory to blend around the target position. However, TOBJ has a convergence problem when the jerk constraint exceeds 100x of the maximum velocity constraint due to the huge differences in numerical magnitude between jerk und time. Further improvement will be left to future work by using more advanced optimization strategies. The second approach combines a trapezoidal trajectory with a seven-degree polynomial. In this approach, I compute a point-to-point motion for every two waypoints by using a standard trapezoidal acceleration model. By specifying a blend percentage, the seven-degree polynomial can be used to find a curve segment around the target position. To find a time-optimal trajectory that fully considers all kinematic constraints, I iteratively increase the trajectory time until these constraints are no longer violated for all degrees of freedom. The second approach can be directly extended to the Cartesian space by using the introduced quaternion interpolation algorithm. These two approaches do not suffer from convergence problems and show good performance in my experiments when compared against state-of-the-art approaches without jerk limitations.

Part III

Algorithms in 3D Computer vision

Chapter 4
Gaussian Process Implicit Surfaces based 6D Pose Estimation

Chapter Summary

In this chapter, I proposed a surface-to-surface (S2S) point registration algorithm by exploiting the Gaussian Process Implicit Surfaces for partially overlapping 3d-surfaces. Unlike traditional approaches, which separate the corresponding search and update steps in the inner loop, I formulate the point registration as a nonlinear non-constraints optimization problem which does not expect to find any corresponding points between two point sets. According to the implicit function theorem, I form one point set as a Gaussian Process Implicit Surfaces utilizing the signed distance function, which implicitly creates three manifolds. Points on the same manifold share the same function value, indicated as $\{1; 0; -1\}$. In consequence, the problem is converted to find a rigid transformation, which minimizes the inherent function value. In the case of a partially overlapping 3D surface, the Fast Point Feature Histogram (FPFH) algorithm is applied to both point sets and a Principal Component Analysis (PCA) is performed on the result. Based on this, the initial transformation can then be computed. I conduct experiments on multiple datasets to evaluate the effectiveness of my proposed approach against existing state-of-the-art methods.

- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "6D Pose Estimation for Flexible Production with Small Lot Sizes based on CAD Models using Gaussian Process Implicit Surfaces," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020

The images created, algorithms designed, data from experiments and text written by me in this publication will be directly referenced in this chapter. The original version is referred to 169

Contributions

I took a leading role in the composition and revision of the manuscript. I have made the following significant personal contributions in discussing and developing the ideas, implementation, and evaluation: introducing the GPIS to reformulate the 6D pose estimation problem and designing a Lie-manifold optimization procedure for solving the GPIS based pose estimation. I am the leading developer of the algorithm implementation and am responsible for the experimental evaluation.

4.1 Introduction

While the majority of European companies consists of small and medium-sized enterprises, only a very limited number of them currently uses robot systems in their production. In contrast to larger companies, they mainly deal with small lot sizes and constantly changing production processes. Adapting a robot systems to new products and parts is however very time-consuming and requires expert knowledge in robotics that is not commonly found in shop floor workers [126]. While a number of modern robot systems currently on the market proposes an easier programming concept based on reusable skills, these approaches still require a manual adaptation to new processes. In contrast to this, approaches from service robotics are able to automatically solve declarative goal specifications by using semantic knowledge in combination with reasoning and inference [169].

Synthesizing robot programs based on semantic product, process, and resource descriptions enables an automatic adaptation to new processes and involves handling the recognition of objects and parts in the environment. These parts are typically designed in CAD systems and described via a boundary representation [127]. In order to grasp them with a robot, a full 6D pose estimation is required. Given the small lot size production of SMEs with constantly changing objects, it is not feasible to train object recognition models over a long period of time. Recognizing these parts efficiently by directly using their CAD models during execution is therefore essential in achieving short changeover times. Fig. 4.1 shows an example of such an assembly use case for a mechanical gearbox together with a point cloud scene captured by the 3D camera sensor attached to the robot.

Point registration is one of the main approaches in computing the pose transformation by two given point sets and is widely used in MRI/CAT scan alignment [42] and robot manipulation [131]. The problem is especially challenging when two noisy point sets only partially overlap without initial alignment. A standard approach for point registration is based on the Iterative Closest Point (ICP) algorithm [50, 155]. It is interesting due to its intuitive and straightforward implementation. The identification of corresponding points follows a greedy search algorithm that is subjective to local minima and identifies incorrect points for some rotations. Furthermore, the success of ICP heavily relies on a good initial alignment. There are many variants which aim at optimizing the process for correspondence search, such as widening the convergence basin, heuristic global search, relaxed assignments, or distance fields, which however often fail to achieve a better performance and typically follow the principle of point to point (p2p) or point to surface (p2s) registration. Furthermore, with increasingly powerful neural networks, many researchers started applying deep learning to the problem of computing a pose transformation [78]. However, these approaches still follow the concept of finding the corresponding point and use a RANASAC-based Perspective-n-Point(PnP) algorithm to acquire the 6D pose. In addition, they require a large data set to encode a surrogate task and cannot be transferred to another task efficiently. For a flexible production application however, efficiency is a key factor required in any suitable algorithm.

This work, to the best of my knowledge, is the first one to consider the surface-to-surface (s2s) point registration and to describe one surface as an implicit function by employing Gaussian process regression, also referred to as Gaussian Process Implicit Surfaces (GPIS). I define three manifolds by borrowing the idea of signed distance functions (SDF) [196] with values $\{1, 0, -1\}$. All points on a surface should have the same value of 0. Instead of searching the corresponding points between two different point sets, the goal is now to find a rigid transformation that makes function value zero by transferring the point using this transformation. I evaluate the presented GPIS-based point registration on multiple point sets. In comparison to state-of-the-art algorithms, the presented approach can arrive at or exceed the same accuracy. I prioritize robustness over convergence speed and my approach can achieve

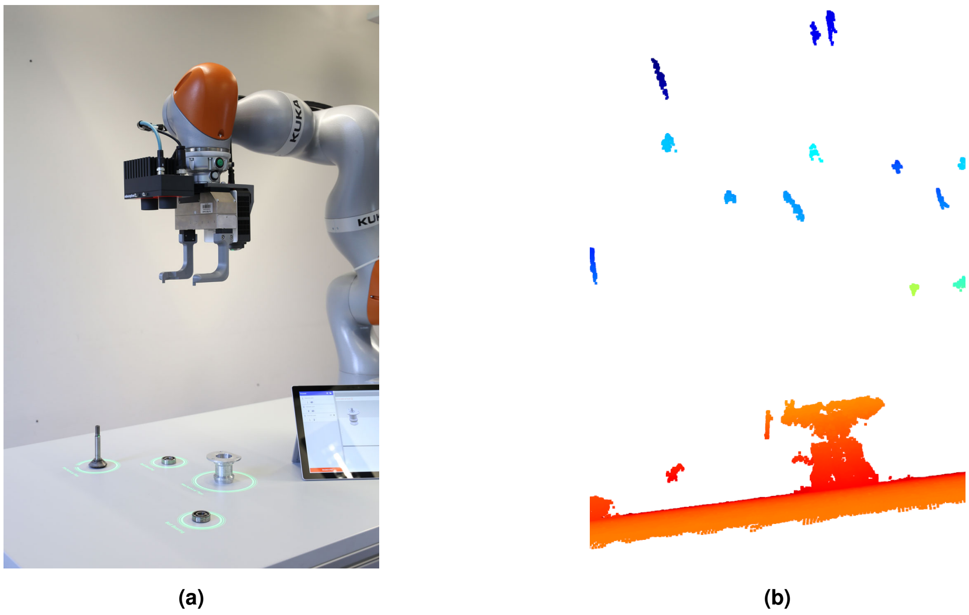


Figure 4.1: Robot setup for assembling a gear box with (a) a lightweight robot and a 3D camera sensor, (b) point cloud scene of mechanical gearbox parts on the table.

more robust results than most of the existing algorithms. The primary advantage in contrast to other algorithms is that my approach does not require finding the corresponding point iteratively. The initialization for the transformation can be calculated based on a FPFH and PCA. The registration problem is efficiently solved with a Lie algebra-based Gaussian Newton solver.

4.2 Related Work

6D pose estimation is widely used and extensively studied and point registration technology is commonly used to find the spatial relationship between two point sets. Most of the advanced algorithms in this field are based on ICP and several variants exist [132, 155]. The typical work flow for geometric registration consists of two stages: initial (global) alignment and local refinement. Initial alignment is either based on simple Euclidean distance or on more complex sampling-based algorithms that identify matching points by utilizing local geometrical descriptors like Fast Point Feature Histogram (FPFH) [53, 148]. RANSAC [138] can be applied against outliers. In the following, the initial rigid transformation can be estimated by using a least squared method or by using the branch-and-bound framework (BnB) [187], which is a global optimization algorithm. In both cases, finding a good initial alignment can be computationally expensive. After the initial alignment, local refinement is executed by alternating the steps for finding the nearest neighborhood and the steps for updating the transformation based on the greedy search algorithm. This is susceptible to local minima and can only produce an accurate result with a good initialization. Several variants can be used to improve the performance: Fitzgibbon et al. [44] proposed a Levenberg-Marquardt algorithm that uses finite differences to optimize the objective function. Granger et al. [50] applied Expectation-Maximization (EM) principles to consider Gaussian noise, which can improve the robustness of local registration. Li et al. [91] utilize the Gaussian mixture model to model the surface uncertainty so that it increases the robustness of the registration. Myronenko et al.[113] introduced Coherent Point Drift (CPD), which is agnostic as to the used transformation model and similar to GMM takes a probabilistic approach to the alignment of point sets. Chavdar et al. [123] applied stochastic optimization to consider noise robustness, outlier resistance, and optimal global alignment. Yang et al. [187] proposed Globally Optimal ICP (Go-ICP) by using the Branch and Bound framework to derive the lower and upper bound for the error function and to integrate a local ICP in the same frame. Guo et al. [53] introduced the Fast Global Registration (FGR) algorithm that uses a scaled Geman-McClure estimator to describe the error function, optimizes the objective function by using Block Coordinate Descent, and applies FPFH to search the corresponding set before optimization.

All algorithms mentioned above share the requirement of finding the correct corresponding pair. This is followed by using either greedy search or a global optimizer to get the final rigid transformation. With continuous improvement in the field of deep learning, many researchers began to learn the 6D pose directly by using RGB images [184]. However, a large number of point sets is required to learn the surrogate task and the underlying relationship between learned loss function and the pose accuracy is still unclear. Such an approach cannot be easily deployed in a flexible production line.

4.3 Problem Formulation

The standard ICP algorithm aims to estimate a rigid transformation $\mathbf{T} = \{\mathbf{R}, \mathbf{p}\}$ between given two point sets $\mathcal{X} = \{\mathbf{x}_i\}, i = 0, \ldots, n$ and $\mathcal{Y} = \{\mathbf{y}_i\}, i = 0, \ldots, m$, that minimizes the object function

$$E(\mathbf{T}) = \min \sum_{i=1}^{n} \left(\mathbf{x}_i - \mathbf{T}\mathbf{y}_{\arg\min_{j=0,\ldots,m} \left(\|\mathbf{x}_i - \mathbf{T}\mathbf{y}_j\|^2 \right)} \right) \tag{4.1}$$

by iteratively finding corresponding points. The objective function in (4.1) however is a non-convex function and therefore susceptible to local minima. The basic ICP algorithm can be formulated as:

1. Set the initial guess for the transformation $\mathbf{T}_0 = \{\mathbf{R}_0, \mathbf{p}_0\}$ and $\mathcal{P}' = \mathcal{P}$, and closest point dataset $\mathcal{X}' = \{\emptyset\}$

2. Find point correspondence for each point $\mathbf{p}'_i$ from the set $\mathcal{P}'$ to the set $\mathcal{X}$ by optimized the object function $j = \text{argmin}_{k=0\cdots N_x} d(\mathbf{p}'_i, \mathbf{x}_k)$, and add $\mathbf{x}'_j$ to $\mathcal{X}'$

3. Find alignment by optimizing 4.1 to find $\mathbf{T}_t = \{\mathbf{R}_t, \mathbf{p}_t\}$, which can be solved by using the singular value decomposition.

4. Apply alignment to update $\mathcal{P}'$ by $\mathbf{p}'_i \leftarrow \mathbf{R}_t \mathbf{p}'_i + \mathbf{p}_t$ to computer the Euclidean distance error $e_i = \|\mathbf{x}'_i - \mathbf{p}'_i\|$

5. Update error to check if the loop will be break, otherwise go back to step 2

The implementation for ICP is straightforward, however the performance for ICP can not be guaranteed for many application. There have some significant drawback for this implementation: Firstly, a good initial guess should be provided, otherwise the ICP will be convergent to the local minimum. Secondly, at the step 2, searching for the closest point for each point $\mathbf{p}'_i$ by using the minimal Euclidean distance shares the similar idea of greedy search algorithms, which can also be easily stuck in the local minimum, and step 2 is actually a non-convex optimization problem, which is quite expensive for computation. There have many variants for ICP. like EM-ICP [50], LM-ICP [44], GMM-basedICP [91], Coherent Point Drift (CPD)[113], all of these variants are trying to improve the performance step 2.

In this chapter, I consider the problem from a different angle: aligning two point sets independent of corresponding points. I model one of the point set as an implicit surface,

(a) Thin Plate Spline (b) Radial Basis Function

Figure 4.2: Different kernel functions that can be used for modeling a Gaussian Process Implicit Surfaces.

which can capture the local structure of the surface and then transfer another point set to this implicit surface. I assume that all points on the same surface share the same function value of 0. Then, the point registration problem is turned into finding a transformation that minimizes the objective function

$$E(\mathbf{T}) = \frac{1}{2}\sum_{j=0}^{m}\left\|f(\mathbf{T}\mathbf{y}_j, \mathcal{X})\right\|^2. \tag{4.2}$$

4.4 Preliminary Knowledge

4.4.1 Gaussian Process Implicit Surfaces

In most robotics applications, sensing the environment is very important for knowing the surrounding. There are many methods to describe the object in simulation or the real scenario, for example, mesh with triangles, which are described as vertexes, and face. The data format for standard mesh can be ply, obj, stl, or step file. However, such descriptions have a drawback, which is discrete and can not be directly integrated into an objective function. Many application is required to constraints the contact points on the surface. An alternative solution for describing an object is the so-called Implicit surface [174]. According to the implict function theorem, the implicit function can be formally described as:

$$f(x) = 0, \tag{4.3}$$

where f is a scalar function that takes an input $x \in \mathbb{R}^d$ and results in a $d-1$ dimensional manifold $\mathcal{S}$. For point registration, I will constrain d to dimension 3. Many existing methods can be used to describe the implicit surface, e.g., trimmed B-splines [58], transcendental functions, or thin plate splines [37]. In this work, Gaussian Process Implicit Surfaces [181] will be used to describe the 3D mesh surface. Unlike thin plate splines, shown in Fig. 4.2a, the Radial Basis Function (RBF) kernel (Fig. 4.2b) can only describe a local patch whose distribution will rapidly converge to zero when a point is not near the queried point and not on the surface, which however is in contrast to my assumption. As an alternative, thin plate splines will be selected via

$$k_{ij}(r) = 2r_{\text{dis}}^3 + R_{\text{Mdis}}^3 - 3R_{\text{Mdis}}r_{\text{dis}}^2, \tag{4.4}$$

with a maximum radius of R_{Mdis} inside the training point cloud sets $\mathcal{X}$. The parameter r_{dis} is the distance between two points, which can be computed with following formula:

$$r_{\text{dis}} = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{k=0}^{k=2}(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2}. \tag{4.5}$$

It can be computed as

$$R_{\text{Mdis}} = \max\left\{\|\mathbf{x}_i - \mathbf{x}_j\|\right\}, \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}\,. \tag{4.6}$$

It can be straightforward derived that $k_{ij}(r)$ is bounded

$$0 \leq k_{ij}(r) \leq R^3_{\text{Mdis}} \tag{4.7}$$

and $k_{ij}(r)$ is rotation invariant due to the $\mathbf{R}\|\mathbf{x}_i - \mathbf{x}_j\| = \|\mathbf{x}_i - \mathbf{x}_j\|$.

Gaussian Process Implicit Surfaces can be considered as a standard regression problem, which is expressed as

$$\mathbf{f} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}(\mathcal{X},\mathcal{X}) + \sigma^{\mathrm{T}}\mathbf{I}\sigma\right), \tag{4.8}$$

where $\mathcal{N}$ denotes a normal distribution with the mean $\mu(\mathcal{X}) = \mathbf{0}$ and variance $\mathbf{K}(\mathcal{X},\mathcal{X}) + \sigma^{\mathrm{T}}\mathbf{I}\sigma$. The covariance matrix $\mathbf{K}(.,.)$ consists of $k_{ij}(\mathbf{x}_i, \mathbf{x}_j)$ and σ corresponds to the noise.

The goal therefore is to predict the value by evaluating the following formulas for the queried point $\mathbf{y}_j$:

$$f_*(\mathbf{y}_j|\mathcal{X}) = \mathbf{k}_*^{\mathrm{T}}\left[\mathbf{K} + \sigma^{\mathrm{T}}\mathbf{I}\sigma\right]^{-1}\mathbf{f} = \mathbf{k}_*^{\mathrm{T}}\alpha \tag{4.9}$$

$$V(\mathbf{y}_j|\mathcal{X}) = k(\mathbf{y}_j, \mathbf{y}_j) - \mathbf{k}_*^{\mathrm{T}}\left[\mathbf{K} + \sigma^{\mathrm{T}}\mathbf{I}\sigma\right]^{-1}\mathbf{k}_*\,. \tag{4.10}$$

The kernel function $\mathbf{k}_*(\mathcal{X}, \mathbf{y}_j)$ is used to describe the correlation between source point set $\mathcal{X}$ and target point set $\mathbf{y}_j$. The function value $f_*(\mathbf{y}_j|\mathcal{X})$ is a prediction of $\mathbf{y}_j$ with the corresponding variance, which is expressed as $V(\mathbf{y}_j|\mathcal{X})$ and can be used to evaluate the reliability of the predicted value. For simplification, this value is not included inside the objective function. The 3D model points according to SDF are denoted as $\left\{\mathbf{X}_i \in \mathcal{X}^0 \mid \mathbf{X}_i = \{\mathbf{x}_i, \sigma_i, 0\}\right\}$. To aid the training of an implicit function, two additional point sets will be created: $\left\{\mathbf{X}_i \in \mathcal{X}^1 \mid \mathbf{X}_i = \{\mathbf{x}_i, \sigma_i, 1\}\right\}$ lies outside the surface and $\left\{\mathbf{X}_i \in \mathcal{X}^{-1} \mid \mathbf{X}_i = \{\mathbf{x}_i, \sigma_i, -1\}\right\}$ lies inside the surface. How to generate these two additional point sets is presented in [90]. The final training point sets consist of $\mathcal{X} = \mathcal{X}^0 \cup \mathcal{X}^1 \cup \mathcal{X}^{-1} \in \mathbb{R}^{n\times 3}$.

After modeling the training point sets as the GPIS in the sense of a manifold, the objective function E can be further expressed as

$$f_j(\mathbf{y}_j|\mathcal{X}) = \mathbf{k}_j(\mathcal{X}, \mathbf{y}_j)^{\mathrm{T}}\alpha = \sum\nolimits_{i=0}^{n} k(\mathbf{x}_i, \mathbf{y}_j)\alpha_i \tag{4.11}$$

$$E = \frac{1}{2}\sum\nolimits_{j=0}^{m} f_j^2(\mathbf{y}_j|\mathcal{X}) = \sum\nolimits_{j=0}^{m} \alpha^{\mathrm{T}}\mathbf{k}_j\mathbf{k}_j^{\mathrm{T}}\alpha\,, \tag{4.12}$$

where m is the number of points in the target point cloud and n is the number of points in the source point cloud. f_j is the predictive value given by $\mathbf{y}_j$ that is equal to zero if the target point lies on the mesh surface. The main benefits of this formulation are that no corresponding points between two point sets are required and that it converts the problem into a standard nonlinear squares problem, which can be solved by standard convex solvers.

4.4.2 Lie algebra for optimization

The transformation matrix $\mathbf{T}$ consists of a rotation matrix $\mathbf{R}$ and a translation $\mathbf{p}$. First of all, I will describe some properties of the rotation matrix. $\mathbf{R}$ is a special orthogonal group SO(3), which needs to satisfy the following condition: $\left\{\mathbf{R} \in \mathrm{SO}(3) | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det\mathbf{R} = 1\right\}$. The orthogonal condition imposes the constraints for the rotation matrix, which reduces the number of degrees of freedom to three. Unlike Lie group $SO(3)$, the corresponding Lie algebra $\mathfrak{so}(3) = \{\Phi = \phi^\wedge \in \mathbb{R}^{3\times 3} | \phi \in \mathbb{R}^3\}$ is a vector space with a skew-symmetric matrix Φ.

$$\Phi = \phi^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3},\ \phi \in \mathbb{R}^3\,.$$

The inverse of the operator $(.)^\wedge$ is defined as $(.)^\vee$ and leads to $\phi = \Phi^\vee$. The exponential map will convert the Lie algebra $\mathfrak{so}(3)$ to the Lie group SO(3) by utilizing the matrix exponential formulas: $\mathbf{R} = \exp\left(\phi^\wedge\right)$

$$\exp(\mathbf{A}) = \mathbf{I} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \cdots = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{A}^n \tag{4.13}$$

$$\mathbf{R} = \exp\left(\phi^\wedge\right) = \sum\nolimits_{n=0}^{\infty} \frac{1}{n!}\left(\phi^\wedge\right)^n = \exp(\theta\mathbf{a}) = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{a}\mathbf{a}^T + \sin\theta\mathbf{a}^\wedge, \tag{4.14}$$

with $\mathbf{R} \in \mathrm{SO}(3)$ and $\phi \in \mathbb{R}^3$, $\phi^\wedge \in \mathfrak{so}(3)$. Furthermore, the $\exp(\boldsymbol{\phi}^\wedge)$ can be simplified as:

$$\exp(\boldsymbol{\phi}^\wedge) = \exp(\theta\mathbf{a}) = \cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{a}\mathbf{a}^T + \sin\theta\mathbf{a}^\wedge \tag{4.15}$$

And $\theta = \|\boldsymbol{\phi}\|$ and $\mathbf{a} = \frac{\boldsymbol{\phi}}{\theta}$ is interpreted as unit-length axis of rotation, and $\mathbf{I} \in \mathbb{R}^{3\times3} = \mathbf{a}\mathbf{a}^T - \mathbf{a}^\wedge\mathbf{a}^\wedge$. It can be shown that $\mathbf{R} = \exp\left((\theta + 2\pi m)\mathbf{a}^\wedge\right)$, which indicates the singularity of lie algebra, namely, it has no unique solution for the inverse of lie group, but if by limiting the angle of rotation of $|\theta| \leq \pi$, each rotation $\mathbf{R}$ in lie group SO(3) has a unique solution θ in lie algebra $\mathfrak{so}(3)$, and the calculate can be done:

$$\mathrm{tr}(\mathbf{R}) = \mathrm{tr}\left(\cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{a}\mathbf{a}^\mathrm{T} + \sin\theta\mathbf{a}^\wedge\right) \tag{4.16}$$

$$= \cos\theta\underbrace{\mathrm{tr}(\mathbf{I})}_{3} + (1-\cos\theta)\underbrace{\mathrm{tr}\left(\mathbf{a}\mathbf{a}^\mathrm{T}\right)}_{\mathbf{a}^\mathrm{T}\mathbf{a}=1} + \sin\theta\underbrace{\mathrm{tr}\left(\mathbf{a}^\wedge\right)}_{0} \tag{4.17}$$

$$= 2\cos\theta + 1 \tag{4.18}$$

it can be solved that

$$\theta = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) + 2\pi m \tag{4.19}$$

and I will pick a value which lies inside $|\theta| \leq \pi$, and the axis can be solved by using $\mathbf{R}\mathbf{a} = \mathbf{a}$, namely $\mathbf{a}$ is the eigenvector of $\mathbf{R}$ with the eigenvalue is equal to 1. Similar to the rotation matrix, the transformation matrix $\mathbf{T} = \exp(\xi^\wedge)$ which can be interpreted in terms of Lie groups SE(3) [72] with the corresponding Lie algebra $\mathfrak{se}(3)$.

$$\mathfrak{se}(3) = \left\{\xi = \begin{bmatrix} \rho \\ \phi \end{bmatrix} \in \mathbb{R}^6, \rho \in \mathbb{R}^3, \phi \in \mathfrak{so}(3), \xi^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ 0^\mathrm{T} & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}\right\} \tag{4.20}$$

where ρ can be considered as translation but not equivalent in the translation in the SE(3), and $\phi \in \mathfrak{so}(3)$ has the same meaning shown in the rotation matrix.

4.4.3 Jacobian of Lie Algebra

In this optimized formulation, the target point set is transferred by $\mathbf{T}$ to align the source point set, which is embedded into $f(\mathcal{X})$. The gradient-based optimization uses the Jacobian matrix to search for the minimum solution. In a pure rotation variance point registration, the translational information can be neglected. I will describe in detail the gradient of $\mathbf{R}\hat{y}_j = \frac{\partial(\mathbf{R}\hat{\mathbf{y}}_i)}{\partial\mathbf{R}}$, which is computed based on a gradient perturbation method $\mathbf{R} = \exp(\psi^\wedge)\mathbf{R}_{\mathrm{op}}$ with $\psi \in \mathbb{R}^3$ and leads to two methods can be used to acquire the gradient:

1. Newton's difference quotient, which requires Baker-Campbell-Hausdorff (BCH) formula. The Jacobian matrix will be computed at each iteration, which is complicated.

$$\frac{\partial(\mathbf{R}\hat{y}_i)}{\partial\phi} = \lim_{\phi\to0}\frac{\exp\left((\phi + \delta\phi)^\wedge\right)\hat{y}_i - \exp\left(\phi^\wedge\right)\hat{y}_i}{\delta\phi} \tag{4.21}$$

2. For the first method, Baker-Campbell-Hausdorff (BCH) formula [6] is applied to approximate the formula with small perturbation $\delta\phi$:

$$\exp\Big(\big(\phi + \mathbf{J}_l^{-1}(\phi)\mathbf{J}_l(\phi)\delta\phi\big)^\wedge\Big) = \exp\Big(\big(\mathbf{J}_l(\phi)\delta\phi\big)^\wedge\Big)\exp(\phi^\wedge) \tag{4.22}$$

The drawback of using BCH formula is that it requires to compute Jacobian matrix $\mathbf{J}_l$ at every iteration, which is however computation complicated. Based on those observations, in this paper, the left side perturbation method for calculation of gradient is applied:

$$\begin{aligned}\frac{\partial(\mathbf{R}\hat{\mathbf{y}}_j)}{\partial\psi} &= \lim_{\psi\to 0}\frac{\exp(\psi^\wedge)\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j - \mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j}{\psi} \\ &= \lim_{\psi\to 0}\frac{(\mathbf{I}+\psi^\wedge)\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j - \mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j}{\psi} \\ &= \lim_{\psi\to 0}\frac{-(\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_i)^\wedge\psi}{\psi} = -(\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j)^\wedge,\end{aligned} \tag{4.23}$$

where $a^\wedge b = -b^\wedge a$. Furthermore, by taking into account the transformation information and without consideration of translation, the kernel function $k(\mathbf{y}_j)$ is adapted as $k(\mathbf{R}\hat{\mathbf{y}}_j)$ and is approximated by using the first order of the Taylor series as:

$$\begin{aligned}k_i(\mathbf{R}\hat{\mathbf{y}}_j) &= k_i\big(\exp(\psi^\wedge)\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j\big) \\ &\approx k_i(\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j) + \Big(\frac{\partial k_i}{\partial \mathbf{y}_j}\Big)^{\mathrm{T}}_{|_{\mathbf{y}_j=\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j}}\frac{\partial \mathbf{R}\hat{\mathbf{y}}_j}{\partial\psi}\psi \\ &\approx \underbrace{k_i(\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j)}_{\beta_{i,j}} + \underbrace{\Big(\frac{\partial k_i}{\partial \mathbf{y}_j}\Big)^{\mathrm{T}}_{|_{\mathbf{y}_j=\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j}}\big(-(\mathbf{R}_{\mathrm{op}}\hat{\mathbf{y}}_j)^\wedge\big)}_{\delta_{\mathbf{i,j}}{}^{\mathrm{T}}}\psi,\end{aligned} \tag{4.24}$$

where $\beta_{i,j}\in\mathbb{R}$, $\big(\frac{\partial k_i}{\partial \mathbf{y}_j}\big)^{\mathrm{T}}\in\mathbb{R}^{1\times 3}$, and $\delta_{\mathbf{i,j}}{}^{\mathrm{T}}\in\mathbb{R}^{1\times 3}$ is the gradient vector conditioned on the $\mathbf{y}_j$. It can be shown, that $k_i(\mathbf{R}\mathbf{y}_j)$ can be approximated as $\beta_{i,j}+\delta_{i,j}^{\mathrm{T}}\psi$, which can completely capture the local curvature of the manifold.

In a similar vein, I apply the perturbation method for calculating the gradient of $\mathbf{T}$ and consider the directional of the derivative of $\mathbf{T}$ with respect to the perturbation $\epsilon = [\delta\phi, \delta\rho]\in\mathbb{R}^6$, which can be computed as

$$\begin{aligned}\frac{\partial(\boldsymbol{T}\boldsymbol{p})}{\partial\epsilon} &= \lim_{\epsilon\to 0}\frac{\exp\big(\epsilon^\wedge\big)\exp\big(\xi^\wedge\big)\boldsymbol{p} - \exp\big(\xi^\wedge\big)p}{\epsilon} \\ &\approx \lim_{\epsilon\to 0}\frac{\big(I+\epsilon^\wedge\big)\exp\big(\xi^\wedge\big)\boldsymbol{p} - \exp\big(\xi^\wedge\big)\boldsymbol{p}}{\epsilon} \\ &= \lim_{\epsilon\to 0}\frac{\epsilon^\wedge\exp\big(\xi^\wedge\big)p}{\epsilon} \\ &= \lim_{\epsilon\to 0}\frac{\begin{bmatrix}\delta\phi^\wedge & \delta\rho \\ \mathbf{0}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{R}p+t \\ 1\end{bmatrix}}{\delta\xi} \\ &= \lim_{\epsilon\to 0}\frac{\begin{bmatrix}\delta\phi^\wedge(\boldsymbol{R}\boldsymbol{p}+\boldsymbol{t})+\delta\boldsymbol{\rho} \\ 0\end{bmatrix}}{\epsilon} = \begin{bmatrix}\boldsymbol{I} & -(\boldsymbol{R}\boldsymbol{p}+\boldsymbol{t})^\wedge \\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix} \triangleq (\boldsymbol{T}\boldsymbol{p})^\odot.\end{aligned} \tag{4.25}$$

The operator $(.)^\odot : \mathbb{R}^4 \to \mathbb{R}^{4\times 6}$ is defined as

$$\begin{bmatrix}\boldsymbol{\varepsilon} \\ \eta\end{bmatrix}^\odot = \begin{bmatrix}\eta\mathbf{I} & -\boldsymbol{\varepsilon}^\wedge \\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix} \tag{4.26}$$

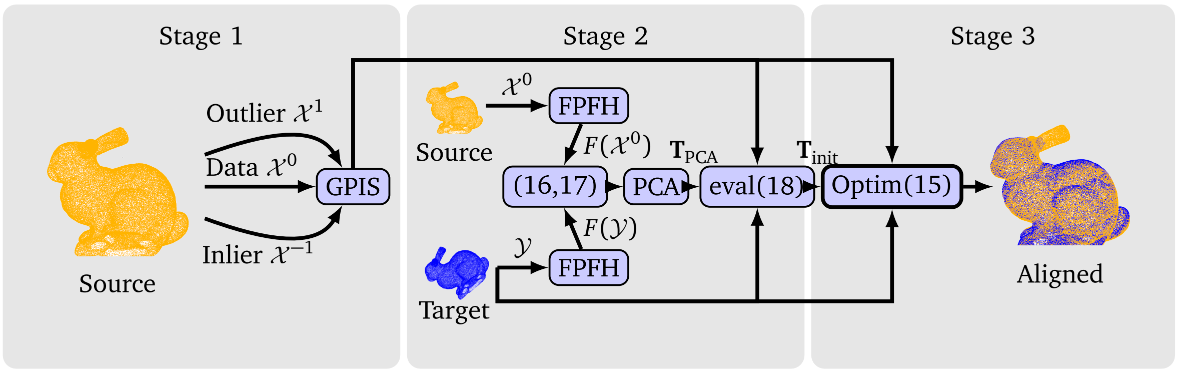


Figure 4.3: The algorithm consists of three stages: In the first stage, two additional point sets $\mathcal{X}^1$ and $\mathcal{X}^2$ are created to augment the original point set as $\mathcal{X} = \mathcal{X}^0 \cup \mathcal{X}^1 \cup \mathcal{X}^{-1}$, which is used to form the implicit function GPIS. In stage two, I compute FPFH for each point in the source and target point set and a cross-checking is executed to identify a corresponding group. The PCA is utilized to compute the initial transformation by evaluating the objective function. In the last stage, the alignment is optimized by a convex solver.

where $\varepsilon \in \mathbb{R}^3$ and η is a scalar that maps the vector space to a higher manifold. For simplification, I omit $\mathbf{x}_i$ in $k_{i,j}(\mathbf{x}_i, \mathbf{T}\hat{\mathbf{y}}_j)$ and use $k_{i,j}(\mathbf{T}\hat{\mathbf{y}}_j)$ instead. The first order Taylor series in kernel function, it can be approximated with respect to the perturbation ϵ as

$$\begin{aligned} k_i(\mathbf{T}\hat{\mathbf{y}}_j) &= k_{i,j}\big(\exp(\epsilon^{\wedge})\mathbf{T}_{\text{op}}\hat{\mathbf{y}}_j\big) \\ &\approx k_{i,j}(\mathbf{T}_{\text{op}}\hat{\mathbf{y}}_j) + \underbrace{\left(\frac{\partial k_{i,j}}{\partial \mathbf{y}_j}\right)^{\mathrm{T}}\Big|_{\mathbf{y}_j=\mathbf{T}_{\text{op}}\hat{\mathbf{y}}_j} \frac{\partial \mathbf{T}\hat{\mathbf{y}}_j}{\partial \epsilon}}_{\frac{\partial k_{i,j}}{\partial \epsilon}} \epsilon \\ &= \beta_{i,j} + \delta_{\mathbf{i},\mathbf{j}}{}^{\mathrm{T}}\epsilon\,, \end{aligned} \tag{4.27}$$

where $\beta_{i,j} = k_i(\mathbf{T}_{\text{op}}\hat{\mathbf{y}}_j) \in \mathbb{R}$, $\left(\frac{\partial k_{i,j}}{\partial \mathbf{y}_j}\right)^{\mathrm{T}} \in \mathbb{R}^{1\times 4}$ and $\delta_{\mathbf{i},\mathbf{j}}{}^{\mathrm{T}} = \left(\frac{\partial k_i}{\partial \mathbf{y}_j}\right)^{\mathrm{T}}\Big|_{\mathbf{y}_j=\mathbf{T}_{\text{op}}\hat{\mathbf{y}}_j}\big((\mathbf{T}\hat{\mathbf{y}}_j)^{\odot}\big) \in \mathbb{R}^{1\times 6}$. The derivative of the kernel function $\frac{\partial k_{i,j}}{\partial \mathbf{y}_j}$ in (4.27) is expressed as

$$\frac{\partial k_{i,j}}{\partial r_{ij}} = 6r_{ij}(r_{ij} - R_{\text{Mdis}}), \qquad \frac{\partial r_{ij}}{\partial \mathbf{y}_j} = \frac{\operatorname{sign}\big(\mathbf{y}_j - \mathbf{x}_i\big)\,|\mathbf{y}_j - \mathbf{x}_i|}{\sqrt{\|\mathbf{y}_j - \mathbf{x}_i\|^2}} \tag{4.28a}$$

$$\begin{aligned} \frac{\partial k_{i,j}}{\partial \mathbf{y}_j} &= \frac{\partial k_{i,j}}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \mathbf{y}_j} = 6r_{ij}(r_{ij} - R_{\text{Mdis}})\frac{\operatorname{sign}\big(\mathbf{y}_j - \mathbf{x}_i\big)\,|\mathbf{y}_j - \mathbf{x}_i|}{\sqrt{\|\mathbf{y}_j - \mathbf{x}_i\|^2}} \\ &= 6(r_{ij} - R_{\text{Mdis}})\operatorname{sign}\big(\mathbf{y}_j - \mathbf{x}_i\big)\,|\mathbf{y}_j - \mathbf{x}_i| \end{aligned} \tag{4.28b}$$

It can be straightforward to adapt the derivative $\frac{\partial k_{i,j}}{\partial r_{ij}}$ if the kernel function is changed.

4.5 GPIS-Based S2S Registration Algorithm

For the partially overlapping situation, the transformation matrix $\mathbf{T}$ is considered in the kernel's corresponding distance function with $r = \|\mathbf{x}_i - \mathbf{R}\hat{y}_j - \mathbf{p}\| = \|\mathbf{x}_i - \mathbf{T}\hat{y}_j\|$. Therefore, the equation (4.11) is updated as $f_j = \mathbf{k}_j(\mathcal{X}, \mathbf{T}\hat{y}_j)^{\mathrm{T}}\alpha$. By combining this with the approximation of the kernel function (4.27), the objective function (4.12) can be further simplified as

$$f_j = \sum\nolimits_{i=0}^{n} (\beta_{i,j} + \delta_{i,j}^{\mathrm{T}} \epsilon)\, \alpha_n = \mathbf{k}_j^{\mathrm{T}}(\beta, \delta^{\mathrm{T}} \epsilon)\, \alpha \tag{4.29}$$

$$E(\mathbf{T}) = \frac{1}{2} \sum\nolimits_{j=0}^{m} \alpha^{\mathrm{T}} \mathbf{k}_j\,(\beta, \delta^{\mathrm{T}} \epsilon)\, \mathbf{k}_j^{\mathrm{T}}(\beta, \delta^{\mathrm{T}} \epsilon)\, \alpha\,, \tag{4.30}$$

with $\mathbf{k}_j^{\mathrm{T}}(\beta, \delta^{\mathrm{T}}\epsilon) = \left[\beta_{1,j} + \delta_{1,j}^{\mathrm{T}}\epsilon, \quad \cdots, \quad \beta_{n,j} + \delta_{n,j}^{\mathrm{T}}\epsilon\right] \in \mathbb{R}^{1\times n}$. As a result, (4.30) is converted to a nonlinear quadratic equation with the approximated nonlinear kernel function $\mathbf{k}_j(\beta, \delta^{\mathrm{T}}\epsilon)$ and the argument for this optimized problem is changed from the Lie group $\mathbf{T} \in \mathrm{SE}(3)$ to the perturbation variable of the Lie algebra $\epsilon \in \mathfrak{se}(3)$.

4.5.1 Gradient of objective function

By taking the derivative of $E(\mathbf{T})$ with respect to ϵ^{T}, I get

$$\begin{aligned}
\frac{\partial E(\mathbf{T})}{\partial \epsilon^{\mathrm{T}}} &= \sum\nolimits_{j=0}^{m} \frac{\partial\big(\alpha^{\mathrm{T}} \mathbf{k}_j(\beta, \delta^{\mathrm{T}}\epsilon)\big)}{\partial \epsilon^{\mathrm{T}}} \mathbf{k}_j^{\mathrm{T}}(\beta, \delta^{\mathrm{T}}\epsilon)\alpha \\
&= \sum\nolimits_{j=0}^{m} \Big(\sum\nolimits_{i=0}^{n} \frac{\partial(\alpha_i\beta_i + \alpha_i \epsilon^{\mathrm{T}}\delta_i)}{\partial \epsilon^{\mathrm{T}}}\Big) \mathbf{k}_j^{\mathrm{T}}(\beta, \delta^{\mathrm{T}}\epsilon)\alpha \\
&= \sum\nolimits_{j=0}^{m} \Big(\sum\nolimits_{i=0}^{n} \alpha_i \delta_{i,j}\Big) \mathbf{k}_j^{\mathrm{T}}(\beta, \delta^{\mathrm{T}}\epsilon)\alpha \\
&= \sum\nolimits_{j=0}^{m} \mathbf{J}_j \Big(\sum\nolimits_{i=0}^{n} \beta_{i,j}\alpha_i + \delta_{i,j}^{\mathrm{T}} \alpha_i \epsilon\Big),
\end{aligned} \tag{4.31}$$

where $\mathbf{J}_j$ is defined as $\sum_{i=0}^{n} \alpha_i \delta_{i,j} \in \mathbb{R}^{6\times 1}$ and can capture the surface's curvature by summing up all gradients in the kernel function. I can further define $\mathcal{X}_j = \sum_{i=0}^{n} \beta_{i,j}\alpha_i$. To get the optimum perturbation $\epsilon^\star$ at the current position, the formula $\frac{\partial E(\mathbf{T})}{\partial \epsilon^{\mathrm{T}}}$ is forced to be zero, leading to

$$\sum\nolimits_{j=0}^{m} \mathbf{J}_j \sum\nolimits_{i=0}^{n} \delta_i^{\mathrm{T}} \alpha_i \epsilon^\star = -\sum\nolimits_{j=0}^{m} \mathbf{J}_j \mathcal{X}_j$$

$$\mathbf{J}\epsilon^\star = -\sum\nolimits_{j=0}^{m} \mathbf{J}_j \mathcal{X}_j \tag{4.32}$$

$$\epsilon^\star = -\mathbf{J}^{-1} \sum\nolimits_{j=0}^{m} \mathbf{J}_j \mathcal{X}_j\,, \tag{4.33}$$

with $\mathbf{J} = \sum_{j=0}^{m} \mathbf{J}_j \mathbf{J}_j^{\mathrm{T}} \in \mathbb{R}^{6\times 6}$. $\mathbf{T}$ is therefore updated as

$$\mathbf{T}_{\mathrm{op},h} \leftarrow \exp\big((\epsilon^\star)^\wedge\big)\mathbf{T}_{\mathrm{op},h-1}, \tag{4.34}$$

which captures the local structural manifold by means of the Lie algebra. The optimization process follows the principle of the Gauss-Newton algorithm. I can further adapt $\mathbf{J}$ as $\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J})$, which is the LM algorithm.

4.5.2 Initial alignment using PCA and FPFH

A good initial guess for the optimization is important in order to guarantee a good result and run-time. I present a new method for computing the initial alignment by employing the PCA and FPFH [148] algorithms. First, a FPFH is calculated for each point in the two point sets, which are referred to as $F(\mathcal{X}^0)$ and $F(\mathcal{Y})$. I then embed $F(\mathcal{X}^0)$ into a k-d tree $\mathrm{Kd}_{F(\mathcal{X}^0)}$ and a nearest neighbor search is performed for each feature $F(\mathbf{y}_j) \in F(\mathcal{Y})$, such that $\mathcal{G}_1$ is a group pair set of the results $\mathbf{x}_{i|j}$ for the queries $\mathbf{y}_j$:

$$\mathcal{G}_1 = \left\{ \{\mathbf{y}_j, \mathbf{x}_{i|j}\} \mid \mathrm{Kd}_{F(\mathcal{X}^0)}\big(F(\mathbf{y}_j)\big), \, \forall \mathbf{y}_j \in \mathcal{Y} \right\}. \tag{4.35}$$

Subsequently, I embed $F(\mathcal{Y})$ into another k-d tree $\mathrm{Kd}_{F(\mathcal{Y})}$ and perform a nearest neighbor search for each result stored in $\mathcal{G}_1$, such that $\mathcal{G}_2$ is a group pair set of the results $\mathbf{y}_{j|i}$ for the queries $\mathbf{x}_{i|j}$ that are stored in $\mathcal{G}_1$:

$$\mathcal{G}_2 = \left\{ \{\mathbf{x}_{i|j}, \mathbf{y}_{j|i}\} \mid \mathrm{Kd}_{F(\mathcal{Y})}\big(F(\mathbf{x}_{i|j})\big), \, \forall \mathbf{x}_{i|j} \in \mathcal{G}_1 \right\} \tag{4.36}$$

I only keep the subset $\mathcal{G}_1 \cap \mathcal{G}_2$ that contains bidirectional nearest neighbors and refer to these as $\mathcal{X}'$ and $\mathcal{Y}'$. Statistical analysis techniques are then applied to remove any outliers in these groups [149]. The final selected points are grouped together and are denoted as $\mathcal{X}_{\text{group,FPFH}} \in \mathbb{R}^{n_{\text{FPFH}} \times 3}$ and $\mathcal{Y}_{\text{group,FPFH}} \in \mathbb{R}^{m_{\text{FPFH}} \times 3}$. After this, a PCA is used to compute the initial transformation $\mathbf{T}_{\text{PCA}}$ between $\mathcal{X}_{\text{group,FPFH}}$ and $\mathcal{Y}_{\text{group,FPFH}}$. Given a point cloud $\mathbf{P} \in \mathbb{R}^{n \times 3}$, PCA is performed by

$$\frac{\sum \left(\mathbf{P}_i - \overline{\mathbf{P}}\right)\left(\mathbf{P}_i - \overline{\mathbf{P}}\right)^T}{n} = \mathbf{E}\Lambda\mathbf{E}^T \tag{4.37}$$

where $\mathbf{P}_i \in \mathbb{R}^3$ is the i^{th} point of $\mathbf{P}, \overline{\mathbf{P}} \in \mathbb{R}^3$ is the center of $\mathbf{P}, \mathbf{E}$ is the eigenvector matrix composed of eigenvectors $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ (a.k.a., principal axes), and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$ are the corresponding eigenvalues (a.k.a.,principal values). By aligning the principal axes to the three axes of the world coordinate, I obtain the canonical pose as $\mathbf{P}_{\text{can}} = \mathbf{PE}$. The rotation-invariant property of $\mathbf{P}_{\text{can}}$ can be facilely derived as follows: By multiplying an arbitrary rotation matrix SO(3), I can get an rotated version of $\hat{\mathbf{P}} = \mathbf{RP}$

$$\frac{\sum \left(\mathbf{RP}_i - \mathbf{R}\overline{\mathbf{P}}\right)\left(\mathbf{RP}_i - \mathbf{R}\overline{\mathbf{P}}\right)^T}{n} = \mathbf{R}\left(\frac{\sum \left(\mathbf{P}_i - \overline{\mathbf{P}}\right)\left(\mathbf{P}_i - \overline{\mathbf{P}}\right)^T}{n}\right)\mathbf{R}^{\mathrm{T}} = (\mathbf{RE})\Lambda(\mathbf{RE})^{\mathrm{T}} \tag{4.38}$$

where $\mathbf{RE}$ becomes the new principal axes. Therefore, I can have an rotation invariant canonical pose

$$(\mathbf{PR}^{\mathrm{T}})_{\text{can}} = \mathbf{PR}^{\mathrm{T}} \cdot \mathbf{RE} = \mathbf{PE} = \mathbf{P}_{\text{can}} \tag{4.39}$$

However, the PCA-based transformation contains the sign ambiguities, since the eigenvector with $\pm$ can satisfy the eigen decomposition. Consequently, There have totally 8 different eigenvectors variants by by assigning different signs. Note, that $\mathbf{T}_{\text{PCA}}$ has four different possibilities according to the right-hand rule (Fig. 4.4). In this four cases, the determinate of $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ is equal to one, see Table 4.1 For choosing the properly $\mathbf{T}_{\text{PCA}}$, I can evaluate the formula

$$\mathbf{T}_{\text{init}} \leftarrow \min_{k \in \{0, \cdots 3\}} \sum\nolimits_{j=0}^{m} \left\| f\left(\mathbf{T}_{\text{PCA},k} \hat{\mathbf{y}}_j, \mathcal{X}\right) \right\|^2, \tag{4.40}$$

I select the transformation matrix $\mathbf{T}_{\text{init}}$ that has the smallest function value. The whole process is illustrated in Fig. 4.3 and the algorithm is summarized in Alg. 1.

4.6 Evaluation

In order to compare my algorithm against the state of the art, I evaluated it against other registration algorithms regarding accuracy RMSE and time on the Stanford 3D Scanning Repository's Happy Buddha, Stanford Bunny, and Chinese Dragon [173] as well as Blender's Suzanne model. PCL-ICP [149] is the standard implementation of the ICP algorithm in the

Table 4.1: Determinants and geometric meanings of the eigenvector matrix by assigning different signs.

	Determinant	Geometric meaning
$(+\mathbf{e}_1, +\mathbf{e}_2, +\mathbf{e}_3)$	1	Rotation 4.4a
$(-\mathbf{e}_1, -\mathbf{e}_2, +\mathbf{e}_3)$	1	Rotation 4.4b
$(+\mathbf{e}_1, -\mathbf{e}_2, -\mathbf{e}_3)$	1	Rotation 4.4c
$(-\mathbf{e}_1, +\mathbf{e}_2, -\mathbf{e}_3)$	1	Rotation 4.4d
$(-\mathbf{e}_1, +\mathbf{e}_2, +\mathbf{e}_3)$	−1	Improper rotation
$(+\mathbf{e}_1, -\mathbf{e}_2, +\mathbf{e}_3)$	−1	Improper rotation
$(+\mathbf{e}_1, +\mathbf{e}_2, -\mathbf{e}_3)$	−1	Improper rotation
$(-\mathbf{e}_1, -\mathbf{e}_2, -\mathbf{e}_3)$	−1	Improper rotation

Algorithm 1 Optimization of transformation matrix by Gauss-Newton/Levenberg-Marquardt algorithm

```
Require: $\mathcal{X}, \mathcal{Y}, H$
 1: Modeling GPIS $f(\mathcal{X})$                                        ▷ Section 4.4.1
 2: Compute FPFH features $F(\mathcal{X}^0)$ and $F(\mathcal{Y})$          ▷ Section 4.5.2
 3: Calculate $\mathcal{G}_1$ and $\mathcal{G}_2$                           ▷ (4.35), (4.36)
 4: $\{\mathcal{X}_{\text{group,FPFH}}, \mathcal{Y}_{\text{group,FPFH}}\} \leftarrow \text{StaticalRemove}(\mathcal{G}_1 \cap \mathcal{G}_2)$
 5: $\mathbf{T} \leftarrow \text{PCA}(\mathcal{X}_{\text{group,FPFH}}, \mathcal{Y}_{\text{group,FPFH}})$   ▷ (4.40)
 6: $\mathbf{T}_{\text{op},0} \leftarrow \arg\min \sum_{j=0}^{m} f(\mathcal{X}, \mathbf{T}\mathbf{y}_j)$
 7: for $h = 1 : H$ do
 8:     Approximate $k_i(\mathbf{y}_j)\ \forall \mathbf{y}_j \in \mathcal{Y}$   ▷ (4.27)
 9:     if Gauss-Newton then
10:         Set $\mathbf{J} = \sum_{j=0}^{m} \mathbf{J}_j \mathbf{J}_j^{\mathrm{T}}$
11:     end if
12:     if Levenberg-Marquardt then
13:         Set $\mathbf{J} = \sum_{j=0}^{m} \mathbf{J}_j \mathbf{J}_j^{\mathrm{T}} + \lambda\, \text{diag}(\mathbf{S})$
14:     end if
15:     calculate $\epsilon^\star = -\mathbf{J}^{-1} \sum_{j=0}^{m} \mathbf{J}_j \sum_{i=0}^{n} \beta_i \alpha_i$   ▷ (4.33)
16:     Update $\mathbf{T}_{\text{op,h}}$                                    ▷ (4.34)
17:     if $\|\epsilon^\star\|_F \leq \epsilon$ then break
18:     end if
19: end for
20: return $\mathbf{T}_{\text{op}^\star}$
```

Point Cloud Library, which is a local registration algorithm as it relies on a good initial alignment. SAC-IA-ICP [148] employs FPFH to get the initial alignment and then uses ICP to iteratively align the two point clouds. It is therefore considered a global point registration algorithm. GoICP and its variant GoICPT with trimming [187] are global registration algorithms that use the BnB algorithm in their implementations and also support partial overlapping point registration. Global registration RANSAC (Gl.RANSAC) [197] requires no initial alignment. Instead, it utilizes RANSAC for the initialization alignment by searching corresponding points in the FPFH feature space. Fast Global Registration (FGR) is another registration algorithm that utilizes FPFH for searching corresponding points. The algorithm presented in this paper is labeled GPIS-S2SPR. In this section, three different experiments were conducted. In order to reduce the computational burden, the tested point sets were downsampled for every algorithm into small scale numbers (1500–2500) by using voxel filtering. All evaluations were performed on a laptop with a 2.6 GHz Intel Core i7-6700HQ and 16 GB of RAM.

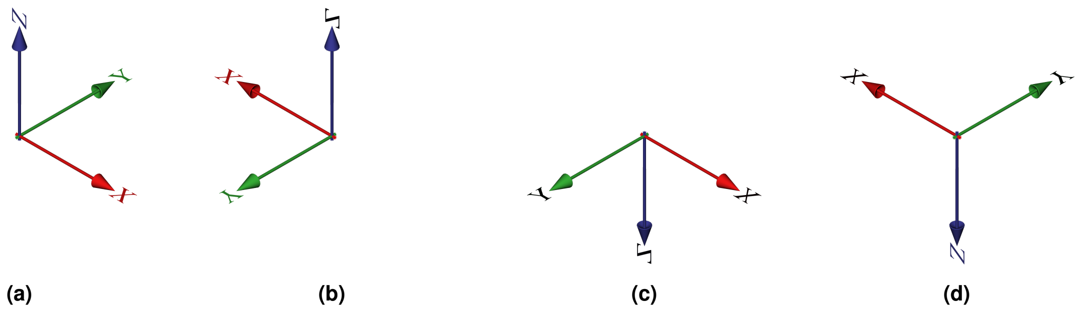


Figure 4.4: The four possible coordinate systems for the PCA when computing the initial transformation matrix.

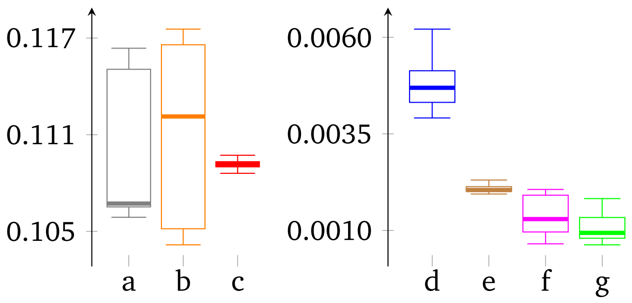


Figure 4.5: RMSE for the Stanford Bunny with partial overlap (85%) without Gaussian noise: (a) GoICP, (b) GoICPT (10%), (c) PCL-ICP, (d) Gl.RANSAC, (e) SAC-IA-ICP, (f) FGR, (g) GPIS-S2SPR.

4.6.1 RMSE for random transformations

For exploring the capability of my algorithm, I evaluated algorithms using the Stanford Bunny point set with 50 290.000 points without Gaussian noise and only partial overlap (85%). I reduced the number points by applying a voxel grid filter with a size of 0.005 [149]. I ran each algorithm on a set of 40 random transformation matrices (Fig. 4.5).

From the results, I can see that GoICP and its variant GoICP with trimming (10%), as suggested in [187], performed worse in this experiment with median values of 0.110 and 0.108. PCL-ICP was evaluated using an identity matrix as initialization and behaved slightly better with a median value of 0.109 and a smaller variance in comparison to GoICP. Gl.RANSAC showed significant improvement with a median value of 0.005 and SAC-IA-ICP achieved a median value of 0.002. FGR achieved a median value of 0.001. In contrast to the previous algorithms however, its mean value of 0.017 deviated from the median and is much higher. This is due to its lack of robustness in handling large rotational changes, which is further evaluated in the next experiment (Section 4.6.2). My algorithm showed the best overall performance, with a median value of 0.001 and a small variance.

4.6.2 Rotation and translation invariance

Rotation and translation invariance are essential factors for point registration. I conducted two experiments with the Stanford Bunny dataset to evaluate these two properties. For the first experiment, I rotated the source point set 50.000° around the y axis and translated it with a vector of $[0.1, 0.2, 0.3]$, as illustrated in the first row of Fig. 4.6. In the second one, I rotated the point set 180.000° around the z axis without translation, as shown in the second row of Fig. 4.6. The same initial alignment is used for each algorithm in both cases. I repeated both experiments 40 times. The best results are shown in Fig. 4.6. PCL-ICP and FGR show entirely different behaviors in these two cases, while they failed with a more than 0.1 RMSE

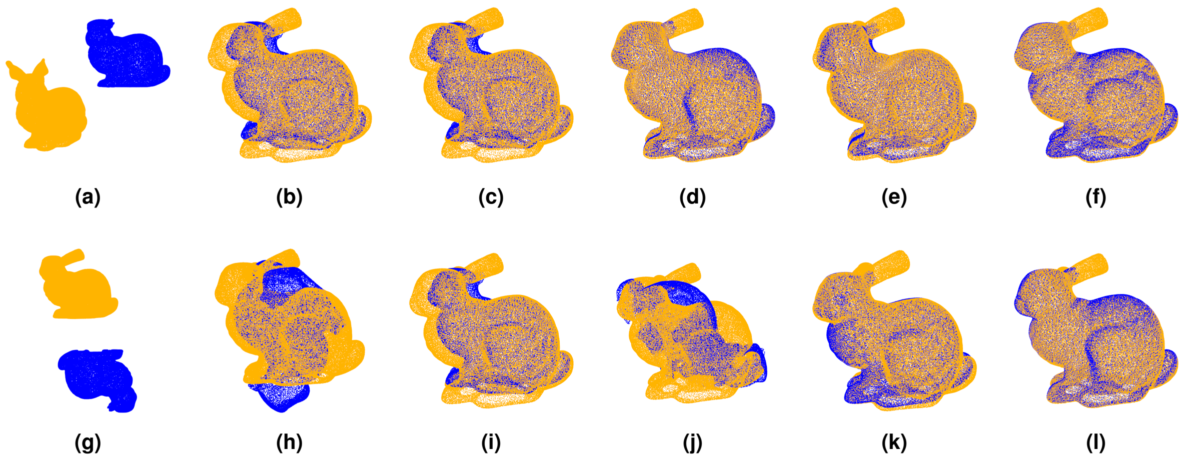


Figure 4.6: Stanford Bunny together with alignment results of selected algorithms for a–f 50.000° rotation around the y axis and small translation, g–l 180.000° rotation around the z axis with translation set to zero. The first column a g indicated Source/Target. From the second to last column are the algorithms for the comparison: PCL-ICP, SAC-IA-ICP, FGR, GL.RANSANC, and GPIS-S2SPR.

value in the second case. For these two algorithms, I conducted further experiments with different rotations. FGR failed with a high probability for high rotation values and I therefore conclude, that FGR is not rotation invariant. SAC-IA-ICP showed no significant difference in both experiments, achieving roughly the same mean value of 0.009. Gl.RANSAC also showed similar performance in both cases with mean values of 0.003 and 0.006. In this experiment, rotation and translation showed no significant effect in my GPIS-S2SPR approach, with an approximate RMSE of 0.002.

4.6.3 Noise and overlap robustness

I evaluated the algorithms on all four point sets with the number of points varying from 30000 to 50000. Furthermore, I applied three different levels of noise based on a Gaussian distribution with variances set to 0, 0.00025, and 0.0005, respectively. I also evaluated the capability of point registration in a partially overlapping scenario, where only a subset of the points from the source point cloud is used for the target point cloud. Three different overlap factors were used in the experiments: 100%, 85%, and 65%. Furthermore, I used an identity matrix for the initial alignment in each test to maintain identical conditions. Each algorithm was executed 40 times for each configuration, leading to a total of 1440 times for all possible combinations. The results with individual RMSE values ϵ and runtime t for each configuration are listed in Table 4.2. The algorithms GoICP and GoICPT (10% trimming) consistently showed the worst performance in all test cases with regard to the mean value of RMSE and total computation time. The BnB algorithm in these algorithms is very expensive to compute and the constant switch between ICP and BnB was not able to achieve a global optimum solution. PCL-ICP was able to converge very fast but is susceptible to local minima. SAC-IA-ICP achieved the best performance in case of an overlapping factor of 100% with an RMSE of 0.001. However, the RMSE value increased drastically to 0.010 in case of a partial overlap, Gl.RANSAC explores the corresponding points in terms of FPFH and showed similar performance. FGR converged very fast but showed a bad performance in all experiments for the reasons explained in Section 4.6.2. My GPIS-S2SPR algorithm does not rely on identifying corresponding points and is therefore stable for different transformations. In this experiment, the RMSE for GPIS-S2SPR is stable in terms of the overlapping factor and

Table 4.2: Benchmark results for all algorithms on four different point sets with three levels of Gaussian noise and three different overlap factors. The best RMSE value ϵ for each configuration is highlighted in *green*.

		noise = 0.00000						noise = 0.00025						noise = 0.00050					
		1.00		0.85		0.65		1.00		0.85		0.65		1.00		0.85		0.65	
		ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t
Bunny [173]	PCL-ICP	0.081	0.5	0.090	0.4	0.067	0.3	0.046	0.4	0.082	0.6	0.046	0.3	0.069	0.4	0.107	0.6	0.073	0.3
	GoICP	0.115	20.3	0.110	20.1	0.097	20.0	0.046	20.1	0.024	20.1	0.063	20.2	0.089	20.1	0.101	20.1	0.046	20.1
	GoICPT	0.100	21.5	0.108	21.5	0.099	21.4	0.106	21.5	0.104	21.7	0.098	21.4	0.103	21.4	0.109	21.7	0.103	21.4
	SAC-IA-ICP	0.001	6.1	0.002	5.6	0.010	7.1	0.001	6.6	0.002	6.1	0.011	8.2	0.001	7.0	0.003	6.4	0.011	7.9
	Gl.RANSAC	0.001	1.6	0.005	1.7	0.005	2.3	0.001	1.7	0.005	1.8	0.005	1.9	0.001	1.7	0.005	1.9	0.005	2.3
	FGR	0.017	0.4	0.009	0.4	0.017	0.3	0.004	0.4	0.007	0.4	0.020	0.3	0.004	0.4	0.007	0.4	0.015	0.3
	GPIS-S2SPR	0.001	0.6	0.001	0.5	0.001	0.5	0.001	0.5	0.001	0.6	0.002	0.5	0.001	0.5	0.002	0.6	0.002	0.7
Suzanne [11]	PCL-ICP	0.134	0.8	0.108	0.8	0.116	0.7	0.049	0.6	0.127	1.3	0.095	0.6	0.131	0.9	0.115	0.6	0.106	0.9
	GoICP	0.089	23.5	0.055	21.6	0.086	22.0	0.089	21.8	0.081	21.8	0.089	21.8	0.064	21.8	0.072	21.7	0.117	21.5
	GoICPT	0.092	23.2	0.059	21.5	0.084	21.5	0.079	21.6	0.071	21.7	0.097	21.7	0.045	21.5	0.091	21.6	0.062	21.5
	SAC-IA-ICP	0.001	11.2	0.005	10.6	0.018	10.3	0.001	11.7	0.006	10.7	0.027	11.4	0.001	12.4	0.006	11.5	0.018	11.1
	Gl.RANSAC	0.014	1.7	0.018	1.7	0.040	2.0	0.016	1.8	0.011	1.8	0.039	2.3	0.013	1.9	0.025	1.9	0.022	2.7
	FGR	0.049	0.6	0.039	0.6	0.060	0.5	0.070	0.6	0.049	0.6	0.061	0.5	0.046	0.7	0.071	0.6	0.052	0.6
	GPIS-S2SPR	0.003	0.8	0.002	1.1	0.002	1.1	0.001	4.3	0.002	2.1	0.002	1.8	0.002	1.3	0.003	1.2	0.002	1.6
Dragon [173]	PCL-ICP	0.087	0.5	0.100	0.3	0.079	0.4	0.090	0.2	0.065	0.3	0.096	0.4	0.071	0.5	0.082	0.5	0.101	0.4
	GoICP	0.017	21.6	0.022	21.7	0.018	21.4	0.034	21.5	0.035	21.7	0.021	21.5	0.018	21.5	0.053	21.5	0.033	21.6
	GoICPT	0.022	21.4	0.014	21.4	0.021	20.0	0.011	19.9	0.009	20.0	0.084	20.1	0.013	20.1	0.044	20.2	0.017	20.2
	SAC-IA-ICP	0.001	5.9	0.003	5.3	0.009	4.8	0.001	6.0	0.003	5.4	0.009	4.9	0.001	6.5	0.004	5.8	0.009	5.2
	Gl.RANSAC	0.001	2.1	0.005	2.6	0.005	2.9	0.001	2.3	0.005	2.5	0.004	2.8	0.001	2.6	0.005	2.5	0.005	2.8
	FGR	0.012	0.5	0.022	0.5	0.024	0.4	0.012	0.5	0.016	0.4	0.017	0.4	0.021	0.5	0.015	0.5	0.013	0.4
	GPIS-S2SPR	0.002	0.7	0.002	0.9	0.002	1.0	0.002	0.8	0.002	0.9	0.003	1.0	0.002	0.7	0.002	1.0	0.003	0.9
Buddha [173]	PCL-ICP	0.094	0.2	0.086	0.4	0.110	0.3	0.095	0.5	0.076	0.2	0.071	0.2	0.124	0.4	0.047	0.4	0.064	0.4
	GoICP	0.043	21.5	0.032	21.4	0.085	21.3	0.075	21.4	0.050	21.3	0.052	21.3	0.051	21.4	0.067	21.5	0.012	21.4
	GoICPT	0.013	20.2	0.025	20.1	0.028	20.2	0.016	20.2	0.031	20.2	0.022	20.1	0.023	20.0	0.031	19.9	0.020	20.0
	SAC-IA-ICP	0.001	5.5	0.007	5.2	0.009	6.3	0.001	6.2	0.014	7.3	0.017	7.4	0.001	6.7	0.011	7.5	0.015	6.2
	Gl.RANSAC	0.002	1.1	0.005	1.3	0.006	1.8	0.002	1.0	0.005	1.4	0.006	1.6	0.002	1.1	0.005	1.5	0.005	1.8
	FGR	0.022	0.4	0.022	0.4	0.019	0.3	0.016	0.4	0.019	0.4	0.015	0.3	0.024	0.4	0.022	0.4	0.021	0.3
	GPIS-S2SPR	0.003	1.0	0.002	0.8	0.002	0.8	0.001	0.6	0.003	0.7	0.002	0.8	0.002	0.7	0.002	0.7	0.003	1.2

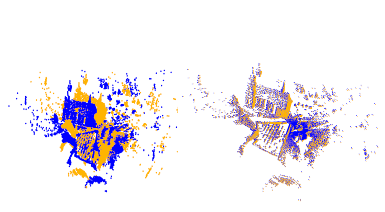
Table 4.3: Mean RMSE over all noise levels and overlap factors for all point sets and algorithms. The best result is highlighted in *green*.

Data	PCL-ICP	GoICP	GoICPT	SAC-IA-ICP	Gl.RANSAC	FGR	**GPIS-S2SPR**
Bunny	0.073	0.077	0.103	0.005	0.004	0.011	0.001
Suzanne	0.109	0.083	0.076	0.009	0.022	0.055	0.002
Dragon	0.086	0.028	0.013	0.004	0.004	0.017	0.002
Buddha	0.085	0.052	0.052	0.008	0.004	0.020	0.002
Mean	0.088	0.060	0.061	0.007	0.009	0.026	0.002

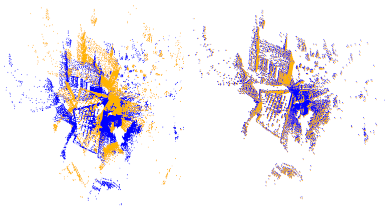
noise for all point sets. As it is able to consider noise in its formulation, the RMSE is approximated equal to 0.002 for all noise levels. Table 4.3 shows the total RMSE computed over all possible combinations for each point set and algorithm. GPIS-S2SPR showed the best overall performance. In terms of computation time, it is not always the fastest approach, since GPIS is time-consuming due to the computation of an inverse of the covariance matrix with a complexity of $\mathcal{O}(m^3)$.

4.6.4 Evaluation with scanned datasets

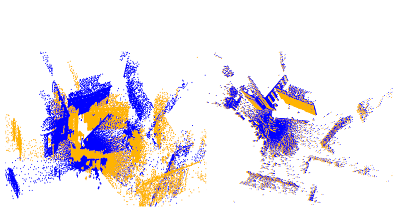
To further verify my algorithm, I evaluate the point sets from semantic-8 [54] and Urban Scenes Velodyne Point Cloud Dataset [83]. The corresponding results are shown in Fig. 4.7. I compare my algorithm with PCL-ICP in Fig. 4.7a, where the RMSE of PCL-ICP is 48 times that of my algorithm. In Fig. 4.7b–4.7f, each sub-figure consists of two images, where the left one is the initial state, and the right one is the result of point registration. It can be seen that my algorithm can work in different scenarios, such as urban scenes [83] and different kinds



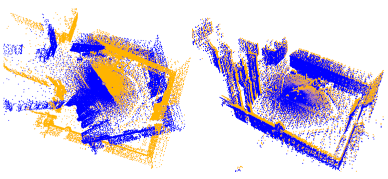
(a) bildstein ICP/ours



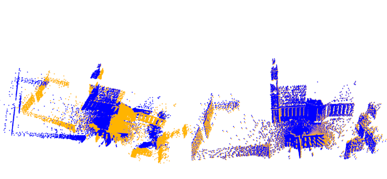
(b) bildstein



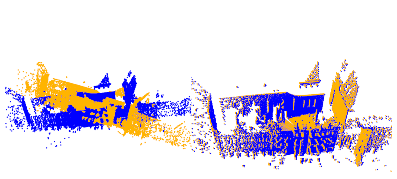
(c) stgallencathedral6



(d) stgallencathedral1



(e) stgallencathedral3



(f) neugasse

Figure 4.7: Comparison between PCL-ICP and GPIS-S2SPR with examples from the large scale point cloud classification dataset Semantic-8 [54]. In a, the alignment result of PCL-ICP is on the left and the result of GPIS-S2SPR on the right. From b–f, left shows the initial pose of source and target point set, while right shows the result of applying GPIS-S2SPR.

of buildings. Furthermore, I evaluate my algorithm with two additional point sets from [197] and Shapenet [22], which are shown in Fig. 4.8. The source point sets in Fig. 4.8 are indicated as blue points and the target point sets as orange points. The initial setting for source and target point sets are demonstrated in Fig. 4.8a, 4.8b and 4.8c. From Fig. 4.8d, 4.8e, and 4.8f, I can see that the alignment accuracy is very high in both point sets with an RMSE value of 0.002, 0.0001, and 0.0001, respectively.

4.7 Conclusion

I propose a new algorithm for a partially overlapping 3D surface registration algorithm. In this algorithm, I abandon the traditional idea of point to point or point to plane correspondence search to register the points. Instead, I view the 3D surface as a Gaussian Process Implicit Surfaces, which utilizes the signed distance function to describe three manifolds. Furthermore, I convert the point registration as a nonlinear least-squares problem to find a rigid transformation between two point sets. For accelerating the optimization process, I use a Principal Component Analysis (PCA) together with Fast Point Feature Histograms descriptors to compute the initial transformation. Moreover, I derive the Jacobian matrix by applying the Lie algebra perturbation method, which approximated the kernel function with the first-order Taylor series. The whole optimization follows the principle of Gauss-Newton algorithm. By slightly adapting the Jacobian matrix with a damping value, I can convert the algorithm to Levenberg-Marquardt solver. My approach demonstrated a higher accuracy performance and more robust rotation invariant properties compared to state-of-the-art methods.

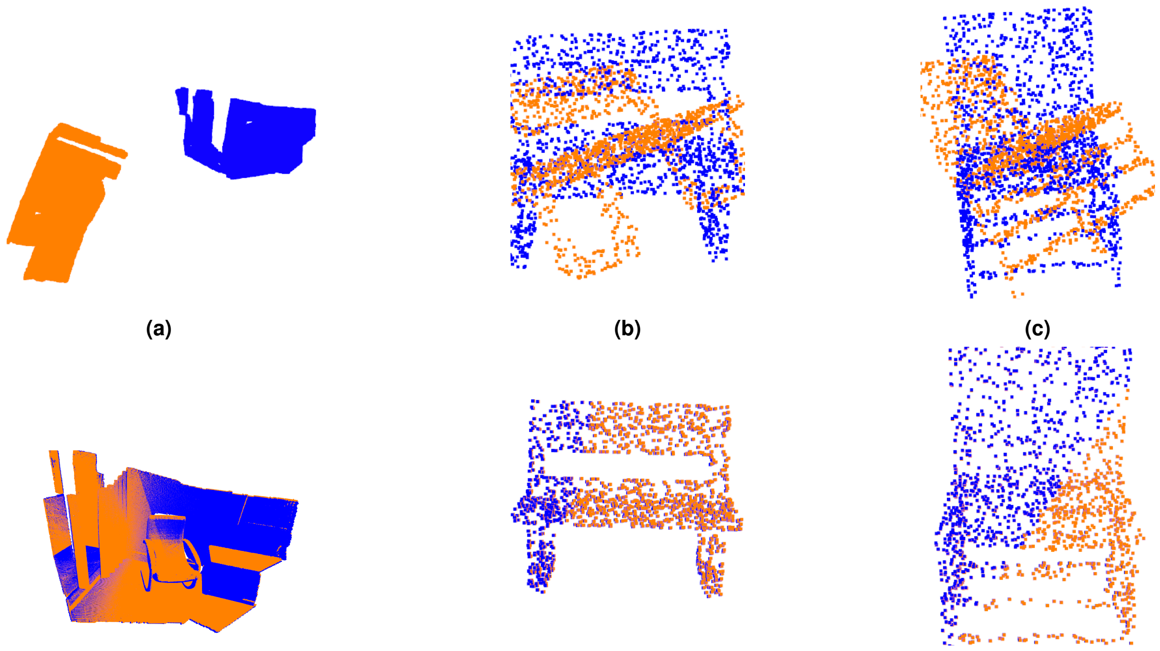


Figure 4.8: Evaluation of GPIS-S2SPR with additional scanned point sets from [197] and Shapenet [22]. The first row shows the initial setting for source and target pointsets. The second row shows the alignment results.

Chapter 5
Rotation Invariance Learning for Point Cloud Classification

Chapter Summary

Rotation invariance is a crucial property for 3D object classification, which is still a challenging task. State-of the- art deep learning-based works require a massive amount of data augmentation to tackle this problem. This is however inefficient and classification accuracy suffers a sharp drop in experiments with arbitrary rotations. I introduce a new descriptor that can globally and locally capture the surface geometry properties and is based on a combination of spherical harmonics energy and point feature representation. The proposed descriptor is proven to fulfill the rotation-invariant property. A limited bandwidth spherical harmonics energy descriptor globally describes a 3D shape and its rotation invariant property is proven by utilizing the properties of a Wigner D-matrix, while the point feature representation captures the local features with a KNN to build the connection to its neighborhood. I propose a new network structure by extending PointNet++ with several adaptations that can hierarchically and efficiently exploit local rotation invariant features.

This chapter is a slightly modified version of peer-reviewed conference paper ©2021 IEEE. Reprinted, with permission, from

- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, “Deep Hierarchical Rotation Invariance Learning with Exact Geometry Feature Representation for Point Cloud Classification,” IEEE International Conference on Robotics and Automation (ICRA), 2021

The images created, algorithms designed, data from experiments and text written by me in this publication will be directly referenced in this chapter. The original version is referred to 186

Contributions

I took a leading role in the writing and revising of the manuscript in this article. I have made the following significant personal contributions to the formulation, implementation, and evaluation of the algorithms in this paper: identifying the problem in the point cloud-based object classification, introducing the spherical harmonic based rotation invariance descriptor, designing the neural network. I am the leading developer of the algorithm implementation and am responsible for experimental evaluation

5.1 Introduction

Convolutional neural networks (CNN) [86] have shown tremendous success in image processing due to their translation-invariant capability of detecting local patterns regardless of their position in the image and their ability to process regular data, such as image grids or 3D voxels. However, the more challenging rotation-invariant property is still missing in the designed structure [25]. Data augmentation is a common approach to address this issue. The infinite property of the rotation group however makes this approach less efficient and comes with a high computational cost. A big neural network with rotation-augmented data is required to generalize the data set. In 3D, geometric irregular data formats such as point clouds increase the difficulty of handling the rotation transformation, while irregular data formats suffer from a permutation problem $N!$. To address this issue and to inherit the benefits of convolutional networks, which can process regular data formats, previous work such as [26, 105] voxelized geometric shapes. [26, 75, 182] proposed a rotation-equivariant network with newly designed spherical convolutional operators. However, the voxelization of 3D geometry induces a trade-off between resolution and computational cost. The pioneering work PointNet used a spatial transformation network to learn an affine transformation, which still did not fulfill the requirement. Inspired by CNNs, which use different receptive fields to aggregate the local features, DGCNN used a dynamic k-nearest neighbors algorithm (KNN) to exploit local information. However, its classification results still suffer a sharp drop in rotation experiments. For alleviating the issue, I introduce two different rotation-invariant features (RIF). The first one is spherical harmonics [69], which transform the Cartesian pose to the spectral domain by using a non-commutative Fourier analysis methods and are related to the power spectrum in the perspective of signal processing. The second feature can locally describe the geometry relationship by creating a Darboux frame at each object point with a KNN-graph. This geometry point feature is also utilized in the point feature histogram [150] and fast point feature histogram [148]. The rotation-invariant feature aims at separating the rotated point cloud and the network so that the input space is invariant to arbitrary rotation perturbation. Furthermore, I design a new network structure that can hierarchically extract the local features by applying the farthest point sampling strategy. The proposed network structure is composed of RIMapping, PF Abstraction, and Classification blocks. In the RIMapping block, rotation-invariant features are fed to a feature transformation network, which maps the lower level feature to a high level embedding space. Two consequent abstraction layers work on these high-level embedding features. For further exploiting the local geometry information, a fully connected point feature graph is built on each cluster and the resultant features are fed to a point feature transformation. Afterward, a global abstraction layer can aggregate all previous embedding features together to obtain a global feature. The Classification block is a standard fully connected network to classify the objects. I evaluated my proposed network on ModelNet40 with different experimental settings and achieve or exceed most state-of-the-art approaches.

My primary contributions are two-fold: a) I introduce a novel geometry rotation-invariant feature descriptor, which can globally and locally represent a 3D shape. b) a new rotation-invariant classification network structure is designed, which can efficiently exploit local geometric features.

5.2 Related Work

With recent good results from deep learning in image-based recognition, 3D visual recognition has also received more attention and rapid development. It benefits from deep learning

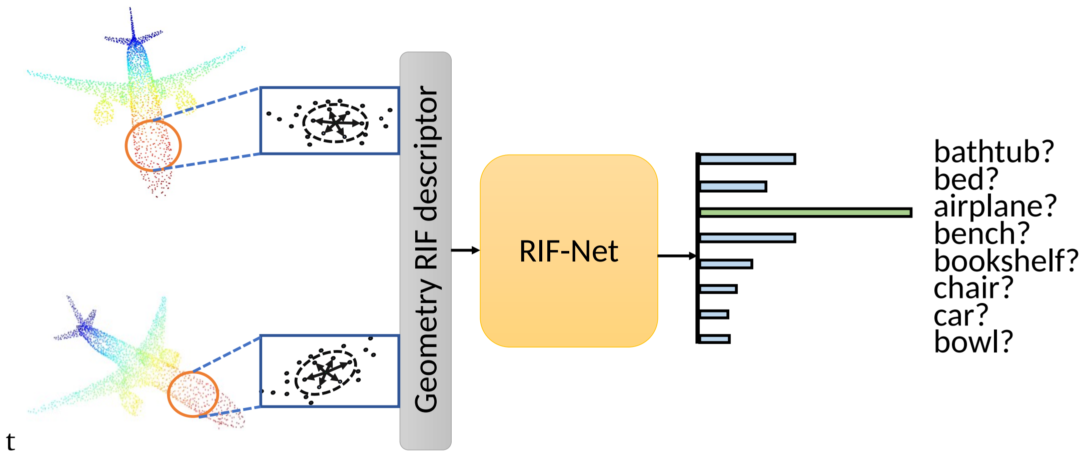


Figure 5.1: The structure of the RIF-Net: The geometry RIF features compose of globally spherical harmonics and locally simplified point feature rotation-invariant features, which are fed to my designed structure. The rotation invariant feature separates the raw point cloud from the network. As a result, the RIF-net can classify the object regardless of the rotation transformation in the world space

in extracting and learning geometric features more efficiently, but the recognition of 3D geometry differs from image-based recognition in many factors. One main aspect are the representation formats, where 3D geometry uses various methods such as point cloud-based representation, implicit surfaces based representation, or volumetric-based representations. These different formats lead to different learning methods. In contrast, the imaged-based representation is interpreted in regular data, where the conventional CNN is designed to handle such regular data. The permutation $N!$ is a common problem in the irregular data format. Based on these observations, previous work seeks to utilize benefits from conventional CNNs by voxelizing the 3D geometric shape [105, 135, 191, 198] or by using multi-view images [67, 135, 165]. However, the trade-off between the resolution and computational cost makes generalization impossible. Most 3D convolutional neural networks sacrifice high resolution to obtain fast calculations to build upon a shallow network. To alleviate the negative impact of accuracy due to resolution, an Octet [143] is proposed by hierarchically partitioning the space using a set of unbalanced octrees to exploit sparse input data.

In contrast to a volumetric representation, PointNet [134] is the first work that directly feeds the point cloud into a set of shared MLP networks and uses the max pool operator to extract global features. It shows a significant improvement in the perspective of 3D shape reasoning and computational cost. PointNet, however, does not extract local information. Follow-up work such as PointNet++ [136] progressively aggregated local features using the farthest point sampling strategy. Moreover, DGCNN [177] introduced a dynamic KNN to build a local graph and aggregated the edge features to obtain a better feature representation. A point-based neural network satisfies many properties, e.g., permutation invariance with a shared MLP and max pool operator and translation equivariance with a relu operator [75]. This network is shown to solve many classical problems such as classification, part segmentation, and instance segmentation. The rotation-invariant property is however still missing in the designed structures. PointNet applies a spatial transformer network [60] to predict an affine transformation matrix. Other work attempts to augment the data set by generating a lot of SO(3) combinations. However, SO(3) is infinite, and data augmentation wastes computational resources and cannot guarantee effectiveness. To alleviate this issue, previous work proposed a rotation-equivalence network structure. [26, 39, 75] designed a spherical-based convolutional operator utilizing the properties of spherical harmonics. [170] proposed tensor field networks, which map point clouds to point clouds under the constraint of SE(3) equivariance by utilizing a spherical harmonics filter. Spherical representations for 3D data are not novel and have been used for retrieval tasks before the deep learning era [68, 69].

Spherical-based CNNs were initially designed for voxelized shapes and suffered a loss of geometric information, as there is no bijection between $\mathbb{R}^3$ and 2-dimensional sphere S^2 [23] as mentioned above. Instead of proposing a new convolutional operator, [23] introduced rigorously rotation-invariant (RRI) features by transforming the point from Cartesian space into an embedding space and showed a good improvement in experiments. However, the RRI features focus only on the local feature using the same dynamic KNN as DGCNN.

5.3 Geometric Rotation Invariant Feature Descriptor

The concepts of Rotation invariance and Rotation equivariance are susceptible to ambiguity. To remove the ambiguity, the definition of these two concepts are formulated.

Definition 1: Rotation invariance A mapping function $\phi(\cdot)$: $\mathcal{V} \rightarrow \mathcal{V}$ is considered as rotation invariant, if $\forall g_i \in \mathrm{SO}(3)$ and $\forall \mathbf{q} \in \mathbb{R}^3$, it has property

$$\phi(g_1\mathbf{q}) = \phi(g_2\mathbf{q}), \tag{5.1}$$

Definition 2: Rotation equivariance A mapping function $\phi(\cdot)$: $\mathcal{V} \rightarrow \mathcal{V}$ is considered as rotation equivariance, if $\forall g_i \in \mathrm{SO}(3)$ and $\forall \mathbf{q} \in \mathbb{R}^3$, it has property

$$\phi(g_1\mathbf{q}) = g_1\phi(\mathbf{q}), \tag{5.2}$$

Pioneering works in processing point clouds are PointNet and DGCNN, where EdgeConv from DGCNN and mini-PointNet from PointNet++ utilize the edge feature represented as an implicit geometry feature by considering geometric constraints between points. It can be mathematically formulated as

$$\mathbf{x}_i = \max_{j:(i,k)\in\Xi} \mathrm{f}_\Theta(\mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i) \tag{5.3}$$

where $\mathbf{x}_i$ is the Cartesian pose in pointNet++ and is high-level features in the DGCNN. In general, the geometric point descriptor module helps to enhance the scattered point clouds using additional internal geometric relations. The edge features $\mathbf{x}_i - \mathbf{x}_j$ and pose point $\mathbf{x}_i$ do not satisfy the property described in (5.1). Furthermore, edge features under a dynamic KNN can only represent the local geometric context for point clouds in the embedding space. For alleviating this issue, two rotation-invariant descriptors will be introduced, that globally (spherical harmonics descriptor) and locally (point feature descriptor) represent the geometry shape.

5.3.1 Rotation invariant spherical harmonics descriptor

Definition: Spherical harmonics define an orthonormal basis over the sphere, with the parameterization

$$(x, y, z) = (\sin(\theta)\cos(\varphi), \sin(\theta)\sin(\varphi), \cos(\theta)), \tag{5.4}$$

where (x, y, z) is a location defined on a unit sphere with co-latitude θ and longitude φ and the orthonormal basis function given by Rodrigues' formula can be described as

$$Y_l^m(\theta, \varphi) = K_l^m P_l^m(\cos\theta)e^{im\varphi}, \tag{5.5}$$

with the normalized constant variable K_l^m with

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}} \tag{5.6}$$

and the associated Legendre polynomials P_l^m. The parameters l and m are the spherical harmonic degree and order, respectively. Furthermore, the order should satisfy the constraint $-l \le m \le l$. The orthogonality property of spherical harmonics is shown as

$$\int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} Y_l^m Y_{l'}^{m'*} \sin\theta d\varphi d\theta = \delta_{ll'}\delta_{mm'} \tag{5.7}$$

It can be shown that all the above normalized spherical harmonic functions satisfy

$$Y_l^{m*}(\theta,\varphi) = (-1)^m Y_l^{-m}(\theta,\varphi). \tag{5.8}$$

where the superscript $*$ denotes complex conjugation. The real spherical harmonics are sometimes known as tesseral spherical harmonics. These functions have the same orthonormality properties as the complex ones above. The harmonics with $m > 0$ are said to be of cosine type and those with $m < 0$ of sine type. The reason for this can be seen by writing the functions in terms of the Legendre polynomials P_l^m with Condon-Shortley phase convention as

$$Y_{lm} = \begin{cases} (-1)^m \sqrt{2} K_{|m|}^l P_l^{|m|}(\cos\theta)\sin(|m|\varphi) & \text{if } m < 0 \\ \sqrt{\frac{2l+1}{4\pi}} P_l^m(\cos\theta) & \text{if } m = 0 \\ (-1)^m \sqrt{2} K_m^l P_l^m(\cos\theta)\cos(m\varphi) & \text{if } m > 0 \end{cases} \tag{5.9}$$

Moreover, for any rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$, the rotated SH $Y_l^m(R\cdot)$ can be expressed as a linear combination of other SHs of the same degree l

$$Y_l^m(\mathbf{R}\cdot) = \sum_{m'=-l}^{l} \left[D_{\mathbf{R}}^{(l)}[m,m']\right]^* Y_l^{m'}, \tag{5.10}$$

where $D_{\mathbf{R}}^{(l)}[m,m'] \in \mathbb{C}^{(2l+1)\times(2l+1)}$ is called the Wigner D-matrix. Note that the Wigner matrices $\mathbf{D}^l$ are all orthonormal and irreducible representations of SO(3) [48], which considers them as *smallest* representations possible. The elements for the $(2\ell+1)\times(2\ell+1)$ matrix D^ℓ of degree ℓ are

$$D_{m,n}^{\ell}(\alpha,\beta,\gamma) = e^{im\alpha} d_{m,n}^{\ell}(\beta) e^{in\gamma}, \tag{5.11}$$

where α,β,γ are ZYZ Euler angles representing the rotation, $-\ell \le m,n \le \ell$, and $d_{m,n}^\ell(\beta)$ is real and proportional to the Jacobi polynomial $P_k^{(a,b)}(\cos\beta)$, with k,a,b being functions of ℓ,m,n. In accordance with the unitary of $D_{\mathbf{R}}^{(l)}$, the energy within a subspace is preserved. Therefore, for any given vector $\mathbf{c} \in \mathbb{C}^{2l+1}$, the Wigner D-matrix shows a norm preservation property [2, 140] as $\|D_{\mathbf{R}}^{(l)}\mathbf{c}\| = \|\mathbf{c}\|$.

The theory of spherical harmonics says that any spherical function f(θ,φ) is decomposed as the sum of its harmonics:

$$f(\theta,\varphi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{m=l} a_{lm} Y_l^m(\theta,\varphi), \tag{5.12}$$

The associated Legendre polynomials is defined associated Legendre polynomials

$$P_l^m(x) = (-1)^m 2^l \left(1-x^2\right)^{m/2} \sum_{k=m}^{l} \frac{k!}{(k-m)!} x^{k-m} \binom{l}{k} \binom{\frac{l+k-1}{2}}{k} \tag{5.13a}$$

$$P_l^{-m} = (-1)^m \frac{(l-m)!}{(l+m)!} P_l^m \tag{5.13b}$$

The coefficient a_l^m is described as

$$a_l^m = \int_\Omega f(\theta,\varphi) Y_l^{m*}(\theta,\varphi) d\Omega = \int_0^{2\pi} d\varphi \int_0^{\pi} d\theta \sin\theta f(\theta,\varphi) Y_l^{m*}(\theta,\varphi)$$

Eqn. (5.12) can be seen as a kind of Fourier series on the sphere.

Information loss for a limited bandwidth

Since I cannot solve $l \rightarrow \infty$, I limit the band l to a constant degree $n_{\text{sh,deg}}$. The spherical function can be simplified as

$$f(\theta,\varphi) = \sum_{l=0}^{n_{\text{sh,deg}}} \sum_{m=-l}^{m=l} a_{lm} Y_l^m(\theta,\varphi) \tag{5.14}$$

The information loss is defined as

$$\text{Loss} = \left\| \sum_{l=0}^{n_{\text{sh,deg}}} f_l - \sum_{l=0}^{\infty} f_l \right\|_2 . \tag{5.15}$$

Furthermore, the numerical solution of coefficients a_l^m can be approximated by using the Monte Carlo integration approach.

$$a_l^m = \frac{4\pi}{n_{\text{sh,deg}}} \sum_{j=0}^{n_{\text{sh,deg}}} f_j(\theta_m, \phi_m) Y_{l,j}^m(\theta_m, \phi_m) \tag{5.16}$$

Spherical Harmonics energy descriptor

Polygonal-based surface representations are typically described as Cartesian coordinates (x, y, z). For spherical harmonics, the surfaces are represented by $\mathrm{f}(\theta,\phi)$, therefore the mesh must be transformed into spherical polar coordinates (r,θ,ϕ) about the origin. In this case I define $\mathrm{f}(\theta,\phi) = r$ [120] with the energy spectrum descriptor of spherical harmonics [69]

$$\mathcal{X}_{sh}(f) = \{\|f_0(\theta,\varphi)\|, \|f_1(\theta,\varphi)\|, \ldots\} \tag{5.17}$$

with the frequency components

$$f_l = \left[a_{l,-l} Y_l^{-l}, a_{l,-l+1} Y_l^{-l+1}. \cdots, a_{l,l} Y_l^l\right]. \tag{5.18}$$

Utilizing the norm preservation property of Wigner D-matrices [39, 69, 140], I can prove that $\|f_l\|$ is a rotation-invariant descriptor.

5.3.2 Rotation invariant point feature descriptors

I employ point feature representations to encode the neighborhood's geometrical properties, which provides an overall point density and pose invariant multi-value feature. The surface normal [147] is estimated by using PCA on the k-neighborhood. For each pair $\mathbf{p}_s$ and $\mathbf{q}_t$ with $\mathbf{q}_t \in \mathcal{N}(\mathbf{p}_s)$, Darboux frame at $\langle \mathbf{p}_s, \mathbf{n}_s \rangle$ is defined as

$$\mathbf{u} = \mathbf{n}_s, \quad \mathbf{v} = \frac{(\mathbf{p}_t - \mathbf{p}_s)}{\|\mathbf{p}_t - \mathbf{p}_s\|_2} \times \mathbf{u}, \quad \mathbf{w} = \mathbf{u} \times \mathbf{v}. \tag{5.19}$$

The point features descriptor [149] is described as a quadruplet $\langle \alpha, \phi, \theta, d_{\text{st}} \rangle$ with

$$\begin{aligned} d_{\text{st}} &= \|\mathbf{p}_t - \mathbf{p}_s\|_2, & \alpha &= \mathbf{v} \cdot \mathbf{n}_t, \\ \phi &= \mathbf{u} \cdot \frac{(\mathbf{p}_t - \mathbf{p}_s)}{d_{st}}, & \theta &= \operatorname{atan2}(\mathbf{w} \cdot \mathbf{n}_t, \mathbf{u} \cdot \mathbf{n}_t). \end{aligned} \tag{5.20}$$

Furthermore, I augment the distance $r_s = \left\| \boldsymbol{p}_s \right\|_2$ to the point feature, therefore, I get a quintuple feature descriptor.

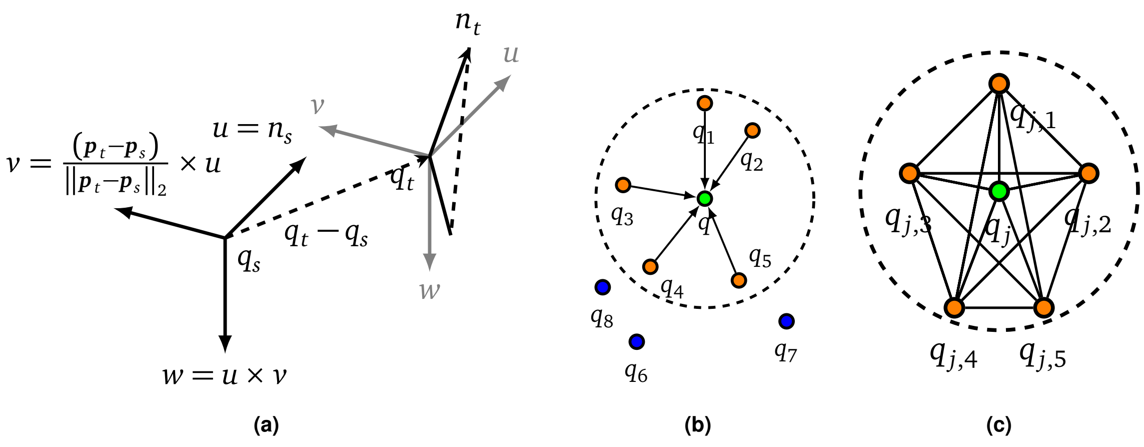


Figure 5.2: a illustrates the Darboux frame between a point pair. b calculates the simplified point features of a source point $\mathbf{q}$ with a KNN graph for each point pair. A fully connect point feature (PF) graph c is built on a given source point $\mathbf{q}_i$.

Proof of rotation-invariant property

Given a point set $S = \{\mathbf{p}_i \mid p_i \in \mathbb{R}^n\}_{i=0}^{N-1}$. It is obvious that the L^2 norm is a rotation-invariant operator $\mathbb{R}^n$ to $\mathbb{R}$ due to norm preservation: $\|\mathbf{Rx}\|_2^2 = \|\mathbf{x}\|_2^2$. It can be easily extended that the inner product $\langle\cdot,\cdot\rangle$ between two arbitrary points preserves the rotation-invariant property. In addition, the cross product has the property $R\mathbf{a} \times R\mathbf{b} = R(\mathbf{a} \times \mathbf{b})$ under proper rotations R. I define the Darboux frame at $\mathbf{q}_i$ as a triple tuple: $\mathcal{O}_i = \langle \mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i \rangle$. By applying a rotation matrix to the point set, I can get $\mathbf{q}_j = \mathbf{R}\mathbf{q}_i$ with the corresponding Darboux frame $\mathcal{O}_j = \langle \mathbf{u}_j, \mathbf{v}_j, \mathbf{w}_j \rangle$. It leads to

$$\mathbf{u}_j = \mathbf{R}\mathbf{u}_i, \tag{5.21a}$$

$$\mathbf{v}_j = \mathbf{R}\frac{(\mathbf{p}_t - \mathbf{p}_s)}{\|\mathbf{p}_t - \mathbf{p}_s\|_2} \times \mathbf{R}\mathbf{u}_i = \mathbf{R}\mathbf{v}_i, \tag{5.21b}$$

$$\mathbf{w}_j = \mathbf{u}_j \times \mathbf{v}_j = \mathbf{R}(\mathbf{u}_i \times \mathbf{v}_i) = \mathbf{R}\mathbf{w}_i \tag{5.21c}$$

Based on these relation, I can conclude that $\mathcal{O}_j = \mathbf{R}\mathcal{O}_i = \langle \mathbf{R}\mathbf{u}_i, \mathbf{R}\mathbf{v}_i, \mathbf{R}\mathbf{w}_i \rangle$. As a result, the PF descriptor is proven to be rotation-invariant:

$$d_{st,j} = \|\mathbf{p}_{t,j} - \mathbf{p}_{s,j}\|_2 = \|\mathbf{R}\mathbf{p}_{t,i} - \mathbf{R}\mathbf{p}_{s,i}\|_2 = d_{st,i} = d_{st} \tag{5.22a}$$

$$\alpha_j = \langle \mathbf{v}_j, \mathbf{n}_{t,j} \rangle = \langle \mathbf{R}\mathbf{v}_i, \mathbf{R}\mathbf{n}_{t,i} \rangle = \langle \mathbf{v}_i, \mathbf{n}_{t,i} \rangle = \alpha_i \tag{5.22b}$$

$$\phi_j = \left\langle \mathbf{u}_j, \frac{(\mathbf{p}_{t,j} - \mathbf{p}_{s,j})}{d_{st}} \right\rangle = \left\langle \mathbf{R}\mathbf{u}_i, \mathbf{R}\frac{(\mathbf{p}_{t,i} - \mathbf{p}_{s,i})}{d_{st}} \right\rangle = \phi_i \tag{5.22c}$$

$$\begin{aligned}\theta_j &= \operatorname{atan2}\left(\langle \mathbf{w}_j, \mathbf{n}_{t,j} \rangle, \langle \mathbf{u}_j, \mathbf{n}_{t,j} \rangle\right) \\ &= \operatorname{atan2}\left(\langle \mathbf{R}\mathbf{w}_i, \mathbf{R}\mathbf{n}_{t,i} \rangle, \langle \mathbf{R}\mathbf{u}_i, \mathbf{R}\mathbf{n}_{t,i} \rangle\right) = \theta_i\end{aligned} \tag{5.22d}$$

5.3.3 Geometry rotation invariant feature-descriptor

The SH-energy and PF descriptors are shown to be rotation-invariant descriptors. The SH-energy descriptor focuses on capturing the global features of the 3D shape and the PF descriptor aims at describing the local features. It is straightforward to concatenate both descriptors

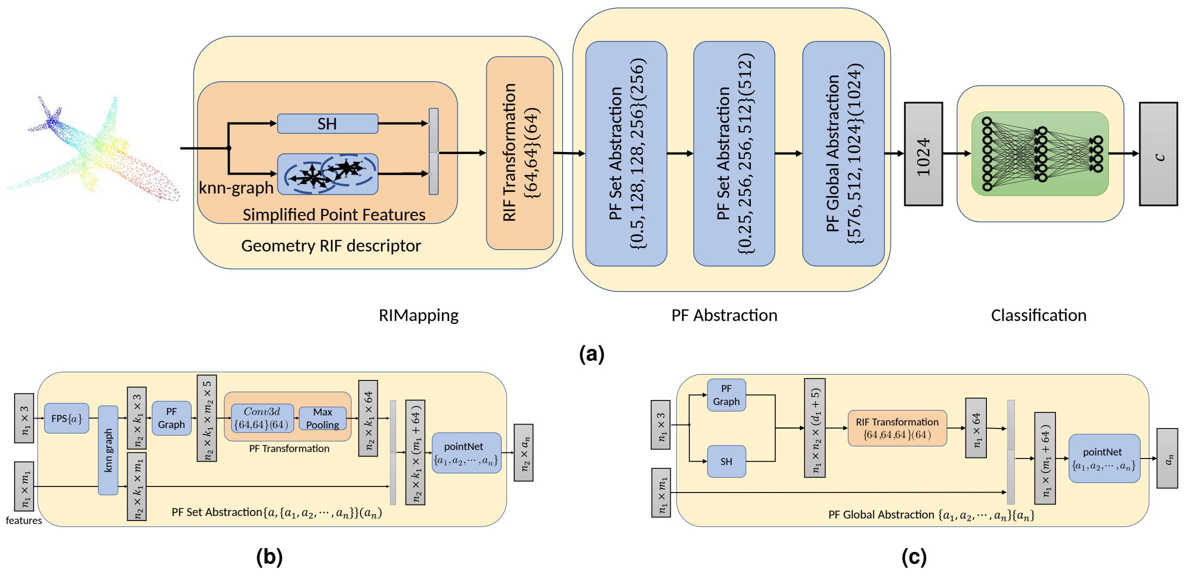

SH
knn-graph
Simplified Point Features
Geometry RIF descriptor
RIF Transformation {64,64}(64)
PF Set Abstraction {0.5, 128, 128, 256}(256)
PF Set Abstraction {0.25, 256, 256, 512}(512)
PF Global Abstraction {576, 512, 1024}(1024)
1024
c
RIMapping
PF Abstraction
Classification
(a)
$n_1 \times 3$
FPS$\{a\}$
knn graph
$n_2 \times k_1 \times 3$
PF Graph
$n_2 \times k_1 \times m_2 \times 5$
$Conv3d$ $\{64,64\}(64)$
Max Pooling
PF Transformation
$n_2 \times k_1 \times 64$
$n_1 \times m_1$
features
$n_2 \times k_1 \times m_1$
$n_2 \times k_1 \times (m_1 + 64)$
pointNet $\{a_1, a_2, \cdots, a_n\}$
$n_2 \times a_n$
PF Set Abstraction$\{a, \{a_1, a_2, \cdots, a_n\}\}(a_n)$
(b)
$n_1 \times 3$
PF Graph
SH
$n_1 \times n_2 \times (d_1 + 5)$
RIF Transformation $\{64,64,64\}(64)$
$n_1 \times 64$
$n_1 \times m_1$
$n_1 \times (m_1 + 64)$
pointNet $\{a_1, a_2, \cdots, a_n\}$
a_n
PF Global Abstraction $\{a_1, a_2, \cdots, a_n\}\{a_n\}$
(c)

Figure 5.3: The model architecture a consists of the RIMapping, PF Abstraction, and Classification blocks. In the RIMapping block, a spherical harmonics energy descriptor and simplified point features (SPF) at each point are computed to form a Geometry RIF descriptor, which is fed to a RIF Transformation to extract a high-level feature. In the PF Abstraction Block, I have two PF Set Abstraction layers b together with a PF Global Abstraction layer c to obtain a final global feature, which is used in a fully-connected Classification network.

and this results in the rotation-invariant feature (RIF)-descriptor at $\mathbf{q}_i$

$$\mathcal{X}_{rif,i} = \left[(\mathcal{X}_{sh,i}, \mathcal{X}_{pf,i,0}), \quad \ldots, \quad (\mathcal{X}_{sh,i}, \mathcal{X}_{pf,i,k})\right]^{\mathrm{T}}, \tag{5.23}$$

where $\mathcal{X}_{rif,i} \in \mathbb{R}^{k\times(n_{\mathrm{sh}}+n_{\mathrm{pf}})}$ with point feature descriptor

$$\mathcal{X}_{pf,i,j} = [d_{i,j}, \alpha_{i,j}, \phi_{i,j}, \theta_{i,j}, r_{s,i}] \in \mathbb{R}^{n_{\mathrm{pf}}}, \tag{5.24}$$

with $n_{\mathrm{pf}} = 5$ and the spherical harmonics energy descriptor

$$\mathcal{X}_{sh,i} = \left[\|f_{0,i}\|, , \ldots, \|f_{n_{\mathrm{sh,deg}},i}\|\right] \in \mathbb{R}^{n_{\mathrm{sh}}}, n_{\mathrm{sh}} = n_{\mathrm{sh,deg}} + 1\,. \tag{5.25}$$

The advantage of this formulation embodies that I can extract a high layer feature at each point pair under KNN-Graph.

5.4 Network Architecture

The proposed rotation-invariant network consists of three blocks: RIMapping, PF Abstraction, and Classification, where the latter is a feed-forward network. My main contributions are on the design of the RIMapping and PF Abstraction blocks.

5.4.1 Pre-Alignment with Principal Component Analysis

In an experiment on rotation invariance, I transfer a point cloud set $\mathbf{q}$ with an arbitrary rotation matrix $\mathbf{R}$, resulting in a rotated clone $\mathbf{q}^1 = \mathbf{R}\mathbf{q}$. According to [68], I pre-align each input $\mathbf{q}$ based on the PCA to its principle axes, which are indicated as orthonormal coordinates $\mathbf{R}_0$, with the formula of $\mathbf{q}_1 = \mathbf{R}_0^{\mathrm{T}}\mathbf{q}$. It can be shown that the PCA alignment for a rotated clone $\mathbf{q}^1$ with corresponding orthonormal coordinates $\mathbf{R}_1 = \mathbf{R}\mathbf{R}_0$ leads to $\mathbf{q}_2 = \mathbf{R}_1^{\mathrm{T}}\mathbf{q}^1 = \mathbf{R}_0^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{q} = \mathbf{R}_0^{\mathrm{T}}\mathbf{q} = \mathbf{q}_1$. Therefore, pre-alignment can reduce the impact of the rotation matrix on the network.

5.4.2 RIMapping Block

The RIMapping block consists of the Geometry RIF descriptor and RIF Transformation. For acquiring the rotation-invariant features, I leverage the spherical harmonics and point feature representation with a KNN to enrich geometric features for the point cloud, which is represented as a rotation-invariant descriptor with a size of $\mathbb{R}^{n \times k_1 \times (n_{\mathrm{sh}} + n_{\mathrm{pf}})}$ and k_1-neighborhood that can globally and locally manifest a 3D shape. This descriptor provides low-level geometric clues for high-level geometric feature learning, realized with a RIF Transformation. The RIF Transformation layer utilizes a mini PointNet (without input and feature transformation), consisting of a set of shared Conv2d layers with kernel size equal to 1, to extract a global feature employing a max-pooling operator. The output of RIF Transformation is indicated as embedding feature with a size of $\mathbb{R}^{n \times a_n}$. The PF Transformation has the same structure as the RIF Transformation, apart from the different input sizes. Both Transformations intend to aggregate the local details by calculating a weighted average of neighboring features through a shared local fully-connected layer.

5.4.3 PF Abstraction Block

PF Set Abstraction

The extracted information from the RIMapping block is still insufficient for the precise classification task, as max-pooling can only describe an outline and some local details could be omitted. To address the problem, I propose the PF Set Abstraction Layer to hierarchically exploit the local features, which consists of the sampling layer, grouping layer, and PointNet layer. PointNet++ inspires PF Set Abstraction. However, there are several significant adaptations inside the grouping layer. In the sampling layer, I use iterative farthest point sampling (FPS) to obtain n_2 points, indicated as $\mathbf{P}_i, i \in [0, \cdots, n_2]$. Each point $\mathbf{P}_i$ is the center of a local region $\mathcal{C}_i$. In the sequence, a KNN graph is built at point $\mathbf{P}_i$ to obtain k_1-neighborhood, indicated as $[\mathbf{P}_{0,i}, \mathbf{P}_{1,i}, \cdots, \mathbf{P}_{k_1,i}]$, with $i \in [0, \cdots, n_2]$. In contrast to PointNet++, which combines the point with the feature from the last layer and works as input for the PointNet layer, I utilize a PF graph to convert a point to a rotation-invariant feature, where PF graph is a fully connected graph (Fig. 5.2c) and built at each local region $\mathcal{C}_i$. Then, I can get a feature with a size of $\mathbb{R}^{k_1 - 1 \times n_{\mathrm{pf}}}$ for each point $\mathbf{P}_{n_k,i}$ in the region $\mathcal{C}_i$. In the end, a new rotation-invariant point feature for all local regions is obtained, indicated as $\mathbf{f}_{PF} \in \mathbb{R}^{n_2 \times k_1 \times k_1 - 1 \times n_{\mathrm{pf}}}$. Sequentially, I apply the PF Transformation to extract a feature for each local region. I concatenate the previous feature at each center point $\mathbf{P}_i$ with the newly extracted feature to form a new feature representation and feed it to the PointNet layer.

PF Global Abstraction

The global abstraction is the successor layer to the PF Set Abstraction layer, which reduces the original input cloud to $\mathbf{X}_2 \in \mathbb{R}^{n_1 \times 3}$. I build a PF graph at the reduced point set and concatenate it with its spherical energy descriptor, which leads to a new representation with a size of $\mathbb{R}^{n_1 \times (n_1 - 1) \times (n_{\mathrm{pf}} + n_{\mathrm{sh}})}$. In the sequence, I apply the RIF Transformation for extracting a new feature representation. This new feature will concatenate with the feature from the PF Set Abstraction layer. In the end, a mini PointNet is applied to obtain a global feature.

Table 5.1: Comparison of rotation robustness on rotation-augmented ModelNet40 benchmark. My proposed network shows the best performance in the settings z/SO(3) and SO(3)/SO(3). Note, values are given as a percentage.

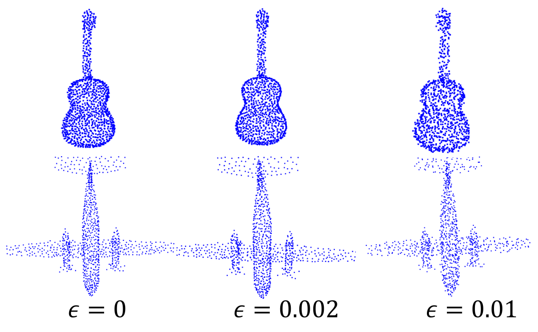
Method	Input(size)	z/z	z/SO(3)	SO(3)/SO(3)
PointNet [134]	pc (1024×3)	89.2	14.7	83.6
PointNet++ [136]	pc (1024×3)	89.3	28.6	85.0
VoxNet [105]	voxel (30^3)	83.0	-	73.0
RotationNet 20x [67]	voxel (20×224^2)	92.4	20.0	80.0
SO-Net [89]	pc+normal (5000×6)	**92.6**	21.1	80.2
DGCNN [177]	pc (1024×3)	92.2	33.5	81.1
Spherical CNN [39]	voxel 2×64^2	88.9	78.6	86.9
ClusterNet [23]	pc (1024×3)	87.1	87.1	87.1
ours($n_{\text{sh,deg}}$=20)	pc(1024×3)	88.4	88.6	88.5
ours($n_{\text{sh,deg}}$=30)	pc(1024×3)	88.6	**88.7**	**88.8**

5.5 Experiments

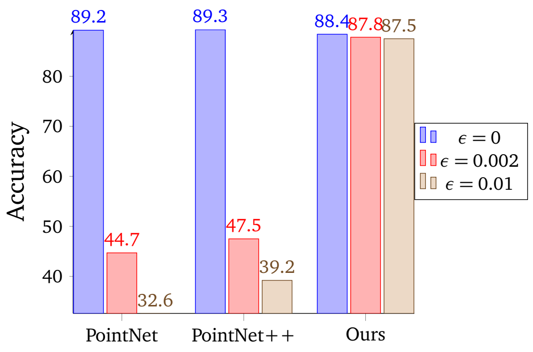
I evaluate my approach regarding rotation robustness and compare it with other state-of-the-art methods. I use ModelNet40 [183] as data set for validating the effectiveness of the proposed network structure. ModelNet40 consists of 40 categories in the form of CAD models (mostly human-made). I use the official split with 9843.000 shapes for training and 2468.000 for testing. I apply the farthest point sampling algorithm to obtain 1024.000 points on mesh faces according to the face area and then shift and normalize the point into a unit sphere with centroid on the origin. During training, I use Adam [71] for 200 epochs with an initial learning rate of 10^{-3}. The algorithm is implemented with PyTorch on Linux with one GeForce RTX 2080Ti GPU.

5.5.1 Evaluation of rotation robustness

For evaluating the property of rotation robustness, I multiply each point cloud from ModelNet40 with a randomly sampled rotation matrix. Based on the same principle as [39], I evaluate my model using three different settings: a) training and testing with azimuthal rotations (z/z), b) training and testing with arbitrary rotations (SO(3)/SO(3)), c) training with azimuth rotations while testing with arbitrary rotations (z/SO(3)). The results are presented in Table 5.1. It can be seen that most networks exhibit a sharp drop in performance in the settings SO(3)/SO(3) and z/SO(3), in particular in the latter one. The network DGCNN [177] shows an outstanding performance in the setting z/z with an accuracy of 92.200 %, while it only achieves 21.100 % in z/SO(3). DGCNN applies the point directly for classification, which changes dramatically when applying a rotation matrix in SO(3). As mentioned before, that point cloud is not a rotation-invariant representation. This is also a common problem for the PointNet-based network. The spherical CNN [39] uses a spherical harmonics-based convolution layer by rasterizing the point cloud, which shows a significant improvement in the setting of z/SO(3). However, the difference between its best performance z/z and z/SO(3) is still significant with a value of 10.300 %. ClusterNet [23] uses the RRI representation together with a cluster abstraction to increase classification performance with a result of 87.100 % in each setting. Note that the result of ClusterNet is directly cited from [23], as the code is not available as open source. Table 5.1 demonstrates that my approach achieves the best



(a)



(b)

Figure 5.4: (a) shows the point cloud with different noise levels and (b) shows the comparison results against point perturbations on rotation setting z/z.

performance in the settings $z/\mathrm{SO}(3)$ and $\mathrm{SO}(3)/\mathrm{SO}(3)$ with $n_{\mathrm{sh,deg}} = 20, 30$. The difference in the results between each setting is very small, approximately 0.200 %. Based on these observations, I can conclude that my proposed network shows the best performance regarding rotation robustness.

5.5.2 Robustness Tests

Evaluation of model against point cloud perturbation

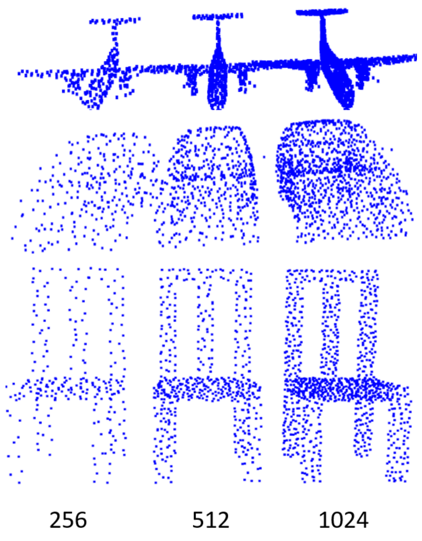
For further evaluation of robustness against perturbation, I conducted experiments by adding perturbation at each point. Many studies have shown that deep learning-based networks can be fooled by using an adversarial attack. Following the same principle, I add a perturbation value to each point with $||\delta|| < \epsilon$, where ϵ is set between 0.002 and 0.01, as shown in Fig. 5.4a. The results are listed in Fig. 5.4b. It can be seen that in these two different perturbation levels, my network with $n_{\mathrm{sh,deg}} = 20$ is more robust under perturbation when compared to PointNet and PointNet++.

Evaluation of model against point cloud density

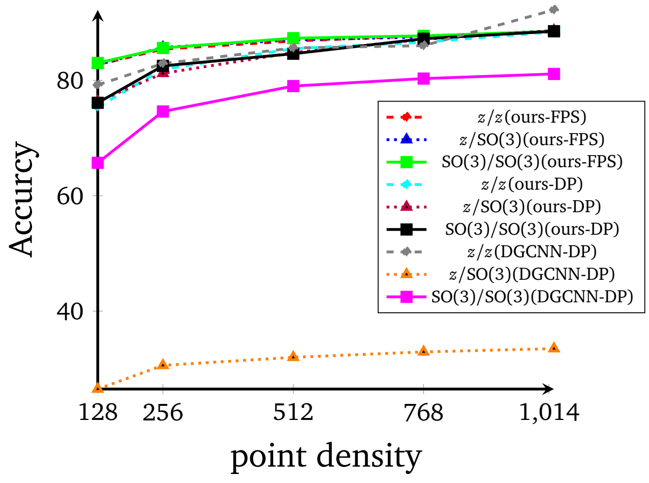
The point cloud density also plays an important role in the classification task. In this section, I downsample ModelNet40 to different point densities in the range of 1024.000 to 128 by using the farthest point sampling strategy (FPS) or random input droupout (DP). The downsampled point clouds are shown in Fig. 5.5a and the corresponding classification accuracy is illustrated in Fig. 5.5b. It is worth noting that my proposed network is very robust against point density changes in these three rotation setups, which decreases the classification accuracy from 88.600 % to 82.700 % by FPS and varies from 88.600 % to 75.400 % by DP. In comparison to DGCNN [177], the results vary from the 92.200 % to 79.200 % in z/z and from 33.500 % to 26.500 % in $z/\mathrm{SO}(3)$. Under the same point density, my model shows no significant change, which further verifies my model's robustness.

Evaluation of model against partial point cloud

In reality, I can only get a partial point cloud by using a single stationary camera. To evaluate the partial point cloud classification model, I train my model with a complete point cloud and test against a partial point cloud. The partial point cloud is obtained by first deleting the completed point cloud with a ratio ρ from 0.1 to 0.2 and then using iterative FPS (Fig. 5.6a).

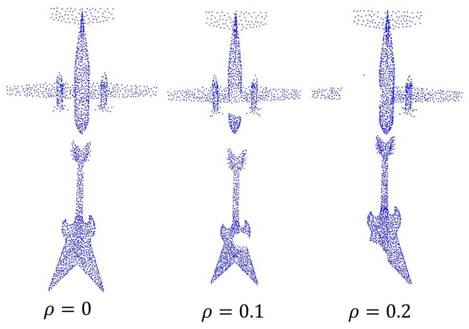


(a)

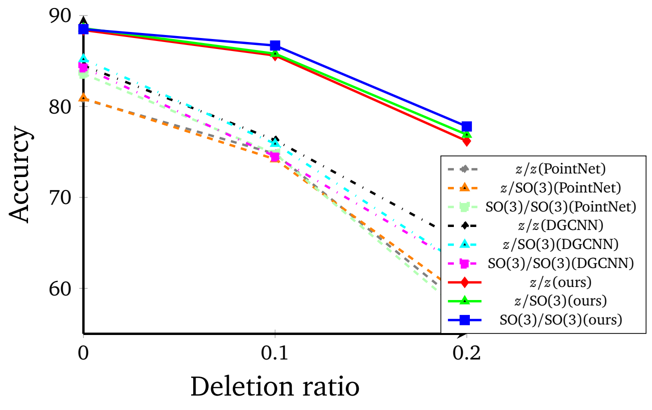


(b)

Figure 5.5: a shows the downsampled point cloud and b illustrates the comparison results of different point densities in three rotation settings with FPS and DP.



(a)



(b)

Figure 5.6: a shows the partial point cloud with the deletion ratio ρ from 0.1 to 0.2 and b shows the results.

The results are illustrated in Fig. 5.6b. I compare my model with PointNet and DGCNN under three rotation settings. Training and testing data set are rotated with a PCA algorithm to reduce the impact of arbitrary rotation. From Fig. 5.6b, I can conclude that my model shows the best performance and far exceeds the other two classification models in all three experiments.

5.5.3 Ablation Studies

Analysis of architecture design

To evaluate my network architecture's effectiveness, I use PointNet as the baseline and connect my individually designed component to it. Note that I realign all data in this section with PCA (Section 5.4.1). In Table 5.2, I can see that it shows a significant improvement when compared to the original PointNet in the setting of z/SO(3) and that the average accuracy

Table 5.2: Effectiveness of designed network block.

Method	z/z	z/SO(3)	SO(3)/SO(3)	mean
PointNet [134]	89.2	14.7	83.6	62.4
PCA+PointNet	80.9	80.9	80.8	80.84
RIMapping($n_{\mathrm{sh,deg}}$=20)+PointNet	82.0	83.2	84.5	83.23
ours(without SH)	85.40	85.7	86.2	85.77
ours($n_{\mathrm{sh,deg}}$=20)	88.4	88.6	88.5	88.5

Table 5.3: Effectiveness of maximum degree $n_{\mathrm{sh,deg}}$.

$n_{\mathrm{sh,deg}}$	z/z	z/SO(3)	SO(3)/SO(3)
8	88.10	87.60	87.60
15	87.80	88.30	87.80
20	88.40	88.60	88.50
30	88.60	88.70	88.80

rate has increased about 18.000 %. Furthermore, I analyzed the effectiveness of the RIMapping block by connecting it to PointNet. The results listed in Table 5.2 show that the accuracy in z/SO(3) and SO(3)/SO(3) improved by 2.300 % and 3.700 %. Comparing my proposed network's worst performance shows that my PF Abstraction block helps in improving the final accuracy in all three settings. I also evaluated the effectiveness of the spherical harmonics energy descriptor. The results are listed in Table 5.2. Without the spherical harmonics energy descriptor, the accuracy is worse when compared against my original design. However, it still shows better performance when compared to the PointNet-based network.

Effectiveness of maximum degree of spherical harmonics

The spherical harmonics descriptor is an essential aspect of my network. Based on the information loss, a higher degree of spherical harmonics leads to a smaller information loss. However, it will also increase the computational complexity to $\mathcal{O}(n^2)$. For evaluating the effectiveness of the $n_{\mathrm{sh,deg}}$, I vary the degree. The results are listed in Table 5.3 and it can be seen that the higher $n_{\mathrm{sh,deg}}$, the better the final classification accuracy.

5.6 Conclusion

I presented a rotation-invariant point cloud-based neural network, which utilizes a global spherical harmonics feature and a local points feature to achieve rotation-invariant properties. Furthermore, a new neural network structure is designed, inspired by PointNet++, but with several adaptations such as PF graph and PF Transformation. The network is applied to 3D object classification, but can be extended to part segmentation and instance segmentation. Via several experiments, I have shown that my network can deal with arbitrary input orientations and achieve competitive performance compared to other state-of-the-art approaches on the ModelNet40 data set. Furthermore, my network shows robustness against point perturbations, point density, and partial point cloud.

Chapter 6

Point Cloud Transformer for Dense Point Cloud Completion

Chapter Summary

This chapter addresses the problem of the point cloud completion problem. Inferring a complete 3D geometry given an incomplete point cloud is essential in many vision and robotics applications. Besides, the irregular nature of point clouds makes this task more challenging. This chapter presents a novel method for shape completion by investigating the point transformer structure. The inherent nature of permutation invariance of the attention mechanism in the Transformer structure makes it well suited for learning point clouds. In addition, the attention mechanism can effectively capture the local context within a point cloud and efficiently exploit its incomplete local structure details. A morphing-atlas-based point generation network further fully utilizes the extracted point Transformer feature to predict the missing region using charts defined on the shape. Shape completion is achieved via the concatenation of all predicting charts on the surface. Extensive experiments on the Completion3D and KITTI data sets demonstrate that the proposed PCTMA-Net outperforms the state-of-the-art shape completion approaches and has a 10% relative improvement over the next best-performing method.

This chapter is a slightly modified version of peer-reviewed conference paper ©2021 IEEE. Reprinted, with permission, from

- **Lin, Jianjie**, Rickert, Markus, Perzylo, Alexander, Knoll, Alois, "PCTMA-Net: Point Cloud Transformer with Morphing Atlas-based Point Generation Network for Dense Point Cloud Completion," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021

The images created, algorithms designed, data from experiments and text written by me in this publication will be directly referenced in this chapter. The original version is referred to 194.

Contributions

I took a leading role in the writing and revising of the manuscript in this article. I have made the following significant personal contributions to the formulation, implementation, and evaluation of the algorithms in this paper: identifying the necessity of point cloud completion in the real scenario, designing the transformer-and manifold-like neural network for recovering the partial point cloud. I am the lead developer of the algorithm implementation and responsible for experimental evaluation.

6.1 Introduction

The use of point clouds as a format of shape representation has increased in the last years due to the rapid development of 3D acquisition technologies such as Lidar and depth cameras. The limited sensor resolution, occlusion, and camera angles however make it challenging to obtain a point cloud representation of the complete shape of an object. As a result, the acquired raw points are typically sparse, noisy, and miss large regions. On the other hand, complete 3D shapes are essential in vision applications, such as semantic segmentation and SLAM [19]. A complete 3D shape can improve the performance of CAD model-based point registration [95] and enables more flexible grasp planning [175]. In this work, I focus on completing partial 3D shapes that suffer from occlusion and limited sensor resolution. Previous work [51, 168, 188] principally followed the encoder-decoder paradigm framework by extracting a latent global feature from an incomplete point cloud. Decoders leverage these feature to predict missing regions. Benefiting from PointNet-based [134] feature extractor networks, the task of shape completion made tremendous progress in recent years. However, the extracted global features from PointNet ignore the geometric relationship within the point clouds due to the max-pooling operation. As a result, these approaches suffer from a loss of structural detail in the reconstruction.

The intuitive solution is to make up for the shortcomings of the PointNet by excavating the semantic affinity within the point cloud. Therefore, I propose a novel framework named Point Cloud Transformer with Morphing-Atlas-based Point Generation Network for Shape Completion (PCTMA-Net) to address this problem. The Transformer [176] is a standard framework for natural language processing and has been further extended to vision tasks for image recognition [35], as well as point cloud classification and segmentation [52]. The Transformer follows the encoder-decoder structure and consists of four main modules: input embedding, positional encoding, (self-)attention mechanism, and positional feed-forward. In this work, I apply only the encoder module and neglect the positional encoding module due to the point cloud's irregular nature. The Transformer's central core is the attention mechanism, which can generate refined attention features by leveraging the global context. The attention weight between any two positions is updated by the dot product of query and key vector. The weighted sum of all attention weights is the attention feature. The concept of query, key, and value vector makes it possible to match and learn the global context. The attention feature of each word is related to all input features. Furthermore, the permutation invariant nature of softmax, dot product, and point-wise feed-forward network makes it well-suited for point cloud learning. The offset attention mechanism introduced in [52] uses the idea of the Laplacian matrix to improve the attention performance further. In this work, I replace the original attention design with the offset attention mechanism. The morphing-atlas-based point generation network is the decoder component in my overall structure. The extracted global feature from the Transformer is further utilized to generate the points. An atlas, as defined in topology, consists of a set of charts on a surface. Therefore, I assume that a missing region of the surface can be recovered by a chart. Based on this assumption, I duplicate the Transformer feature and concatenate it with a predefined grid. I utilize the idea of multi-head attention by linearly projecting the concatenated features to learn n_{chart} different features, where each feature is responsible for generating a chart defined on the surface. I quantitatively and qualitatively evaluated the proposed PCTMA-Net on the Completion3D data set and demonstrate a 10% relative improvement over the next best-performing method for the task of shape completion. Furthermore, the qualitative evaluation on the KITTI data set shows that my proposed network is able to predict more structural details than other state-of-the-art approaches.

My contributions are summarized as follows:

1. I propose a novel shape completion framework named Point Cloud Transformer with Morphing-Atlas-based point generation Network for shape completion, which is inherently permutation-invariant and has the capability of learning the global context within the point clouds and preserving structural details.

2. The integration of the concept of an atlas and the multi-head attention mechanism leads to the generation of high-resolution, high-fidelity, and fine-grained shapes.

3. Extensive experiments are conducted on the Completion3D benchmarks, and the KITTI data set, which indicate that the proposed networks remarkably outperforms other competitive methods.

6.2 Related Work

Shape completion approaches made significant progress in recent years due to the rapid development of deep learning and 3D acquisition technologies. I can roughly categorize the existing work into volumetric-based and multilayer perceptron-based networks from the perspective of network structure and the underlying 3D data representation.

Volumetric-based shape completion: The extension of CNN to 3D convolutional neural networks can be used for dealing with a shape in the volumetric representation [186], [57]. Notable work such as 3D-Encoder-Predictor Networks (3D-EPN) [28] progressively reconstruct the 3D volumetric shape. The work in [55] directly generates the high resolution 3D volumetric shape by combining the global structure with the refinement of local geometry, while [164] introduced a variational auto-encoder to learn a shape prior to inferring the latent representation of complete shapes. GRNet [185] took one step further by introducing Gridding and Gridding Reverse to convert between point clouds and 3D grids. However, a quantization effect is introduced during the transformation of point clouds into a 3D volumetric representation. The computational costs increase cubically to the resolution and therefore make it more challenging to process fine-grained shapes.

Multilayer perceptron (MLP)-based shape completion: Point clouds can be directly obtained by several acquisition techniques. It is much more efficient compared to the voxel-based representation when processing costs are compared. Inspired by PointNet [134] and its successor work [136],[177], several approaches use them for point cloud learning, as the point-wise MLP enables the handling of irregular point clouds and aggregating features using a symmetric function. However, the PointNet network suffers from a loss of structure details. The current state-of-the-art approaches for shape completion such as AtlasNet [51], PCN [193] and Folding-Net [188] use PointNet as their baseline to extract global features and to apply a decoder to predict the missing regions. Unlike PCN and FoldingNet, AtlasNet completes the shape by generating surface elements utilizing the atlas concept. TopNet [168] improves the decoder by using a hierarchical rooted tree. By combining reinforcement learning with an adversarial network, RL-GAN-Net [152] and Render4Completion [57] propose a reinforcement learning agent-controlled GAN to improve the quality and consistency of the generated complete shape. However, most of these studies suffer from information loss on structural details, as they predict the whole point cloud only from a single global shape representation. SA-Net [180] extended these approaches with a skip-attention mechanism to preserve more structural details. PF-Net [59] introduced a point pyramid decoder to generate a shape in different resolution levels.

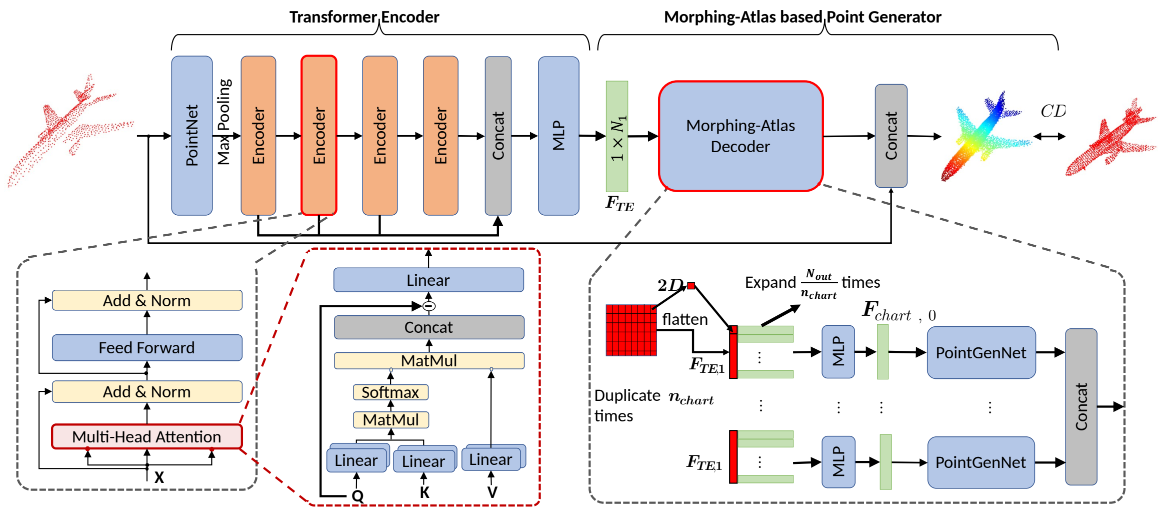


Figure 6.1: The overall structure of PCTMA-Net. The whole structure consists of a Transformer encoder and morphing-atlas point generation decoder. The Transformer encoder aims to extract features from the input point clouds by using an $N\times$ stacked encoder layer which consists of an attention mechanism and positional feed-forward network. The morphing-atlas-based surface reconstruction decoder uses multi-chart point generation networks for point cloud completion by concatenating the features from the Transformer encoder and mesh grid.

6.3 The Architecture of PCTMA-Net

6.3.1 Overview

The overall structure of PCTMA-Net is illustrated in Fig 6.1, which aims to learn a semantic affinity within a partial point cloud by using a Transformer encoder. The complete 3D shape is reconstructed with a morphing-atlas decoder utilizing the extracted feature from the Transformer encoder. I formulate the whole shape completion pipeline as: Given a partial point cloud, indicated as $\mathcal{P}$ with N_{in} points, where each point is represented in 3D coordinates $\mathbf{x} = [x_i, y_i, z_i]$, I first convert this partial point cloud into a feature vector $\mathbf{F}_0$ by a PointNet. The difference to previous work [51, 188], which relies on only the global feature for shape completion, is that I further utilize the Transformer encoder to process the feature to obtain a piece of semantic affinity information for predicting the missing regions. The extracted feature is later fed to the morphing-atlas point generator for completing the shape.

6.3.2 Point Cloud Transformer Encoder

The Transformer encoder of PCTMA-Net first transforms an incomplete point cloud to the feature space using an input embedding network. I then feed the extracted feature to $N\times$ stacked encoder layers, where they share a similar philosophy of design as the original paper [176], except for the attention mechanism. The purpose of the encoder layer is to learn a semantically rich and discriminate representation for each point. The encoder can be mathematically formulated in the following: By given a partial point cloud $\mathcal{P} \in \mathcal{R}^{N_{\text{in}}\times d}$ with N_{in} points each having a d-dimensional feature description, an embedding feature $\mathbf{F}_0$ is firstly learned with an input embedding network, indicated as $\text{F}_{\text{embedding}}$. The difference to the embedding network presented in [52] is that I defined $\text{F}_{\text{embedding}}$ as a PointNet followed by a max-pooling operator. As a result, I acquire a d_{model}-dimensional embedding feature $\mathbf{F}_0 \in \mathcal{R}^{d_{\text{model}}}$ instead of $\mathbf{F}_0 \in \mathcal{R}^{d_{\text{model}} \times N_{\text{in}}}$[52]. It will improve the shape completion performance, as the $\mathbf{F}_0$ after max-pool operator can reduce redundant information and make the training more efficient. The global feature $\mathbf{F}_0$ is later fed to $\text{F}_{\text{encoder}_i}$:

$$\mathbf{F}_i = \text{F}_{\text{encoder}_i}(\mathbf{F}_{i-1}),\ i = [1, \dots, N]. \tag{6.1}$$

Furthermore, I concatenate the features from each encoder layer and follow up by two cascade LBR layers to form an effective global feature

$$\mathbf{F}_{\text{e}} = \text{BatchNorm}(\mathbf{F}_i \oplus \cdots \oplus \mathbf{F}_N) \tag{6.2a}$$

$$\mathbf{F}_{\text{TE}} = \text{LBR}\big(\text{LBR}(\mathbf{F}_{\text{e}})\big), \tag{6.2b}$$

where $\mathbf{F}_{\text{i}} \in \mathcal{R}^{d_{\text{model}}}$, $\mathbf{F}_e \in \mathcal{R}^{N \times d_{\text{model}}}$ and $\mathbf{F}_{\text{TE}} \in \mathcal{R}^{d_{\text{model}}}$. The operator $\oplus$ is denoted as concatenation, and the function LBR represents a linear layer followed by BatchNorm and ReLU operators. The F_{encoder_i} consists of two sub-layers, namely self-attention mechanism and positional forward feedback:

$$\text{F}_{\text{encoder}_i}\big(\mathbf{F}_{i-1}\big) = \text{FFN}_i\big(\text{attention}_i(\mathbf{F}_{i-1})\big), \tag{6.3a}$$

$$\text{FFN}_i(\mathbf{x}) = \text{LBR}_{i,1}\big(\text{LBR}_{i,0}(\mathbf{x})\big) + \mathbf{x}. \tag{6.3b}$$

The layer FFN_i is a shared positional forward feedback network comprising two cascaded LBRs with the size of $[d_{\text{ff}}, d_{\text{model}}]$, where $d_{\text{ff}} = 2048$ and $d_{\text{model}} = 1024$.

Offset self-attention mechanism Self-attention is a mechanism that calculates the semantic relationship between different elements within a sequence of data. In the context of point cloud processing, attention is employed to build weights between every two positions in the feature space. In comparison to k-nearest neighbors algorithms, the attention mechanism has a larger receptive field. Furthermore, the attention mechanism's permutation invariant property makes it suitable for disordered, irregular data representation such as point clouds. The work in [52] proposed the offset attention by utilizing the idea of a Laplacian matrix $L = D - E$, where E is the adjacent matrix E and D is the diagonal matrix. The attention mechanism is adopted as

$$\begin{aligned} \mathbf{F}_{\text{sa,out}} &= \text{attention}(\mathbf{F}_{\text{sa,in}}) \\ &= \text{LBR}(\mathbf{F}_{\text{sa,in}} - \mathbf{F}_{\text{sa}}) + \mathbf{F}_{\text{in}}. \end{aligned} \tag{6.4}$$

The remaining part of the attention computation operators still follows the same design as in the original paper [176]. $\mathbf{F}_{\text{sa}}$ in (6.4) concatenates the multi-head attention with the following formulation:

$$\mathbf{F}_{\text{sa}} = \text{Linear}(\mathbf{F}_{\text{head}_1} \oplus \cdots \oplus \mathbf{F}_{\text{head}_h}), \tag{6.5}$$

where the attention feature at the i-head $\mathbf{F}_{\text{head}_i}, i \in [1, \dots, h]$ is formulated as

$$\mathbf{F}_{\text{head}_i} = \text{softmax}\Big(\frac{\hat{\mathbf{Q}}\hat{\mathbf{K}}^T}{\sqrt{d_k}}\Big)\hat{\mathbf{V}}, \tag{6.6}$$

with $\hat{\mathbf{Q}} = \text{Linear}(\mathbf{Q})$, $\hat{\mathbf{K}} = \text{Linear}(\mathbf{K})$, $\hat{\mathbf{V}} = \text{Linear}(\mathbf{V})$. The variables $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are projected with a different linear layer, respectively. Following the same principle as the original paper, I set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{F}_{\text{sa,in}} \in \mathcal{R}^{d_{\text{model}}}$. I reshape the linear projected query and key as $\hat{\mathbf{Q}}, \hat{\mathbf{K}} \in \mathcal{R}^{d_{\text{model}} \times 1}$ to obtain the attention weights $\mathbf{A}$ by matrix dot product via $\hat{\mathbf{Q}}\hat{\mathbf{K}}^T$. I normalize $\mathbf{A}$ with $\sqrt{d_k}$ to avoid large values in magnitude, where $d_k = \frac{d_{\text{model}}}{h}$. The equation in (6.6) shows, that the self-attention $\mathbf{F}_{\text{head}_i}$ is equal to the weighted sums of the value vector Linear($\mathbf{V}$) using the corresponding attention weights. The multi-head attention mechanism can jointly capture information from different representation subspace at different positions [176]. Therefore, it can efficiently preserve and capture the point cloud's detailed topology and structure for predicting the missing regions in comparison to [51, 168].

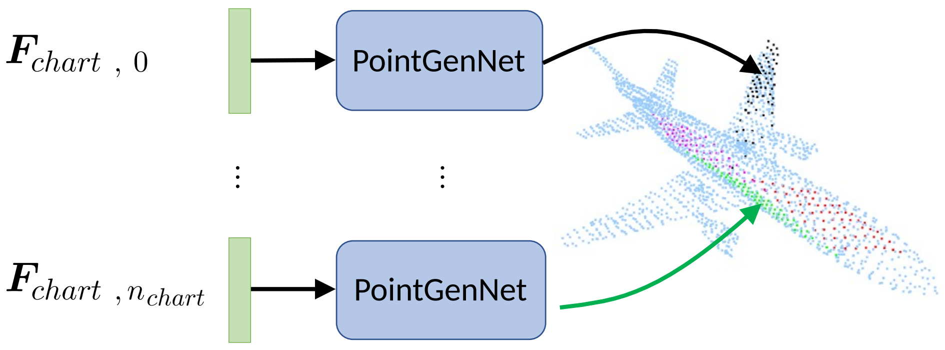


Figure 6.2: Visualization of MA-Network

6.3.3 Morphing-Atlas-Based Point Generation Network

At the first stage, the Transformer encoder extracts a global feature $\mathbf{F}_{\mathrm{TE}}$ for expressing an incomplete point cloud. I then feed the extracted features into a morphing-atlas-based point generator for predicting continuous and smooth shapes. Atlas is defined in the topology for describing a manifold and an atlas is composed of each chart that, roughly speaking, describes the local region of the manifold. In the context of 3D shapes, the manifold can be considered as a shaped surface. Therefore, I can represent a 3D shape by combing all the charts. Based on the Atlas concept, I define a chart as $\mathcal{C}_i$ and let a designed decoder $\mathcal{D}_i$ learn to map a 2D grid to a 3D surface. Furthermore, I introduce a hyper-parameter n_{chart} to control the number of charts defined on a shape to predict a smooth and high-resolution shape. The global feature $\mathbf{F}_{\mathrm{TE}} \in \mathcal{R}^{d_{\mathrm{model}}}$ is duplicated $N_{\mathrm{out}}/n_{\mathrm{chart}}$ times and then concatenated with a mesh grid to describe a new feature, denoted as $\mathbf{F}_{\mathrm{TE},1} \in \mathcal{R}^{(d_{\mathrm{model}}+2)\times(N_{\mathrm{out}}/n_{\mathrm{chart}})}$. It beneficial to linearly project $\mathbf{F}_{\mathrm{TE},1}$ with different learned linear projections. This concept is similar to multi-head attention by allowing the model to obtain the shape features from different representation subspaces at different positions. Therefore, $\mathbf{F}_{\mathrm{TE},1}$ is duplicated n_{chart} times and each $\mathbf{F}_{\mathrm{TE},1}$ is fed to an MLP layer which produces a new hidden code, denoted as $\mathbf{F}_{\mathrm{chart},i} \in \mathcal{R}^{(d_{\mathrm{model}}+2)\times(N_{\mathrm{out}}/n_{\mathrm{chart}})}, i \in [1,\ldots,n_{\mathrm{chart}}]$. In the next step, for each single chart, I feed $\mathbf{F}_{\mathrm{chart},i}$ into a PointGenNetwork, as illustrated in Fig. 6.2. In the end, all charts are concatenated to form a complete shape.

6.3.4 Evaluation Metrics

I apply the Chamfer distance (CD) [40] as a quantitative evaluation metric due to its efficient computation compared to the earth mover's distance [40]. The Chamfer distance measures the mean distance between each point in one point cloud to its nearest neighbor in another point cloud. Let $S_G = [x_i, y_i, z_i]_i^{n_G}$ be the ground truth and $S_R = [x_i, y_i, z_i]_i^{n_R}$ be the reconstructed point by given a partial point cloud. n_G and n_R indicate the number of points in S_G and S_R, respectively. The Chamfer distance d_{CD} of S_G and S_R with $L2$ norm is formulated as

$$d_{\mathrm{CD}} = \frac{1}{n_R}\sum_{x\in S_R}\min_{y\in S_G}||x-y||^2 + \frac{1}{n_G}\sum_{y\in S_G}\min_{x\in S_R}||x-y||^2 . \tag{6.7}$$

6.3.5 Implementation details

I implemented PCTMA-Net in PyTorch, where the model is optimized with an Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, together with a CosineAnnealingLR scheduler. The number of

Table 6.1: Point completion results on Completion3D with ground truth and input resolution (2048 points) compared using Chamfer distance (CD) with L^2 norm. The results are multiplied by 10^4. In my algorithm (PCTMA-Net), I set meshgrid $= 0.05$. The best result is highlighted in *green*, and a lower value is better.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Overall
AtlasNet ($N_{\text{out}} = 2048$) [51]	5.82	29.28	11.02	27.11	34.04	19.11	29.27	15.55	21.40
AtlasNet ($N_{\text{out}} = 16384$) [51]	5.50	19.89	9.23	21.17	30.99	15.34	21.67	14.64	17.31
FoldNet [188]	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
FCN [193]	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
TopNet ($N_{\text{out}} = 16384$) [168]	5.85	21.27	10.03	20.09	22.98	14.65	24.25	11.78	16.36
PointNetFCAE ($N_{\text{out}} = 2048$) [1]	5.81	21.14	8.95	22.01	33.36	15.81	27.52	14.09	18.59
PointNetFCAE ($N_{\text{out}} = 16384$) [1]	4.00	16.70	6.24	14.63	18.15	10.99	15.77	8.55	11.88
SA-Net [180]	5.27	14.45	7.78	13.67	13.53	14.22	11.75	8.84	11.22
GRNet ($N_{\text{out}} = 2048$) [185]	7.64	24.06	12.02	24.62	28.73	18.85	32.90	12.48	20.16
GRNet ($N_{\text{out}} = 16384$) [185]	3.79	14.86	6.71	12.74	13.73	11.05	15.43	6.50	10.60
Ours ($n_{\text{chart}} = 32, N_{\text{out}} = 2048$)	3.60	14.67	7.03	14.04	20.61	10.66	18.01	7.62	12.03
Ours ($n_{\text{chart}} = 128, N_{\text{out}} = 10240$)	3.16	13.53	6.58	13.21	12.93	10.29	14.25	6.98	10.11
Ours ($n_{\text{chart}} = 32, N_{\text{out}} = 16152$)	3.38	13.00	6.12	12.72	11.87	9.18	12.43	7.17	9.48

encoder layers used in the Transformer encoder is set to 4, and I follow the original papers by setting the multi heads in the offset attention mechanism to 8. I trained the network on a Linux system with a 2.600 GHz Intel Core i7–6700HQ, 16.000 GB of RAM, and one Nvidia RTX 2080 Ti GPU.

6.4 Experiments

I compare my proposed shape completion algorithm PCTMA-Net with other state-of-the-art approaches on two large scale data sets: Completion3D [168] and KITTI [47]. The Chamfer distance is employed as a metric in the evaluation.

6.4.1 Shape completion on Completion3D data set

Completion3D [168] from ShapeNet [22] offers a data set, which consists of 28974 training samples and 800 point cloud evaluation samples with a point resolution of 2048 for training and validation, respectively. In the comparison, I use different output resolutions and the quantitative results are summarized in Table 6.1. Note that the results of FoldNet [188], SA-Net [180], and PCN [193] are cited from the Completion3D benchmark leaderboard. Table 6.1 shows that my PCTMA-Net algorithm outperforms the other methods in 6 out of 8 categories with the overall Chamfer distance of 9.480 for $N_{\text{out}} = 16\,152.000$ and $n_{\text{chart}} = 32$. The qualitative visualization of completion results shown in Fig. 6.3 indicates that my approach is able to predict more details. The performance in the quantitative and qualitative evaluations proves the Transformer encoder and the morphing-atlas decoder's effectiveness for predicting and preserving the shape details.

6.4.2 Shape completion on robustness of input resolution

The input resolution can greatly affect the performance of a neural network. In this section, I will study the robustness of input resolution on the different network structures. I downsample the evaluation data set from Completion3D to obtain four levels of input resolutions:

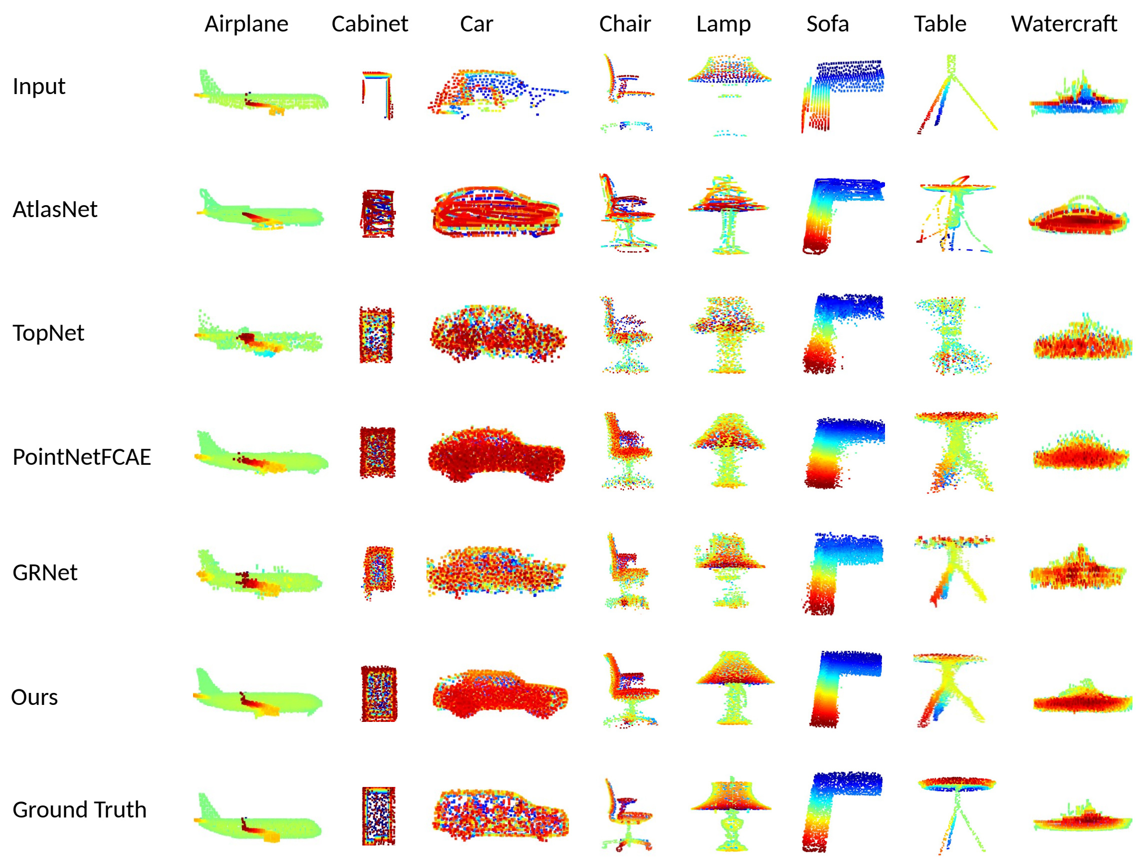
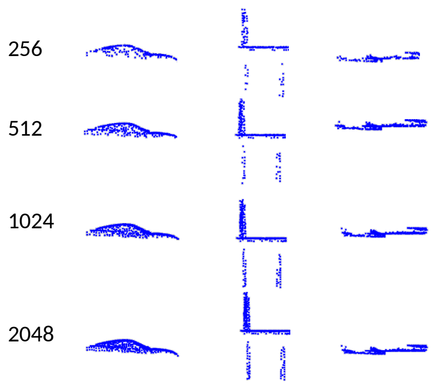


Figure 6.3: Visualization of completion results on the Completion3D evaluation set.

256, 512, 1024, and 2048. The visualization of these four levels of input resolutions is shown in Fig. 5.5a. All networks are trained on an input resolution of 2048 and output a fixed size of 16 384.000 points. For point resolutions less than 2048, I follow the principle in PCN [193] to select points from the input randomly and pad the input cloud to raise the number of points to 2048. I evaluate these four levels of input resolution on the Completion3D data set. The quantitative illustration in Fig. 5.5b indicates that my network has the best robustness and outperforms the other approaches in all four input resolutions experiments.

6.4.3 Shape completion on KITTI data set

For a further study of the application area, I conduct experiments on the KITTI data set [47], which is collected from real-world Velodyne Lidar scans composed of 2401 highly sparse point clouds. Note that the KITTI data set does not include the ground truth in a quantitative evaluation. Therefore, I can only qualitatively visualize the shape completion results. Unlike other work [168, 185], which trains the network with only the car category in ShapeNet [22] and then evaluates the KITTI data set, I use the same trained network as in Section 6.4.1 for evaluation. This evaluation strategy can show the capability of the generalization of one network. The incomplete point clouds from KITTI have diverse input resolutions and are highly sparse. I use the same strategy as in Section 6.4.2 to lift the number of points to 2048. Besides, I transform the incomplete point cloud by using the 3D bounding boxes to get a point cloud that is distributed between $[-0.5, 0.5]$. The qualitative result illustrated



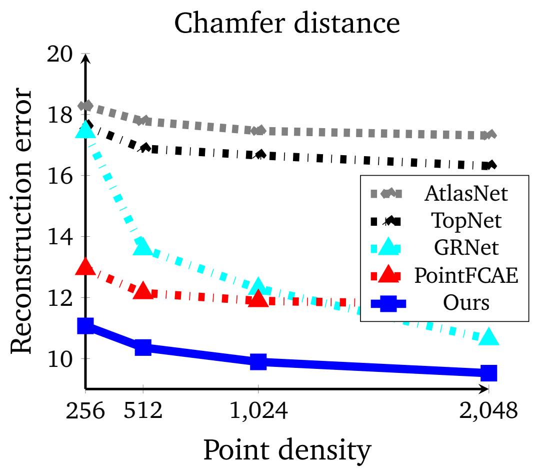


Figure 6.4: In a the point resolution varies from 256, 512, 1024 to 2048. In b, I compare the proposed approach against other state-of-the-art approaches on the Completion3D benchmarks. Lower values are better.

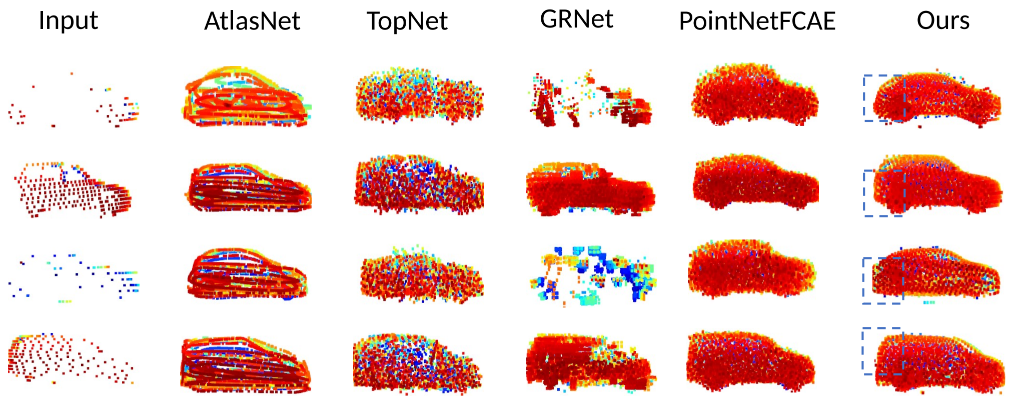


Figure 6.5: Qualitative completion results on the LiDAR scans from KITTI. The incomplete input point cloud is extracted and normalized from the scene with its 3D bounding box.

in Fig. 6.5 indicates that my approach and PointNetFCAE can generate more detailed shape information compared to the other methods.

6.4.4 Ablation Studies

In this section, I will study the effectiveness of my designed structure and chosen hyper parameters. All studies are conducted on the Completion3D data set for consistency. Without loss of generality and without special instructions, I set $N_{\text{out}} = 10\,240.000$ and $n_{\text{chart}} = 32$ in the following experiment.

Effect of Transformer encoder

The Transformer encoder is the main core used in PCTMA-Net, which has two hyper parameters: the number of encoder layers n_{encoder} and the number of heads h used in the attention mechanism. In this section, I will study the effect on shape completion by varying different combinations of these two parameters. I can conclude from Table 6.2, that I can achieve

Table 6.2: The Chamfer distance (CD) on different hyper parameters in the Transformer encoder.

n_{encoder}	2		4		6	
h	4	8	4	8	4	8
CD ($\times 10^4$)	10.86	10.41	10.59	10.21	10.69	10.21

Table 6.3: The Chamfer distance (CD) on the number of charts.

n_{chart}	8	32	128
CD ($\times 10^4$)	10.45	10.21	10.11
parameter ($\times 10^6$)	52.75	93.26	258.73

better shape completion performance with higher numbers of h and n_{encoder}. Taking various factors such as the network parameters into consideration, I set these hyper parameters to $h = 8$ and $n_{\text{encoder}} = 4$.

Effect of number of charts

The hyper parameter n_{char} is used to control the number of charts defined on a shape. In this section, I will study the effectiveness of the number of charts. I summarize the results in Table 6.3. It can be shown, that PCTMA-Net can result in a smaller Chamfer distance with a greater number of charts. However, the parameters of the network will be increased correspondingly, which is shown in the second row of Table 6.3.

Effect of grid strategy

In my proposed morphing-atlas decoder, the pointGenNet maps 2D grids to 3D surfaces. In this section, I will use the plane grid for point generation, which introduces two additional values. I can either randomly sample the value from $[0, 1]$ or use a grid with a predefined grid scale and grid size. The evaluation results on different grid strategies are listed in Table 6.4. It can be shown, that the mesh grid method shows significantly better performance in comparison to the randomly sampled grid methods. I further study the effectiveness of the grid scale by using the same grid size. The results in Table 6.4 show that the mesh grid scale from 0.05 to 0.5 has a similar performance.

Effect of metrics

Most existing work employs the Chamfer distance as a loss function due to its efficient computation. The earth mover's distance (EMD) is another option for point clouds with:

$$d_{\text{EMD}}(S_R, S_G) = \frac{1}{|S_G|} \min_{\Phi: S_R \to S_G} \sum_{X \in S_R} ||x - \Phi(x)||_2, \tag{6.8}$$

where Φ is the bijection function. In this section, I will study the effect on shape completion of different training loss functions. The comparison results in Table 6.5 demonstrate, that for a pure EMD loss function, the shape completion value with the metric of CD has the worst performance. The utilization of CD and EMD in the loss function can reduce the Chamfer distance value, and generate a more uniformly distributed point cloud than the pure CD loss function. As EMD uses the bijection function to force the output to have the same density

Table 6.4: The Chamfer distance (CD) on different grid types. In Meshgrid (k), k indicates the grid scale.

Grid type	Rand grid	Meshgird (0.5)	Meshgrid (0.05)
CD ($\times 10^4$)	11.36	10.25	10.21

Table 6.5: The Chamfer distance (CD) on different loss functions.

Loss function	EMD	CD+EMD	CD
CD ($\times 10^4$)	16.12	10.45	10.21

distribution as the ground truth for coping with the linear assignment problem. It hence can generate a point cloud which is more discriminative to local details. However, EMD is much more computationally expensive with approximately $\mathcal{O}(n^2)$, where n is the number of point cloud, compared to CD.

Effect of point generator

In this section, I study the effect of different point generators on shape completion, introduced in FoldNet [188], TopNet [168], by attaching them to my Transformer encoder. The results are summarized in Table 6.6. All of these three networks have improved to some degree by using the Transformer encoder. FoldNet shows an improvement from 19.07 to 13.22, TopNet improved from 16.36 to 13.49, and the performance of AtlasNet improved from 17.31 to 11.36.

6.5 Conclusion

I propose a novel network named PCTMA-Net for point cloud completion. Through its encoder-decoder structure, PCTMA-Net can effectively capture features of local regions for predicting missing shape parts. The utilization of the concept of an atlas further helps the network to reconstruct a smooth shape with a predefined number of charts. I conducted extensive experiments on the Completion3D and KITTI data sets to validate my proposed network structure's effectiveness. Via the experiments, I can conclude that my approach outperforms other state-of-the-art approaches on these two large data sets.

Table 6.6: The Chamfer distance (CD) on different point generators. I abbreviate my Encoder as TE and connect to different algorithm point generators.

Methods	TE-FoldNet	TE-TopNet	TE-AtlasNet
CD ($\times 10^4$)	13.22	13.49	11.36

Part IV

Algorithms in Grasp Planning

Chapter 7
Gaussian Process Implicit Surface and Bayesian Optimization based Grasp Planning

Chapter Summary

This chapter proposes a general algorithm that combines analytical grasp planning and Bayesian optimization. I parameterize the object surface with the Gaussian Process Implicit Surface for unifying the solving process. Besides, a two-stage optimization procedure is used to figure out the hand posture and hand finger configuration iteratively. The incorporation of Bayesian Optimization and alternating direction method of multipliers is further investigated in the grasp optimization loop. This chapter is a slightly modified version of peer-reviewed conference paper ©2021 IEEE. Reprinted, with permission, from

- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "Grasp Planning for Flexible Production with Small Lot Sizes using Gaussian Process Implicit Surfaces and Bayesian Optimization," IEEE International Conference on Automation Science and Engineering (CASE), 2021

The images created, algorithms designed, data from experiments and text written by me in this publication will be directly referenced in this chapter. The original version is referred to 162

Contributions

I took a leading role in the writing and revising of the manuscript in this article. I have made the following significant personal contributions to the formulation, implementation, and evaluation of the algorithms in this paper: utilizing the GPIS for describing the grasped object, designing two-stage optimization procedure. I am the lead developer of the algorithm implementation, and resposible for experimental evaluation.

7.1 Introduction

Only a limited number of small and medium-sized enterprises in Europe use robot systems in production, mainly dealing with small lot sizes and requiring a more flexible production process. It is, however, very time-consuming and expensive to adapt a robot system to a new production line, and it requires expert knowledge for deploying such a system, which, however, is not commonly available in shop floor workers [126]. Intuitive programming is currently proposed on the market for accelerating programming and remedying the problems caused by a lack of expert knowledge. Moreover, the service robots use semantic knowledge in combination with reasoning, and inference [169] to solve the declarative goal. Automatically synthesizing a robot program based on the semantic product, process, and resource descriptions enable an automatic adaptation to new processes. In this process, the recognition of objects and parts in the environments is involved, which is typically designed in CAD systems and described via a boundary representation [127]. Due to the small lot size production of SMEs, it is not feasible to train the objects over a long period of time by using the data-driven approaches. Based on this observation, it will accelerate the deploying time if I grasp the object firstly in a simulator with the CAD models and then transfer the preplanned grasp to the real world with a 6D pose estimation [95]. Fig. 3.3 shows an example of such a grasping use case for a mechanical gearbox together with a point cloud scene captured by the 3D camera sensor attached to the robot. Dexterous robotic grasping planning has been an active research subject in the robotic community over the past decades. Grasping is essential in many areas such as industrial factories and household scenarios. There have many different kinds of robotic hands, i.e., traditional parallel-jaw grippers, complex multi-fingered hands, or even vacuum-based end effectors. The goal of grasp planning aims to find a proper contact on the object and an appropriate posture of the hand related to the object to maximize grasp quality. This is a challenging task, especially for multi-fingered hands, due to different kinds of object shapes, the complicated geometric relationship between robotic hands and objects, and the high dimensionality of hand configurations. Grasp planning can be divided into analytic approaches [9] on the one side and empirical or data-driven approaches on the other side [13]. The analytic grasp synthesis approach is usually formulated as a constrained optimization problem over criteria that measure dexterity, equilibrium, and stability and exhibit a certain dynamic behavior. Besides, it requires the analysis of statically-indeterminate grasps [9] and under-actuated systems. The latter describes hands, in which the number of the controlled degrees of freedom is fewer than the number of contact forces, therefore further increasing the complexity of grasp synergies. One common assumption made in analytic methods is that precise geometric and physical models are available to the robot. Furthermore, optimizing grasp quality with constraints based on a convex optimization solver such as SQP can not guarantee finding a good grasp. In contrast to analytic approaches, the empirical or data-driven approaches rely on sampling the grasp candidate either from a data set or by first learning a grasp quality and then selecting the best by ranking them according to some specific metric. In this work, I will study how to optimize the palm pose and contact point in the same framework by utilizing a global Bayesian optimization solver under consideration constraints. Gaussian Process Implicit Surface Atlas (GPIS-Atlas) is used to parameterize the diverse shape. Therefore the geometry information can be integrated into the Bayesian optimization framework. Furthermore, GPIS-Atlas has the capability to describe the perfect geometry model in the form of a CAD model or the noisy point clouds [181]. In the work [95], the 6D pose is estimated between an object in the form of CAD model and the corresponding point clouds taken from an Ensenso Camera. Therefore, for finding an appropriate grasp pose for this object, I can directly apply my Bayesian optimization framework with the CAD model instead of working on noisy point clouds. After that, I transform the grasp pose using the 6D pose transformation from [95].

7.2 Related work

Multi-fingered hand grasp planning is still challenging due to the high dimensionality of hand structure and complex graspable object shapes. Automatic grasp planning is a difficult problem because of the vast number of possible hand configurations. Several different approaches have been proposed to find an optimal grasp pose over the past decades. Goldfeder et al. [49] introduced a database-backed grasp planning, which uses shape matching to identify known objects in a database with are likely to have similar grasp poses [49]. Ciocarlie et al. presented [24] Eigengrasp planning defines a subspace of a given hand's DOF space and utilizes the Simulated Annealing planner to find an optimized grasp. Miller et al. [107] proposed a primitive shape-based grasp planning which generates a set of grasps by modeling an object as a set of shape primitives, such as spheres, cylinders, cones, and boxes. Pelosso et al. [125] use an approach based on Support Vector Machines that approximate the grasp quality with a new set of grasp parameters. It considers grasp planning as a regression problem by given a feature vector, which should be defined heuristically. With the continuous success of deep learning vision, researchers utilize deep learning, also in combination with reinforcement learning, to learn a grasp quality directly from an image via large training data sets [61]. Levine et al. [88] used between 6 and 14 robots at any given point in time to collect data in two months and train a convolutional neural network to predict grasp success for a pick-and-place task with a parallel-jaw gripper. Mahler et al. [103] proposed a Dex-Net-based deep learning framework using a parallel-jaw gripper or vacuum-based end effector learn a grasp policy based on millions of grasp experiments. Kalashnikov et al. [66] introduced a scalable self-supervised vision-based reinforcement learning framework to train a deep neural network Q-function by leveraging over 580k real-world grasp attempts. However, the deep learning-based algorithm can only take the 2d image as an input, and the trained neural network cannot be easily transferred to another robotic hand configuration. Varley et al. [175] obtained the geometry representation of grasping objects from point clouds using a 3D-CNN. Ten et.al [124] is the state of the art 6 DOF grasp planner (GPD). Liang et.al. [92] proposed an end-to-end PointNetGPD to detect the grasp configuration from a point sets. Mousavian et al. introduced a 6DOF GraspNet by sampling a set of grasping using a variational autoencoder. In addition to deep learning, the Bayesian optimization-based algorithm in [119] can consider uncertainty in input space to find a safe grasp region by optimizing the grasp quality. Furthermore, it utilizes unscented transformation-based Bayesian optimization (UBO), a popular nonlinear approximation method, to explore the safe region. However, UBO considers only the palm pose optimization without considering the contact point. I present a grasp planning approach in this work, where I combine Bayesian optimization with an analytical approach. I use the Grasp Wrench Space (GWS) [43] as grasp quality metric, which calculates the convex hull over discretized friction cones from the individual contact wrench spaces of all contacts. Due to the GWS metric's complexity, I explore the potential of Bayesian optimization to optimize this highly-nonlinear grasp quality problem. Since the hand posture (hand palm pose) and hand configuration can be considered separately because the finger's contact points on the object surface only depend on the hand posture and forward kinematics of hand joints, I propose a dual-stage approach: In the first stage, I optimize the hand palm pose without considering hand configuration, and in the second stage, I use the result of the first stage and optimize the contact points on the object surface. My approach optimizes a hand palm pose $\mathbf{T} \in SE(3)$ regarding its grasp quality. For this, I present the rotation in hyperspherical coordinates instead of a rotation matrix or quaternion. I further parameterize the object surface as a Gaussian Process Implicit Surface (GPIS) [181] and use a k-D tree to find the closest point between the current palm pose and object surface. Based on GPIS, I can further compute a chart $\mathcal{C}$ and the corresponding normal vector $\mathbf{N}_{\mathcal{C}}$ on this nearest point. Utilizing this chart, I can make a local adaption of the palm pose to find a better location concerning the object surface. In the second stage, I convert the problem of solving constraints between the contact points and the object surface to querying the GPIS given a known contact point. Since the standard framework of Bayesian optimization cannot solve this constraint optimization problem, I use the Alternating Direction Method of Multipliers (ADMM) [17] to assist the contact point optimization by decomposing the whole problem into a set of subproblems.

7.3 Problem Formulation

In general, to define a grasp, I need two sets of variables: the intrinsic variables to define the hand degrees of freedom (DOF) and the extrinsic variables to define the hand's position relative to the target object. Grasp planning is used to find the optimized contact points and an associated hand configuration to maximize grasp quality. The contact point on the object surface is denoted as $\mathbf{c} = [\mathbf{c}_1, \cdots, \mathbf{c}_n]$, where $\mathbf{c}_i \in \mathrm{SE}(3)$, and n is the number of fingers. I will assume that contact happens on the fingertip, and one finger only has one contact on the object surface $\mathcal{O}$. The finger joint of the hand configuration is described as $\mathbf{q} = [\mathbf{q}_1, \cdots, \mathbf{q}_m]$, where m is the DOF. Note that some finger joints are under-actuated (passive joint). Therefore the number of finger joints is not equal to the DOF. The pose of the palm is represented by $\mathbf{T}_{\text{palm}}(\mathbf{R}, \mathbf{p})$. Mathematically, the optimized problem can be formulated as:

$$\max_{\mathbf{c},\mathbf{q},\mathbf{T}_{\text{palm}}} Q(\mathbf{c}, \mathbf{q}, \mathbf{T}_{\text{palm}}) \tag{7.1a}$$

$$\text{s.t.} \quad \mathbf{c} = \mathrm{FK}_{\text{palm2c}}(\mathbf{q}, \mathbf{T}_{\text{palm}}) \in \mathcal{O} \tag{7.1b}$$

$$q_{\min,i} \leq q_i \leq q_{\max,i}, \qquad i = 1 \cdots m, \tag{7.1c}$$

where $Q(\mathbf{c}, \mathbf{q}, \mathbf{T}_{\text{palm}})$ is the GWS, which is a 6-dimensional convex polyhedron, the epsilon and volume quality metric introduced by Ferrari and Canny [43]. The epsilon quality is defined as the minimum distance from the origin to any of the hyperplanes defining the boundary of the GWS. In contrast, the volume quality is the volume of GWS. $\mathrm{FK}_{\text{palm2c}}$ is the forward kinematics from the palm pose to the contact points. The formulation (7.1b) constrains all contact points on the object surface $\mathcal{O}$. Furthermore, a contact is defined as any point where two bodies are separated by less than the contact threshold ϵ_c, but not interpenetrating. In this work, the object surface will be parameterized by using GPIS to easily check if the contact point satisfies the constraint (7.1b). The problem in (7.1) is a high-dimensional nonlinear constraint problem, besides the gradient of the objective function and constraints cannot be analytically computed. Further, the convex optimization solver can only find a local minimum. Based on those observations, Bayesian optimization is applied to find a near-global optimization solution for these problems. Since the palm pose and contact point are nearly independent, I can switch between optimizing the palm pose and the contact point.

7.4 Bayesian Optimization for Grasp Planning

Bayesian optimization is a global optimization method, which can be used to solve the problem

$$x_{\text{optimized}} = \max_{x \in \mathcal{X}} f_{\text{obj}}(x), \tag{7.2}$$

where the objective function $f_{\text{obj}}(x)$ is a black-box function or a function which is expensive to evaluate and $\mathcal{X} \subseteq \mathbb{R}^D$ is a bounded domain. I use the Latin Hypercube Sampling (LHS) to get the initial sampling, and save it as data set $\mathcal{D}_{0:t-1} = \{(\mathbf{x}_0, y_0), \ldots, (\mathbf{x}_{t-1}, y_{t-1})\}$, and learn a Gaussian process model (GP). The essential step is to choose an appropriate acquisition function. Here, I use Expected Improvement [65], which is defined as

$$\text{EI}(\mathbf{x}) = \mathbb{E}\big[\max\big(f_{\text{obj}}(\mathbf{x}) - f_{\text{obj}}(\mathbf{x}^+), 0\big)\big], \tag{7.3}$$

where $\mathbb{E}$ is the expectation function, $f_{\text{obj}}(\mathbf{x}^+)$ is the best observation with the location $\mathbf{x}^+$ so far. The $\text{EI}(\mathbf{x})$ can be evaluated analytically under the GP model as

$$\text{EI}(\mathbf{x}) = \mathbb{1}\big(\sigma(\mathbf{x})\big)\Big(\big(\mu(\mathbf{x}) - f_{\text{obj}}(\mathbf{x}^+) - \xi\big)\Phi(Z) + \sigma(\mathbf{x})\phi(Z)\Big), \tag{7.4}$$

with $Z = \mathbb{1}\big(\sigma(\mathbf{x})\big)\big(\frac{\mu(\mathbf{x}) - f_{\text{obj}}(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})}\big)$, where $\mathbb{1}(x)$ is the indicator function that is equal to 0 for $x \leq 0$ and equal to 1 otherwise. The mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$ are defined in the GP posterior at $\mathbf{x}$, and Φ and ϕ are the CDF (cumulative distribution function) and PDF (probability density function) of the standard normal distribution. ξ is a parameter which balances between the exploration and exploitation. The objective function in my algorithm for optimizing the grasp contact points is grasp quality which consist of epsilon and volume quality.

$$f_{\text{obj}}(\mathbf{x}) = \big(\mathbb{1}(q_\epsilon)q_\epsilon + \lambda q_{\text{volume}}\big), \tag{7.5}$$

where λ is a predefined parameter. The Matérn covariance function ($\nu = 5/2$) is chosen as kernel function for the Gaussian process model in the Bayesian optimization and can be described as:

$$K_{5/2}(d) = \sigma^2 \left(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2}\right) \exp\left(-\frac{\sqrt{5}d}{\rho}\right) \tag{7.6}$$

with hyper parameter σ and length-scale ρ. The parameter d is the distance between two query points. Since I need to optimize the palm pose, which is interpreted as a transformation. I define the distance between two transformation matrices as $\Delta_{\mathbf{T}} = \|\mathbf{p}_1 - \mathbf{p}_2\| + \|\log(\mathbf{R}_1^{\text{T}}\mathbf{R}_2)\|_{\text{F}}/\sqrt{2}$, where the term $\|\log(\mathbf{R}_1^{\text{T}}\mathbf{R}_2)\|_{\text{F}}/\sqrt{2}$ is the geodesic distance defined in the Riemann manifold. To optimize the hyperparameter, the algorithm RProp (Resilient Propagation) [12] is applied, a popular gradient descent algorithm that only uses the signs of gradients to compute updates and can dynamically adapt the step size for each weight independently.

The object surface in my algorithm is described as a GPIS and every point lying on the surface in the set $\mathcal{X}$, should satisfy the equality constraints $\mathcal{X}, = \{\mathbf{x} \in \mathbb{R}^3 : f_{\text{GPIS}}(\mathbf{x}) = 0\}$. Furthermore, the tangent space of each point on these surface will be computed by using

$$\begin{bmatrix} \nabla f_{\text{GPIS}}^{\text{T}}(\mathbf{x}_i) \\ \Phi_i^{\text{T}}(\mathbf{x}_i) \end{bmatrix} \Phi_i(\mathbf{x}_i) = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}, \tag{7.7}$$

where $\Phi_i(\mathbf{x}_i) \in \mathbb{R}^{3\times 2}$ is the basis of tangent space at the location $\mathbf{x}$ and $\nabla f_{\text{GPIS}}(\mathbf{x}_i) \in \mathbb{R}^{3\times 1}$ is the gradient of implicit function for $\mathbf{x}$. By using $\Phi_i(\mathbf{x})$, I can map $\mathbf{x}_i$ to $\mathbf{x}_i'$ with $\mathbf{x}_i' = \mathbf{x}_i + \Phi_i(\mathbf{x}_i)\mathbf{u}_i$, where $\mathbf{u}_i \in \mathbb{R}^{2\times 1}$ is a point in the local coordinate on this chart. The sample value $\mathbf{x}_i'$ is shown as black dots in Fig. 7.1a, the tangent vector on the chart is shown as red and green arrows, while the blue arrow shows the normal vector.

7.4.1 Hand Palm Pose Optimization (HPP-Opt)

To optimize the palm pose, I need to take the transformation $\mathbf{T}_{\text{palm}}(\mathbf{R},\mathbf{p}) \in \text{SE}(3)$ into account, where the rotation matrix $\mathbf{R}$ can be interpreted by using a unit quaternion $\mathbf{q}$. Since the unit quaternion manifold $\mathcal{M}_{\mathbb{H}}$ is an Riemannian manifold, by virtual equality of $\mathcal{M}_{\mathbb{H}}$ and 4D unit hypersphere $\mathcal{S}^3 = \{x \in \mathbb{R}^{3+1} : \|\mathbf{x}\| = 1\}$, the quaternion $\mathbf{q}$ can be represented in the hyperspherical coordinates with ϕ, ψ, θ where ϕ, ψ range over $[0, \pi]$ and θ ranges over $[0, 2\pi)$. Employing hyperspherical coordinates, the constraints of $\mathbf{q}$ disappear. Therefore, the palm pose optimization is converted to an unconstrained optimization, and it can be mathematically formulated as

$$\max_{\phi,\psi,\theta,\mathbf{t}_{\text{palm}}} f_{\text{obj}}\big(\mathbf{q}, \mathbf{T}_{\text{palm}}(\mathbf{R}(\phi,\psi,\theta), \mathbf{t}_{\text{palm}})\big). \tag{7.8}$$

To reduce the search space, I constrain the palm pose between two bounding boxes, represented by an axis-aligned minimum bounding box (AABB), denoted as $\mathcal{X}_{\text{AABB},i}$ with $i \in \{1, 2\}$. The smaller one is shown as an orange cube in Fig. 7.1a. I define the variable $\mathbf{x}_{\text{palm}} = [\mathbf{p}, \phi, \psi, \theta]^{\text{T}} \in \mathbb{R}^{6\times1}$. Sampling a point between two bounding box cannot be formulated mathematically, therefore, I use an ellipsoid $\mathcal{E}_1 = (a_1, b_1, c_1, a_0, b_0, c_0)$ to approximate $\mathcal{X}_{\text{AABB},1}$ and another ellipsoid $\mathcal{E}_2 = (a_2, b_2, c_2, a_0, b_0, c_0)$ to approximate the bigger bounding box $\mathcal{X}_{\text{AABB},2}$. As the result, the palm pose sample $\mathbf{t} = [t_x, t_y, t_z]$ can be formulated as

$$t_x = a_0 + r * a_1 \sin(\theta)\cos(\phi) \tag{7.9a}$$
$$t_y = b_0 + r * b_1 \sin(\theta)\sin(\phi) \tag{7.9b}$$
$$t_z = c_0 + r * c_1 \cos(\theta), \tag{7.9c}$$

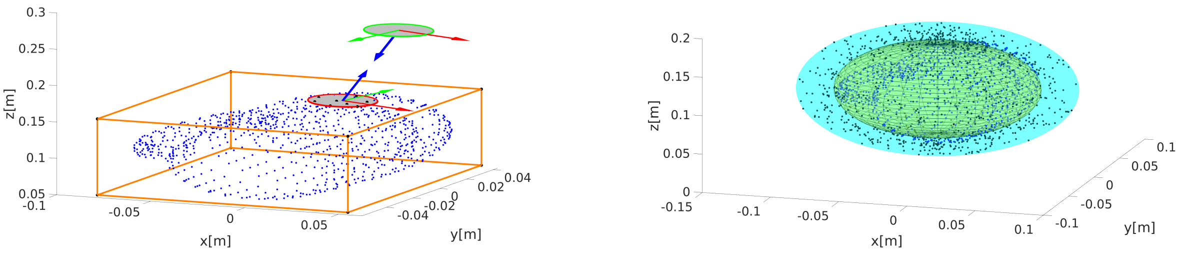
where $m_0 = \frac{k_{\text{min},1}+k_{\text{max},1}}{2}$, $m_1 = \frac{k_{\text{max},1}-k_{\text{min},1}}{2}$, $m \in \{a, b, c\}$, and $k \in \{x, y, z\}$. The parameter r is a random variable which ranges over $[1, r_{\text{max}}]$, where

$$r_{\text{max}} = \sqrt{\frac{1}{\left(\frac{a_1 \sin(\theta)\cos(\phi)}{a_2}\right)^2 + \left(\frac{b_1 \sin(\theta)\sin(\phi)}{b_2}\right)^2 + \left(\frac{c_1 \cos(\theta)}{c_2}\right)^2}}. \tag{7.10}$$

The parameter θ ranges over $[0, \pi]$ and θ ranges over $[0, 2\pi)$. The two ellipsoids are shown in Fig 7.1b. The problem in (7.8) can be solved by using Bayesian optimization, as shown in Alg. 2. The solution found by Bayesian optimization is based on the probability of best grasp distribution. To improve the performance, a local adaption based on GPIS-Atlas is proposed.

GPIS-Atlas based local adaption

The first step is to find the closest point from the current palm pose to the object using k-D tree algorithm. Assuming I found the pose $\mathbf{t}_{\text{closest}}$, a chart $\mathcal{C}_i$ with the center point $\mathbf{t}_{\text{closest}}$ is created



(a) (b)

Figure 7.1: Visualization of a Tangent space on a mug surface and b sampling for initial palm pose.

Algorithm 2 Palm Pose Optimization HPP-Opt

```
Require: $n_{\text{dim}}$, $n_{\text{iter}}$
 1: Get LHS $\mathcal{D}_{0:t-1} = \{(\mathbf{x}_0, y_0), \dots, (\mathbf{x}_{t-1}, y_{t-1})\}$
 2: Fit the Gaussian process Model $p(y|\mathbf{x}, \mathcal{D}_{0:t-1})$
 3: Optimized hyper Parameter Rrop
 4: for t = 1 to $n_{\text{iter}}$ do
 5:     $\mathbf{x}_{\text{palm},t} = \arg\max_{\mathbf{x}} \text{EI}(\mathbf{x}|\mathcal{D}_{0:t-1})$
 6:     Find a collision-free contact and get $\mathbf{q}, \mathbf{x}_{\text{palm}}$
 7:     $\mathcal{G} = \text{GPISAtlas}(f_{\text{GPIS}}, \mathbf{x}_{\text{palm}})$
 8:     $\{\mathcal{G}_{\text{best}}(\mathbf{x}_{\text{palm},t}, \mathbf{q}), y_{\text{max},t}\} = \arg\max_{\mathcal{G}} f_{\text{obj}}(\mathbf{q}, \mathbf{x}_{\text{palm}})$
 9:     Add the new sample $\mathcal{D}_{0:t} = \{\mathcal{D}_{0:t-1}, (\mathbf{x}_{\text{palm},t}, y_{\text{max},t})\}$
10:     update Gaussian process model (GP)
11: end for
12: $\mathbf{x}^+_{\text{palm}} = \arg\max_{\mathbf{x}_i \in \mathbf{x}_{0:t}} f_{\text{obj}}(\mathbf{x}_i)$
13: return $\mathbf{x}^+_{\text{palm}}$
```

by solving (7.7). I denoted the outward unit normal vector of this chart $\mathcal{C}_i$ as $\mathbf{N}_{\text{closest}}$ and the robot hand posture is designed to orient to the direction of a chart normal $\mathbf{N}_{\mathcal{C}}$. I apply the following approaches to get the pose $\mathbf{T}$. Assuming that the normal vector of hand in the initial state points to the z-axis $\mathbf{n}_z$, the hand is currently in local frame $\mathcal{H}_1$ with rotation matrix ${}^{\mathcal{W}}\mathbf{R}_{\mathcal{H}_1}$ with respect to the world frame $\mathcal{W}$. The next hand configuration should point to the normal direction $\mathbf{N}_{\mathcal{C}}$ in local frame $\mathcal{H}_{\mathcal{C}_i}$, therefore the corresponding rotation matrix ${}^{\mathcal{W}}\mathbf{R}_{\mathcal{H}_{\mathcal{C}_i}}$ can be interpreted in angle-axis representation $[n_{\text{axis}}, \theta_{\text{axis}}]$ with $\theta_{\text{axis}} = \text{atan2}(||\mathbf{n}_z \times \mathbf{N}_{\mathcal{C}}||, \mathbf{n}_z \cdot \mathbf{N}_{\mathcal{C}})$ and $n_{\text{axis}} = \mathbf{n}_z \times \mathbf{N}_{\mathcal{C}}$. As a result, I can transform the local frame $\mathcal{H}_1$ to $\mathcal{H}_{\mathcal{C}_i}$ with the rotation transformation as ${}^{\mathcal{H}_1}\mathbf{R}_{\mathcal{H}_{\mathcal{C}_i}} = {}^{\mathcal{W}}\mathbf{R}^{\text{T}}_{\mathcal{H}_1} {}^{\mathcal{W}}\mathbf{R}_{\mathcal{H}_{\mathcal{C}_i}}$. Furthermore, the translation of the palm pose is defined as $\mathbf{t}_{\text{palm}} = \mathbf{t}_{\text{closest}} + \lambda \mathbf{N}_{\text{closest}}$. This means that the new palm pose $\mathbf{x}_{\text{palm}}$ is parallel to the chart $\mathcal{C}_i$ with the distance $||\lambda||$, as shown in Fig. 7.1a. The parameter is optimized so that the hand is not colliding with the object. The whole transformation is defined as $\mathbf{T}_{\text{palm}} = \mathbf{T}(\mathbf{I}, \mathbf{t}_{\text{palm}})\mathbf{T}({}^{\mathcal{W}}\mathbf{R}_{\mathcal{H}_{\mathcal{C}_i}}, \mathbf{0})\mathbf{T}(\mathbf{R}_z, \mathbf{0})$. The transformation $\mathbf{T}(\mathbf{R}_z, \mathbf{0})$ is used to further guarantee no collision. Furthermore, I can define a point set on the chart $\mathcal{C}_i$ as $\mathcal{X}_{\mathcal{C}_i}$ and randomly choose a sample $\mathbf{t}_{\text{sample}} \in \mathcal{X}_{\mathcal{C}_i}$ as $\mathbf{t}_{\text{closest}}$.

7.4.2 ADMM-Based Contact Point Optimization (ADMM-CP-Opt)

The contact point optimization is used to find a set of desired joints $\mathbf{q}$ for each finger and the contact points $\mathbf{c}$ on the object surface. It can be described as

$$\max_{\mathbf{c},\mathbf{q}} \quad f_{\text{obj}}(\mathbf{q}, \mathbf{T}_{\text{palm}}) \tag{7.11a}$$
$$\text{s.t.} \quad \mathbf{c} = \text{FK}(\mathbf{q}, \mathbf{T}_{\text{palm}}), \tag{7.11b}$$
$$|f_{\text{GPIS}}(\mathbf{c}_k)| \leq \epsilon_c, \qquad k = 1 \cdots n \tag{7.11c}$$
$$q_k \in [q_{\min,k}, q_{\max,k}], \qquad k = 1 \cdots m, \tag{7.11d}$$

where FK calculates the forward kinematics, and n represents the number of contact points on the object surface. The parameter m is the DOF of a hand. A contact point is represented as a transformation $\mathbf{T}_{c_i}$. However, each finger has fewer joints than 6, which results in an underestimated system. Consequently, I cannot directly calculate the inverse kinematics based on the contact points, and it is not possible to arbitrarily move the fingertip on the object surface. To relax the constraints, I will not fix the palm pose, but constrain the palm pose on

Algorithm 3 GPISAtlas() function for local adaption

```
Require: $f_{\text{GPIS}}$, $\mathbf{x}_{\text{palm,current}}$
 1: Set $\mathbf{t} = \mathbf{x}_{\text{palm,current}}(0:2)$
 2: Get $\mathbf{t}_{\text{closest}} = \text{KD}(\mathbf{t}_{\text{palm,current}})$
 3: Set $\mathcal{G} = \{\}$, $\lambda \leftarrow 0$, $\theta_z \leftarrow 0$, $\lambda_{\max}$, $n_{\text{seed}}$
 4: Compute Chart $\mathcal{C}_i$ and $\mathbf{N}_{\text{closest}}$ by equation 7.7
 5: for n = 0 to $n_{\text{seed}}$ do
 6:     $\mathbf{t}'_{\text{closest}} = \mathbf{t}_{\text{closest}} + \breve{}_i(\mathbf{t}_{\text{closest}})\mathbf{u}_{\text{rand}}$
 7:     while C doheckCollsion
 8:         $\mathbf{t}_{\text{palm}} = \mathbf{t}'_{\text{closest}} + \lambda\mathbf{N}_{\text{closest}}$
 9:         if $\lambda < \lambda_{\max}$ then
10:             $\mathbf{T}_{\text{palm}} = \mathbf{T}_1(\mathbf{I}, \mathbf{t}_{\text{palm}})\mathbf{T}_2({}^{\mathcal{W}}\mathbf{R}_{\mathcal{H}_{\mathcal{C}_i}}, \mathbf{0})$
11:             $\lambda \leftarrow \lambda + \Delta_\lambda$
12:         else
13:             $\mathbf{R}_z = \text{Angleaxis}(0, 0, 1, \theta_z)$
14:             $\theta_z \leftarrow \theta_z + \Delta_{\theta_z}$
15:             $\mathbf{T}_{\text{palm}} = \mathbf{T}_1(\mathbf{I}, \mathbf{t}_{\text{palm}})\mathbf{T}_2({}^{\mathcal{W}}\mathbf{R}_{\mathcal{H}_{\mathcal{C}_i}}, \mathbf{0})\mathbf{T}_3(\mathbf{R}_z, \mathbf{0})$
16:         end if
17:     end while
18:     Execute the Grasp, get hand configuration $\mathbf{q}$
19:     $\mathcal{G} = \mathcal{G} \cup \{\mathbf{x}_{\text{palm}}, \mathbf{q}\}$
20: end for
21: return $\mathcal{G}$
```

the chart $\mathcal{C}_{\text{palm},i}$, which is parallel to chart $\mathcal{C}_i$ on the object surface, therefore $\mathbf{T}_{\text{palm}} \in \mathcal{C}_{\text{palm},i}$. Since the equality constraints cannot be solved by using Bayesian optimization, the Alternating Direction Method of Multipliers (ADMM) based Bayesian optimization [17] is utilized to solve the contact pose optimization problem (7.11) with the new formulation

$$\max_{\mathbf{q}\in\mathbf{B}} f_{\text{obj}}(\mathbf{q}, \mathbf{T}_{\text{palm}}) + g_c(\mathbf{q}, \mathbf{T}_{\text{palm}}), \tag{7.12}$$

with $g_c(\mathbf{q}, \mathbf{T}_{\text{palm}}) = \mu \sum_{i=0}^{n} \mathbf{c}_i(\mathbf{q}, \mathbf{T}_{\text{palm}})^2$ and $\mathbf{c}_i(\mathbf{q}, \mathbf{T}_{\text{palm}}) = \left| f_{\text{GPIS}}\Big(\text{FK}_i\big(\mathbf{q}, \mathbf{T}_{\text{palm}}(\mathbf{R}, \mathbf{p})\big)\Big) \right| - \epsilon_c$. In order to solve (7.12), ADMM introduces an auxiliary variable $\mathbf{z}$, resulting in

$$\max_{\mathbf{q},\mathbf{z}\in\mathbf{B}} \quad f_{\text{obj}}(\mathbf{q}, \mathbf{T}_{\text{palm}}) + g_c(\mathbf{z}, \mathbf{T}_{\text{palm}}) \tag{7.13a}$$

$$\text{s.t.} \quad \mathbf{q} = \mathbf{z}\,. \tag{7.13b}$$

In the following, I neglect $\mathbf{T}_{\text{palm}}$ in f_{obj} and g_c. By applying Augmented Lagrangian function for equation (7.13), a new objective function is formulated as

$$\mathcal{L}_\rho(\mathbf{q}, \mathbf{z}, \mathbf{y}) = f_{\text{obj}}(\mathbf{q}) + g_c(\mathbf{z}) + \frac{\rho}{2}\,\|\mathbf{q} - \mathbf{z} + \frac{\mathbf{y}}{\rho}\|_2^2 \tag{7.14}$$

Therefore, I can solve $f_{\text{obj}}(\mathbf{q})$ and $g_c(\mathbf{z})$ by alternating over the following sub problems:

$$\mathbf{q}^{k+1} = \arg\max_{\mathbf{q}} f_{\text{obj}}(\mathbf{q}) + \frac{\rho}{2}\,\|\mathbf{q} - \mathbf{z}^k + \frac{\mathbf{y}^k}{\rho}\|_2^2 \tag{7.15a}$$

$$\mathbf{z}^{k+1} = \arg\max_{\mathbf{z}} g_c(\mathbf{z}) + \frac{\rho}{2}\,\|\mathbf{q}^{k+1} - \mathbf{z} + \frac{\mathbf{y}^k}{\rho}\|_2^2 \tag{7.15b}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{q}^{k+1} - \mathbf{z}^{k+1})\,. \tag{7.15c}$$

The optimal condition is defined as $\|\mathbf{q}^{k+1} - \mathbf{z}^{k+1}\|_2 \le \epsilon^{\text{primal}}$ and $\|\rho(\mathbf{z}^{k+1} - \mathbf{z}^k)\|_2 \le \epsilon^{\text{dual}}$, where ϵ^{primal} and ϵ^{dual} are two predefined optimality tolerances. Each sub problem is solved by using Bayesian optimization. The algorithm is summarized in Alg. 4.

Algorithm 4 ADMM-Based Contact Point Optimization (ADMM-CP-Opt)

```
Require: $f_{\text{GPIS}}$, $\mathbf{T}_{\text{palm}}(\mathbf{R}, \mathbf{t}_{\text{palm}})$
 1: Compute Chart $C_i$ and get $\Phi_i(\mathbf{t}_{\text{palm}})$ solving equation 7.7
 2: for n = 0 to $n_{\text{seed}}$ do
 3:     $\mathbf{t}' = \mathbf{t}_{\text{palm}} + \Phi_i(\mathbf{t}_{\text{palm}})\mathbf{u}_{\text{rand}}$
 4:     for iter = 0 to $\max_{iter}$ do
 5:         Solve $\mathbf{q}^{k+1} = \text{BO}_q\big(f_{\text{obj}}(\mathbf{q}) + \frac{\rho}{2}\,\|\mathbf{q} - \mathbf{z}^k + \frac{\mathbf{y}^k}{\rho}\|_2^2\big)$
 6:         Solve $\mathbf{z}^{k+1} = \text{BO}_z\big(g_c(\mathbf{z}) + \frac{\rho}{2}\,\|\mathbf{q}^{k+1} - \mathbf{z} + \frac{\mathbf{y}^k}{\rho}\|_2^2\big)$
 7:         Update $\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{q}^{k+1} - \mathbf{z}^{k+1})$
 8:         Get $\epsilon_1 = \|\mathbf{q}^{k+1} - \mathbf{z}^{k+1}\|_2$, $\epsilon_2 = \|\rho(\mathbf{z}^{k+1} - \mathbf{z}^k)\|_2$
 9:         if $\epsilon_1 \le \epsilon^{\text{primal}}$ and $\epsilon_2 \le \epsilon^{\text{dual}}$ then break
10:         end if
11:         $\mathbf{y}^k \leftarrow \mathbf{y}^{k+1}, \mathbf{z}^k \leftarrow \mathbf{z}^{k+1}$
12:         Update $\text{BO}_q$, $\text{BO}_z$
13:     end for
14: end for
15: return $\mathbf{q}^{k+1}, \mathbf{T}_{\text{palm}}(\mathbf{R}, \mathbf{t}')$
```

7.4.3 Integration of HPP-Opt and ADMM-CP-Opt

The HPP-Opt and ADMM-CP-Opt are in detail introduced in the previous section. This section will integrate both optimizations in one framework and aim to find a near-global optimized grasp. The integration process is summarized in Alg. 5. Since HPP-Opt does not consider the hand finger configuration and the grasp is executed relying on the function of *AutoGrasp* from Graspit! [108], I executed ADMM-CP-Opt by using the result of HPP-Opt. The final solution combines the palm pose from HPP-Opt and the hand finger from ADMM-CP-Opt. It can seem that the ADMM-CP-Opt hat at least the one solution as HPP-Opt.

7.5 Experiment evaluation

Simulation results are introduced in this section to verify the effectiveness of my algorithm. The experiment is executed in the platform Graspit! [108] by using Barret hand Barret hand hat three fingers, finger one hat two joints, where the last joint is under-actuated. Finger two and three have one common joint. Each finger has three joints where both fingers have a passive joint. Therefore the Barret hand fingers have totaled 4 DOFs. The hand palm pose is denoted as a 6-dimensional vector. Therefore the whole Barret hand system has totaled 10 DOFs. The experiment's graspable object is stored in the mesh file, such as one example: mug The GPIS describes the object by using the mesh triangle verities as the input. My approach achieves a 95% success rate on various commonly used objects with diverse appearances, scales, and weights compared to the other algorithm. All evaluations were performed on a laptop with a 2.600 GHz Intel Core i7-6700HQ and 16.000 GB of RAM.

7.5.1 Experiment on HPP-Opt

In this section, the algorithm HPP-Opt is compared with other grasp planning. The first grasp planning approach is a simulated annealing grasp planner using an auto grasp quality energy as a search strategy, which behaves like a random grasp planning. The second approach uses an EigenGrasp planner combined with a simulated annealing solver to guide potential

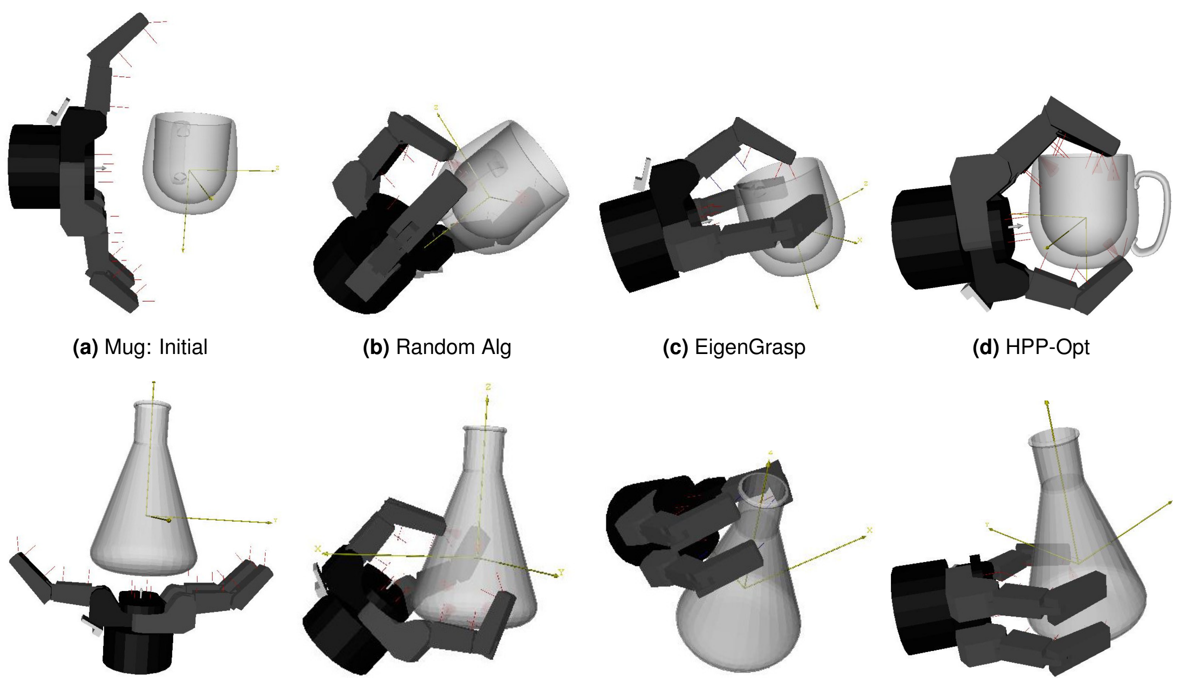
Algorithm 5 Integration of HPP-Opt and ADMM-CP-Opt

```
Require: $n_{\text{dim}}, n_{\text{iter}}$
 1: Get LHS $\mathcal{D}_{0:t-1} = \{(\mathbf{x}_0, y_0), \ldots, (\mathbf{x}_{t-1}, y_{t-1})\}$
 2: Fit the Gaussian process Model $p(y|\mathbf{x}, \mathcal{D}_{0:t-1})$
 3: Optimized hyper Parameter Rrop
 4: for t = 1 to $n_{\text{iter}}$ do
 5:     $\mathbf{x}_{\text{palm},t} = \arg\max_{\mathbf{x}} \text{EI}(\mathbf{x}|\mathcal{D}_{0:t-1})$
 6:     Find an no collision contact, and get $\mathbf{q}, \mathbf{x}_{\text{palm}}$
 7:     $\mathcal{G}_1$=GPISAtlas $(f_{\text{GPIS}}, \mathbf{x}_{\text{palm}})$
 8:     $\{\mathcal{G}_{\text{best}}(\mathbf{x}_{\text{palm},t}, \mathbf{q}), y_{\text{max},t}\} = \arg\max_{\mathcal{G}_1} f_{\text{obj}}(\mathbf{q}, \mathbf{x}_{\text{palm}})$
 9:     $\mathcal{G}_2(\mathbf{x}'_{\text{palm},t}, \mathbf{q}') = \text{Admm-CP-Opt}(f_{\text{GPIS}}, \mathbf{T}_{\text{palm}}(\mathbf{x}_{\text{palm},t}))$
10:     $y'_{\text{max},t} = f_{\text{obj}}(\mathbf{q}', \mathbf{x}'_{\text{palm}})$
11:     Add new sample $\mathcal{D}_{0:t} = \{\mathcal{D}_{0:t-1}, (\mathbf{x}'_{\text{palm},t}, y'_{\text{max},t})\}$
12:     update Gaussian process model (GP)
13: end for
14: $\mathbf{x}^+_{\text{palm}} = \arg\max_{\mathbf{x}_i \in \mathbf{x}_{0:t}} f_{\text{obj}}(\mathbf{x}_i)$
15: return $\mathbf{x}^+_{\text{palm}}$
```

Table 7.1: Evaluation of different grasp planning algorithm for all data sets. The results is an average of first 20 best Grasp Candidates. A greater value of epsilon and volume means a more stable grasp. The best result is highlighted in *green*

	Algorithms							
	random Alg		EigenGrasp		HPP-Opt		ADMM-CP-Opt	
quality	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}
Mug	0	9.5352e-05	1.7281e-04	7.1981e-04	0.0555	0.0097	0.06	0.0161
Flask	0	3.9603e-05	4.9050e-04	2.0142e-04	0.0107	0.0014	0.0107	0.0026
Phone	0	3.9209e-05	0.0017	4.6942e-04	0.0142	0.0035	0.0142	0.0042
Sphere	0.0042	0.0019	0	0.0011	0.0495	0.0121	0.050	0.0279
Bishop	0	8.1935e-05	0.0012	9.7708e-05	0.0094	0.0011	0.0094	0.0016

quality energy. The Bayesian optimization and simulated annealing both are global optimization solvers, where the latter one is a probabilistic technique for approximating the global optimum of a given function. I compare the algorithms with different kinds of shapes. The experiment is conducted in a fixed time of 20 seconds, and I average the first 20 best grasp candidates. The best grasp candidate of mug and flask from whole grasp candidates are visualized in Fig. 7.2. The initial state of Barret hand and object are randomly defined, shown in Fig. 7.2a and 7.2e. It can be seen that the hand configuration selected by random grasp planning is skewed to the object, and contact points on the surface are not properly, and the resulting quality is also worse. The results of EigenGrasp show a better solution where the palm pose is trying to parallel to the object surface. The palm pose selected by HPP-Opt is more reasonable in comparison to the other two algorithms. I show the quantitative result is in table (7.1). In different kinds of geometry shapes, my algorithm can achieve a much more stable grasp than other algorithms. The epsilon quality achieved by the first approach is almost zero besides in the case of a sphere. The epsilon quality by Eigen grasps a minimal value. On average, the epsilon quality of HPP-Opt is 28.5 times greater than Eigen grasp's epsilon quality. And the HPP-Opt's volume quality is 32.1417 times greater than Eigen Grasp's volume quality. Furthermore, I show the best grasp candidate from the first 20 best grasp candidates in Table 7.2.



(a) Mug: Initial **(b)** Random Alg **(c)** EigenGrasp **(d)** HPP-Opt

(e) Flask: Initial **(f)** Random Alg **(g)** EigenGrasp **(h)** HPP-Opt

Figure 7.2: Comparison of different Grasp planning Algorithm examples for different objects with the multi-fingered hand

7.5.2 Experiment of integration HPP-Opt and ADMM-CP-Opt

In section 7.5.1, hand palm pose is optimized based on the Bayesian optimization algorithm combing with local adaption, and hand finger configuration is set based on the function of *AutoGrasp* from Graspit [108]. The principle of *AutoGrasp* is to close each hand finger DOF at a rate equal to a predefined speed factor multiple with its default velocity, and the movement is stopped at the contact point. Therefore ADMM-CP-Opt is used to assist the HPP-Opt to find a better hand finger configuration. The comparison result is visualized in Fig. 7.3. In the case (7.3a), fingers two and three are too close and grasp the bottom of the flask. As a consequence, the resulting triangle has a small internal angle. Applying ADMM-CP-Opt splits the fingers two and three by maximizing epsilon and volume quality and makes the resulting triangle closer to the equilateral triangle with a more stable grasp. The same improvement happen in the Fig. 7.3b - Fig. 7.3f as well. The solution founded by ADMM-CP-Opt is trying to

Table 7.2: The best grasp from 20 grasp candidates. The best result is highlighted in *green*

	Algorithms							
	random Alg		EigenGrasp		HPP-opt		ADMM-CP-Opt	
quality	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}
Mug	0	0.0012	0.0065	0.0011	0.1039	0.0151	0.1158	0.0340
Flask	0	0.0004	0.0130	0.0031	0.0449	0.0018	0.0452	0.0019
Phone	0	0.0004	0.0310	0.0007	0.0208	0.0198	0.0335	0.0006
Sphere	0.0844	0.0301	0	0.0086	0.0914	0.0362	0.1676	0.0728
Bishop	0	0.0012	0.0164	0.0004	0.0390	0.0020	0.0444	0.0045

make the resulting triangle as closer as the equilateral triangle. In Table 7.1, ADMM-CP-Opt improves the volume quality of HPP-Opt. And In table 7.2, ADMM-CP-Opt shows a better grasp than HPP-Opt in most cases under epsilon and volume quality metrics.

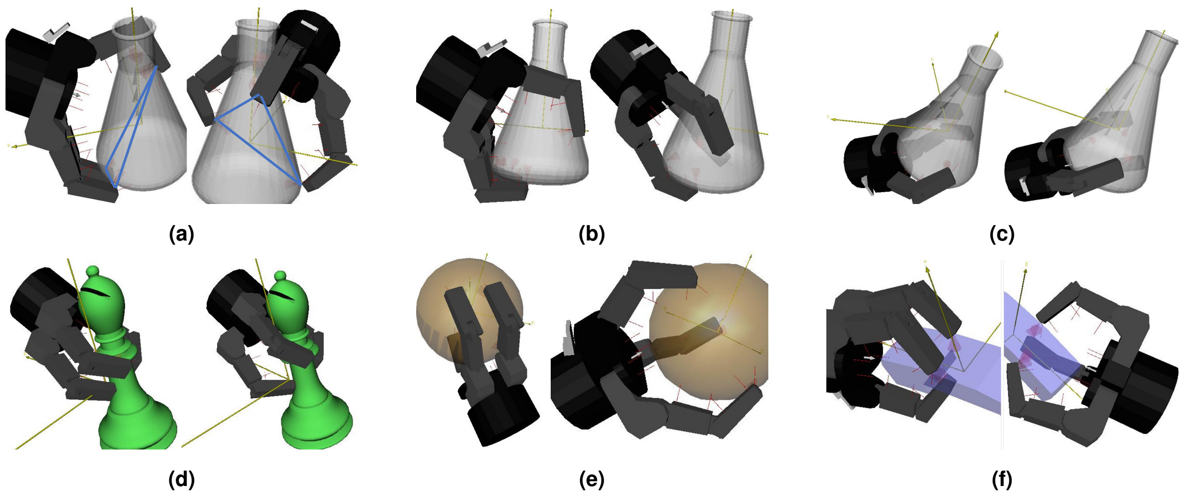


Figure 7.3: The comparison result of HPP-Opt and ADMM-CP-Opt. In each sub figure, the left one is result of HPP-Opt, and the right one is improved by using ADMM-CP-Opt

7.6 Conclusion

I propose a new algorithm for grasp planning with multi-fingered hands by optimizing the hand palm pose and hand finger configuration separately. Using a global Bayesian optimization solver, no initial configuration is required, which shows superiority over the convex optimization solver. I propose a dual-stage optimization process by considering the independence of the hand palm pose and finger configuration. In the first stage, I utilize a GPIS to describe the graspable object so that collision checking of contact points can be integrated into the optimization framework. Furthermore, the chart on the object surface can be computed using the GPIS, which can be used to explore the local information of objects. Two ellipsoids are used to define the palm pose constraints domain. Relying on the first stage result, I apply an ADMM based Bayesian optimization to optimize the contact points. The whole process will switch between HPP-Opt and ADMM-CP-Opt. I collect the best 20 Grasps, and the final grasp is selected by ranking the grasp candidates under consideration of epsilon and volume quality. In this work, I describe only the object in GPIS. In future work, I can describe the hand part into GPIS so that the collision checking can be converted to a problem by querying the distance between two surfaces. Besides, the robot arm is not considered in the grasping scenario. It will be interesting to integrate the constraints of robot arm manipulability in the objective function.

Chapter 8
Point Transformer with Lie Algebra Grasp Representation for an End-to-End 6-DOF Grasp Detection

Chapter Summary

With the recent massive progress in the 6-DOF Grasp learning network, grasp selection for unseens objects attracts a lot of attention. However, most of the existing approaches consist of a complex sequence pipeline for generating the potential Grasp. I propose an end-to-end grasp detection network in this work to create a diverse and accurate 6-DOF grasp posture based on pure point clouds. I adopt the hierarchical PointNet++ with a skip-connection point transformer encoder block to extract contextual local region point features, called LiePFormer-GraspNet, that efficiently generates a distribution of 6-DoF parallel-jaw grasps directly from a pure point cloud. Furthermore, I introduce two different grasp detection loss functions to enable a continuously differentiable property for the network. These two loss functions give the neural network the ability to generalize to unseen objects, like generators. Moreover, I train this GraspNet with the synthesized grasp data set ACRONYM containing 17 million parallel-jaw grasps and generalize well with an actual scanned YCB data set, which contains 77 objects. Finally, I conducted experiments in the PyBullet simulator. As a result, I demonstrate that my proposed grasp detection network can outperform most state-of-the-art approaches regarding the grasp success rate.

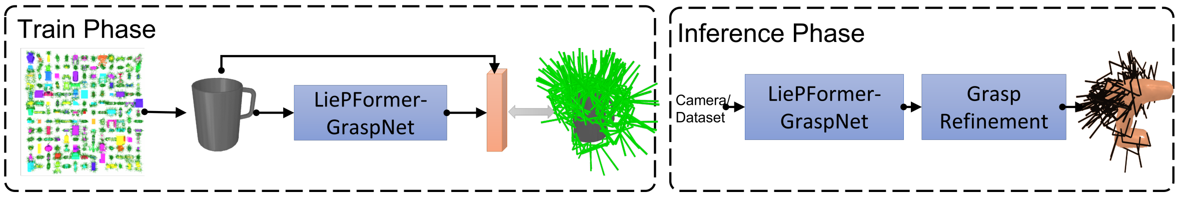


Figure 8.1: The visualization of the overall network pipeline. On the left side, one object from ACRONYM [38] is randomly chosen for training the LiePFormer-GraspNet and outputs diverse grasp configurations and grasp quality for regressing. A point cloud is taken for generating the grasping posture at the inference phase on the right side.

8.1 Introduction

Grasp detection is an essential task in robotic manipulation, especially in unstructured environments such as warehouses or households. Analytical-based grasp approaches can achieve a high grasp quality when provided with an accurate object model [97]. Nevertheless, it is still difficult to generalize to unseen objects and most approaches are limited to simulations. Data-driven approaches, especially machine-learning tools, can complement analytical-based grasp approaches to significantly reduce the required amount of information of object models. Most recent work represents a grasped object as an RGB image, so that the convolutional-based network can be exploited to generate the grasp configurations. These works are classified as 3/4-DOF (degree of freedom) grasping and constrain the gripper vertically to the objects (top-down grasp). This type of grasping can dramatically simplify the pick and place task problem. However, the constraints nature of 3/4-DOF reduces the potential to integrate motion planning, where motion planning algorithms try to explore all possible directions to generate a good motion trajectory. This limitation inspired researchers to study a more general approach (6-DOF grasp) [92, 110, 124, 137], enabling a grasp in an arbitrary direction. The proposed approaches in [92, 124] sample the grasp configuration in terms of the region of interest (ROI) and evaluate each sampled grasp configuration separately. However, the whole process is time-consuming, as the number of generated grasp configurations increases. In [110, 111], the variational auto-encoder (VAE) approach with an evaluation network is introduced for generating diverse grasp configurations. This VAE replaces the heuristic geometry sampler in [92, 124], which can significantly simplify the generation process. An evaluation network will further refine the candidate grasp configuration. I classify such approaches in [110, 111] as a two-stage approach. For further simplifying the grasp detection algorithm, [62, 117, 137] proposed a one-stage strategy, which directly outputs the grasp quality and grasp configuration at the same time. The proposed approaches in [117, 137] consider the grasp detection problem as a one-to-one pose regression, heuristically selecting the ground truth grasp configurations. The coarse-to-fine approach introduced in [62] takes a step further by considering the grasp detection as a one-to-multiple pose regression problem and requires a manual quantization of the grasp orientation for refinement.

In this work, I follow the *one-to-one grasp pose regression* strategy proposed in [117, 137], since the quantized orientation refinement is not restricted to the deep learning framework. I believe there is an infinite number of grasp configurations for an object that can grasp an object and that the similarity of candidate grasp configuration and ground truth depends on the distance between two grasp points. Based on this assumption, I propose the grasp quality loss function, which obeys the Gaussian distribution. Furthermore, most related work represents the orientation via Euler angles and later converts it to a rotation matrix. This conversion is shown to be discontinuous. In addition, translation and rotation regression loss are considered separately from each other. I utilize Lie algebra for representing a transformation matrix to overcome the discontinuous gradient problem and integrate the rotation and translation loss function into one. To generate diverse grasp configurations, I downsample the point cloud into a predefined number of points and consider each downsampled point as a grasp point. The grasp configuration is generated in terms of its corresponding grasp point. My proposed 6-DOF grasp detection algorithm follows the deep learning framework. It is trained with the synthetic data set ACRONYM [38] and evaluated with the scanned YCB data set [20], where the data was taken with a commodity depth camera and saved as a partial point cloud. I do not specify the object category for training. Instead, I utilize the proposed network for the generalization of unseen objects. I reference the grasp network as LiePFormer-GraspNet for two reasons. The first one is that I modify the hierarchically PointNet++ with a stacked skip-connected point transformer encoder as the backbone. Two separated MLP based networks are then used to predict the grasp quality and grasp configuration, respectively. The second reason is that the proposed network can directly output the Lie algebra vector for representing the grasping posture. I rank the grasp quality at the inference phase and select the highest grasp quality as the final candidate. Besides, I follow the concept of GraspIt! [108] by moving the potential grasp configuration in the approach direction until the gripper contacts the object. The overall structure of the network process is illustrated in Fig. 8.1.

8.2 Related Work

Grasp selection or grasp estimation is used to estimate the grasp posture in the world coordinate given the input sensor data, where the data can be image or point cloud. I can roughly categorize the grasp selection approaches into analytical-based and data-driven based. A common assumption in the analytical based grasp approach [9, 116] is that precise geometric and physical models of an object are available to the robot. In a large-scale industrial application, the objects are typically modeled and available in CAD. However, in the unknown scenario environment, such an assumption cannot be easily met. Moreover, surface properties or friction parameters are also essential for the analytical-based approaches. Several methods have tried to address these problems, such as in [194], where the author used a particle filter for an estimate of the surface parameter. In previous work [97], the authors proposed a novel approach with the combination of Gaussian Process implicit surfaces and Bayesian optimization for computing a grasp configuration, which can exploit the GPIS to model the object with noise implicitly. However, the approaches mentioned above are still limited to simulation and also error-prone.

The success of deep learning inspired researchers to exploit neural network structures for grasp detection. I can roughly categorize the deep learning-based grasp approaches into 2D planar grasp and 6-DOF grasp. In the 2D planer grasp, the researcher utilized a five-dimensional vector for presenting robotic grasps posture [63, 87], which are rectangles with a position, orientation and size: (x, y, θ, h, w). In the deep learning framework, three ways are typically applied for obtaining the oriented rectangle-based grasp configuration: classification-based [87], regression-based [80], and detection-based approaches [141]. However, the 2D planer grasp restricts the grasp in the direction of top-down. The 6-DOF grasp provides more possibilities to interact with the objects and considers the flexibility of the robotic manipulator in an environment. Grasp Pose Detection (GPD) proposed in [124] samples diverse candidate grasps around the region of interest (ROI). The generated grasp posture is then fed into an adopted CNN, which is classified from the perspective of grasp score. The PointNetGPD [92] considers the 3D geometry in a different way, which takes the point cloud directly as input for PointNet to learn a grasp quality network. The geometry-based grasp sampler is critical for these two approaches. In [110], an adopted variational

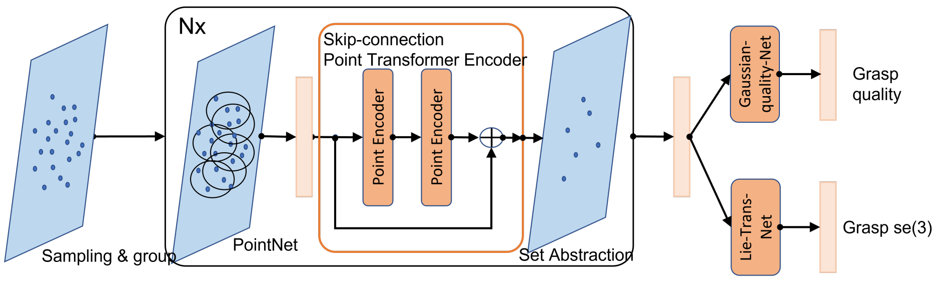


Figure 8.2: The overall structure of LiePFormer-GraspNet. The point cloud is first fed with a PointNet++ for extracting the point feature. Inside the set abstraction, the feature is provided to a skip-connected point transformers encoder for learning the semantic feature with the multi-head attention mechanism. The final contextual global feature will be separated into two MLP networks to learn the Lie algebra-based grasp representation and the Gaussian distributed-based Grasp quality.

Auto-Encoder (VAE) is proposed to train a grasp sampler for generating diverse sets of grasps and the gradient of the evaluation network is utilized to refine the generated grasp. Furthermore, an improved version of 6-DOF GraspNet is proposed in [111] by introducing a learned collision checker conditioned on the gripper information and the raw point cloud of the scene. Contact-GraspNet introduced in [166] reduces the dimensionality of grasp representation to 4-DOF for facilitating the learning process. In contrast to the two-stage approaches, a single-shot strategy is proposed in [137] to regress the grasp configuration by using PointNet++. With the same concept, PointNet++Grasp is introduced in [117], which can directly predict the poses, categories, and scores of all the grasps. These two approaches consider grasp detection as a regression and classification problem, reducing the diversity of grasp distribution.

8.3 Algorithm

8.3.1 Grasp Problem Formulation

Given the point cloud of an object in Cartesian world coordinates, the purpose of the LiePFormer-GraspNet is to find a set of possible 6-DOF grasp configurations for picking up the object with a parallel-jaw gripper from arbitrary direction. I describe the grasp configuration as $g = [c, \mathbf{T}]$, illustrated in Fig. 8.3b. The grasp quality is denoted as c, and the grasp pose is defined as a transformation matrix. In contrast to most grasp detection networks, which represent the grasp pose as $(x, y, z, r_x, r_y, r_z) \in \mathbb{R}^6$, I formulate the grasp transformation matrix using Lie algebra $\mathfrak{se}(3) \in \mathbb{R}^6$. Therefore, the grasp configuration in this work is denoted as $g = [c, \mathbf{v}]$ with $\mathbf{v} = [\mathbf{t}^{\mathrm{T}}, \omega^{\mathrm{T}}]^{\mathrm{T}} \in \mathcal{R}^6$, which is a 6-dimensional vector of coordinates in the Lie algebra $\mathfrak{se}(3)$. The $\mathfrak{se}(3)$ $\mathbf{v}$ comprises two separate 3-dimensional vectors: ω determines the rotation and $\mathbf{t}$ the translation.

8.3.2 Network Structure Design

Similar to most deep learning-based grasp network structures, I explore PointNet++ as my main skeleton to extract the point features. To enhance the capability of PointNet++, I modify its set abstraction structure by integrating a skip-connection point transformer encoder block [176]. The point transformer encoder [99, 195] is shown to have a strong capability to attend to some important features. Furthermore, the point transformer encoder structure with a skip connection for aggregating the feature can propagate larger gradients to initial layers. These layers can learn as fast as the final layers, giving us the ability to train deeper networks. The main benefit of using the hierarchically skip-connection transformer encoder is to learn an attention map between the point feature after the furthest point sampling [134], which is originally aggregated only using the max pooling and neglects the geometry relationship within the features. In addition, the attention mechanism is shown to be invariant to permutations and is therefore useful to the point cloud application. Furthermore, I follow the idea from [52], which replaces the self-attention with an offset-attention to enhance the transformer encoder. The offset-attention originates from Graph convolution networks [18], which show the benefits of using a Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{E}$ to replace the adjacency matrix $\mathbf{E}$, where $\mathbf{D}$ is the diagonal degree matrix. After extracting the network's main skeleton, two additional branches are deployed to compute the grasp quality and grasp configuration separately. These two branches are composed of MLP layers for regression. Here, the grasp quality in the ground truth is normalized between $[0,1]$. I use the sigmoid operator instead of softmax for regression since softmax is suitable for predicting the probability. Finally, the network is fed with a point cloud with the size of $\mathbb{R}^{2000\times 3}$ and predicts 256 grasp configurations to make sure the inference fits in the GPU memory.

8.3.3 Grasp Loss Design

I follow the same strategy as [137] to consider the grasp detection as a task of grasp pose and quality regression. As shown in Fig. 8.2, I simultaneously predict the grasp quality and grasp configuration with two MLPs.

Gaussian-based Grasp Quality Loss

I formulate the prediction of grasp quality as a regression problem. In [62], the authors used the focal loss for computing the classification loss by assigning each subsample grasp point with a predefined score by checking if the grasp point is enclosed by the gripper using the ground truth grasp posture. In this work, I consider the assignment of grasp quality from another perspective. The assumption here should be made firstly that a predicted grasp quality follows the Gaussian distribution. The closer to the assigned ground truth grasp configuration, the higher the grasp quality. To regress the point-wise 6-DoF grasp pose, I assume each point q in the point set has its best grasp configuration g, corresponding to the ground truth. In contrast to most work, which considers the regression unique and fixed, I use the Gaussian distributed-based grasp quality to enlarge the grasp configuration by allowing the predicted point to slightly deviate from the ground truth. The point is denoted as grasp center g_t as assumed in most works. Therefore, for each selected point, I use the k-nearest neighbors (KNN) to find the nearest grasp point inside the ground truth grasp point set, then I can compute the ground truth grasp quality at this grasp point. It can be formulated in the following:

$$c_{j,g} = \exp(-\frac{||\mathbf{q}_j - \mathbf{q}_g||^2}{\delta^2}) \tag{8.1a}$$

$$L_{\text{score}} = \frac{1}{|S|}\sum_j ||c_{j,p} - c_{j,g}||^2, \tag{8.1b}$$

where $\mathbf{q_j}$ is the selected grasp point, and $\mathbf{q_g}$ is the corresponding grasp point, which is chosen by using the KNN. The score $c_{j,g}$ is the computed grasp quality at the grasp point $\mathbf{q}_j$, where $c_{j,p}$

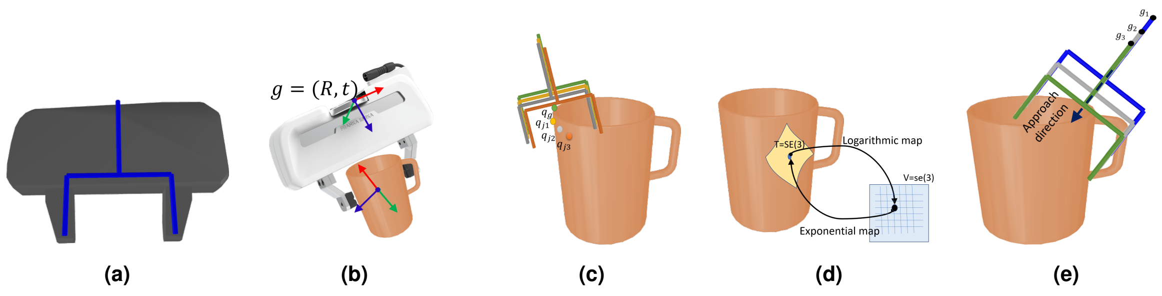


Figure 8.3: Visualization of the parallel Franka Emika gripper setting: (a) a simplified parallel gripper (blue line) with its Franka Emika gripper collision mesh, (b) Franka Emika grasp configuration with respect to the mug, (c) a set of translation-shifted grasp configurations at different grasp points with respect to the ground truth grasp posture, (d) a grasp configuration chart defined on the mug with its corresponding Lie algebra map. (e) Visualization of the refinement process. The generated grasp pose moves in the direction of the gripper approach and stops until the predefined contact condition is met.

is the predicted grasp quality from the network. The Gaussian distribution quality assignment simplifies this process, while in most work a predefined parameter is used to assign the grasp success [110, 111].

Lie Algebra Transformation Loss

The widely used L_1 or L_2 metrics for computing the distance between Euler angles has been proven [137] to suffer from discontinuity and ambiguity in deep learning applications. Two common approaches are typically employed to mitigate the discontinuity for learning a rotation loss. The first one is converting the Euler angle into a rotation matrix and using the Frobenius distance for obtaining the loss such as $||\mathbf{I}_{3\times3} - \mathbf{R}^{\mathrm{T}}\mathbf{R}||_F$. Another approach utilizes normalized quaternions for presenting a rotation matrix. Both expressions are still discontinuous. The rotation expression in [137] introduces a 6D representation of the 3D rotation matrix to remedy the problem of discontinuity, which however increase the complexity. Importantly, all approaches mentioned above consider rotation and translation separately. For normal rotation classification, the separation can simplify the loss function. However, in grasp planning, the rotation matrix and translation vector should be coupled. Therefore, I use Lie algebra for overcoming the discontinuity and ambiguity issue. I consider the grasped object to be composed of a set of charts and I define each grasp configuration at one chart, illustrated in Fig. 8.3d. Based on the manifold theory, or more specifically Lie group theory, SE(3) is a continuous group. Furthermore, SE(3) is a differentiable manifold. SE(3)-TrackNet [179] can be referred to for additional details regarding the benefits of using Lie algebra to train a SE(3) neural network.

As shown in Section 8.3.1, I use Lie algebra $\mathfrak{se}(3)$ for representing a transformation matrix with $\mathbf{v} = [\mathbf{t}^{\mathrm{T}}, \omega^{\mathrm{T}}]^{\mathrm{T}} \in \mathcal{R}^6$. According to the Lie algebra theory [10, 156], the exponential map, which maps elements from the algebra to the manifold and determines the local structure of the manifold, is used to express the transformation matrix as $e^{\mathbf{v}} = \begin{bmatrix} e^{\omega^\wedge}, & \mathbf{Vt} \\ \mathbf{0}, & 1 \end{bmatrix}$, where $\omega^\wedge$ is the skew-symmetric matrix of ω, $e^{\omega^\wedge}$ is defined as $\mathbf{I} + \frac{\sin\theta}{\theta}\omega^\wedge + \frac{1-\cos\theta}{\theta^2}(\omega^\wedge)^2$, and $\mathbf{V} = \mathbf{I}_{3\times3} + \frac{1-\cos\theta}{\theta^2}(\omega^\wedge) + \frac{\theta-\sin\theta}{\theta^3}(\omega^\wedge)^2$. The scale value θ is the norm of ω. It can be shown that using the exponential map, I can easily convert the 6-dimensional vector coordinate to SE(3). In the same way, I can convert SE(3) to $\mathfrak{se}(3)$ using the logarithm map with $\omega = (\mathrm{Log}(\mathbf{R}))^\vee$ and $\mathbf{t} = \mathbf{V}^{-1}$trans. The operator $(\vee)$ converts a skew-symmetric matrix to a vector, which is reciprocal to the operator $(\wedge)$. In this work, the proposed network produces a Lie algebra $\mathfrak{se}(3)$ value for each selected grasp point, denoted as $\mathbf{v}_{j,\mathfrak{se}(3)}$. Using the same strategy introduced in

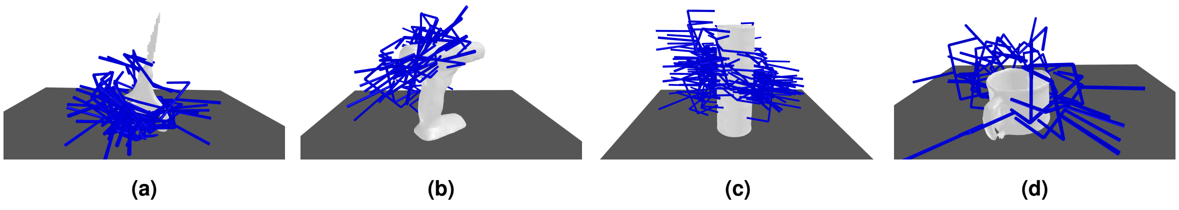


Figure 8.4: Visualization of results from the network after collision checking. The blue lines are the grasp configurations generated by the LiePFormer-GraspNet. The meshes are some examples from the YCB data set.

Section 8.3.3, I can obtain the corresponding ground truth grasp configuration, indicated as $\mathbf{T}_{j,g}$. In addition, I add the translation $\mathbf{q}_j - \mathbf{q}_g$ to $\mathbf{T}_{j,g}$, indicated as $\mathbf{T}_{j,g'}$ (Fig. 8.3c).

The logarithm map and group operation are employed to define the geodesic distance as Euclidean norm by

$$p\left(\mathrm{M}_1, \mathrm{M}_2\right) = \left\|\log\left(\mathrm{M}_1^{-1}\mathrm{M}_2\right)\right\| \tag{8.2}$$

where $M_i \in$ Lie group, defined on a differentiable manifold. The sum of the squared geodesic distance, is given as

$$L = \sum_{i=1}^{N} \rho^2\left(\boldsymbol{\beta}\left(f_i\right), \mathrm{M}_i\right) \tag{8.3}$$

The regression function $\boldsymbol{\beta} : R^d \longmapsto G$ estimates the element M on the matrix Lie group G for a given $d-$dimensional feature vector f as $M = \boldsymbol{\beta}(f)$.

The Baker-Campbell-Hausdorff (BCH) [144] formula formulate the logarithm $\log\left(e^{\mathrm{M}_1} e^{\mathrm{M}_2}\right)$ as a Lie algebra element using only Lie bracket for noncommutative Lie groups. A first order approximation to the BCH [133] is

$$\log\left(e^{\mathrm{M}_1} e^{\mathrm{M}_2}\right) = \mathrm{M}_1 + \mathrm{M}_2 + \frac{1}{2}\left[\mathrm{M}_1, \mathrm{M}_2\right] + O\left(\mathrm{M}_1^2, \mathrm{M}_2^2\right) \tag{8.4}$$

where M_1 and M_2 are $\mathrm{m}_1 = \log(\mathrm{M}_1)$ and $\mathrm{m}_2 = \log(\mathrm{M}_2)$, respectively. The geodesic distance can be approximated by

$$\begin{aligned} \rho\left(\mathrm{M}_1, \mathrm{M}_2\right) &= \left\|\log\left(\mathrm{M}_1^{-1}\mathrm{M}_2\right)\right\| = \left\|\log\left[e^{-\mathrm{m}_1} e^{\mathrm{m}_2}\right]\right\| \\ &= \left\|\mathrm{m}_2 - \mathrm{m}_1 + 0.5\left[-\mathrm{m}_1,\ \mathrm{m}_2\right] + O\left(\ \mathrm{m}_1^2,\ \mathrm{m}_2^2\right)\right\| \\ &\approx \left\|\mathrm{m}_2 - \mathrm{m}_1\right\| \end{aligned} \tag{8.5}$$

Therefore, the sum of squared geodesic distance can be approximated as

$$\begin{aligned} L_{\mathrm{se}(3)} &\approx \sum_{i=1}^{N} \left\|\log\left(\mathrm{M}_i\right) - \log\left(\beta\left(f_i\right)\right)\right\|^2 \\ &= \frac{1}{|S|} \sum_{0}^{j} \left\|\mathbf{v}_{j,\mathfrak{se}(3)} - \log(\mathbf{T}_{j,g'})\right\|^2 \end{aligned} \tag{8.6}$$

It can be easily proven that if $\mathbf{v}_{j,\mathfrak{se}(3)}$ is equal to $\log(\mathbf{T}_{j,g'})$, the predicted grasp configuration is aligned to ground truth. The approximation is sufficient for regression as long as the training samples lie in a small neighborhood of the identify.

Grasp planning Loss

Finally, I combine the grasp quality and transformation loss to form a grasp loss with weighting coefficients as

$$L_{\mathrm{total}} = w_1 L_{\mathrm{score}} + w_2 L_{\mathfrak{se}(3)}\,. \tag{8.7}$$

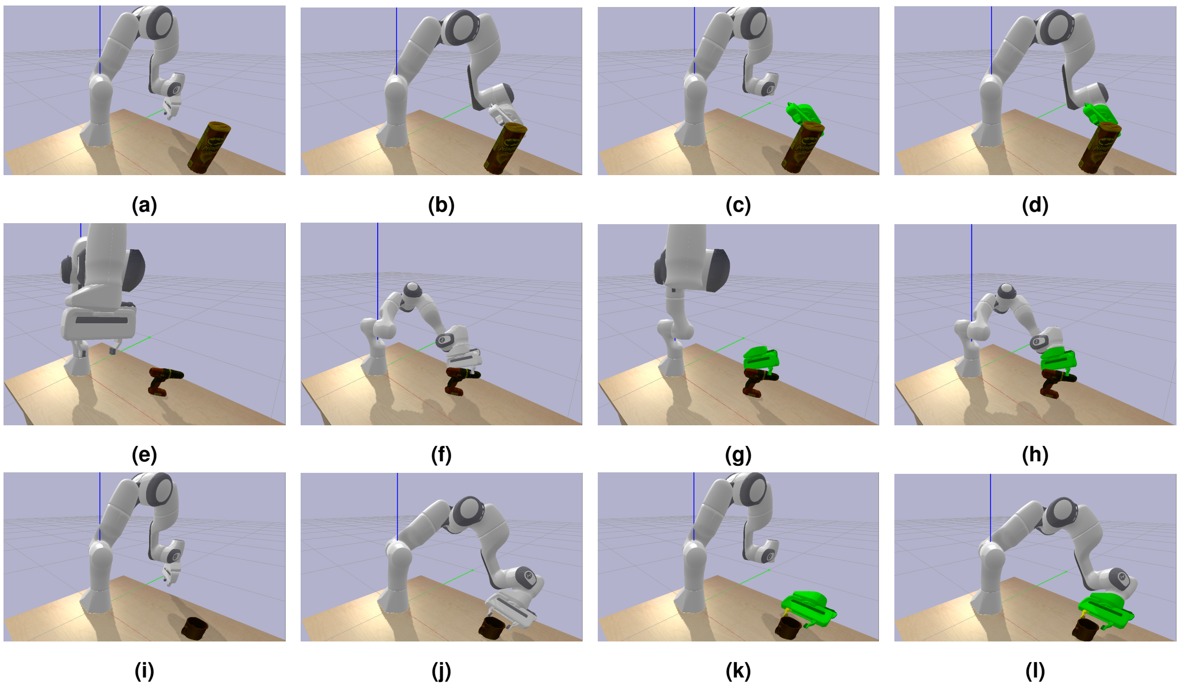


Figure 8.5: PyBullet simulation platform for grasping the objects. The process consists of two steps: in the first step, a grasp configuration at the palm with respect to the point cloud is generated by LiePFormer-GraspNet. Then, the end effector of a robotic manipulator is aligned to the palm pose of the grasp configuration. An inverse kinematics solver is applied for computing the joint configuration. In the end, a point to point motion is executed for grasping the object.

8.3.4 Grasp Configuration Selection and Refinement

LiePFormer-GraspNet will generate various grasp configurations at the inference phase by feeding a point cloud, which can be obtained via a sensor.I first need to prune all infeasible grasps by collision checking [122].Since a higher grasp quality shows a more reliable grasp for the object, I arrange the grasp quality in descent order and select the grasp configuration with the highest grasp quality. My grasp configuration is agnostic to the used parallel gripper. Therefore, at the inference phase, I need to adapt the grasp configuration with respect to the chosen gripper setting. In this work, I use the Franka Emika Panda gripper to evaluate the grasp. The blue gripper in Fig. 8.3a shows the simplified version, while the gray gripper shows the collision mesh. As shown in [38], the predicted grasp configuration is denoted as the pre-grasp, which might lead to a stable grasp when closing the finger. Therefore, I can further refine the grasp configuration based on that pre-grasp by approaching the gripper in the direction of the approach direction. The grasp approach direction is defined in the direction along the gripper handle, as shown in Fig. 8.3e. The approaching process is terminated when the gripper is in contact with the object. More details about this approach strategy can be found in the Graspit! platform [108].

8.4 Experimental Results

8.4.1 Experimental Settings

Table 8.1: Results (in %) of a single object grasping experiment. Success is defined as at least one feasible grasp configuration for each object. My LiePFormer-GraspNet can outperform other state-of-the-art approaches.

Method	Avg.	tomato soup can	pudding box	potted meat can	orange	plum	scissors	e-cups
GPD [124]	86.3	81.6	92.9	78.8	74.8	92.6	97.7	85.8
PointNetGPD [92]	86.0	75.0	94.1	68.6	80.1	94.3	98.4	90.8
LiePFormer-GraspNet (Ours)	94.8	90.0	100.0	96.3	92.3	98.0	100.0	96.8

Table 8.2: Ablation study of skip-connection transformer encoder in LiePFormer-GraspNet (in %).

Method	Avg.	tomato soup can	pudding box	potted meat can	orange	plum	scissors	e-cups
LiePFormer-GraspNet (without transformer)	92.7	85.3	100.0	92.3	85.3	93.0	100.0	92.8
LiePFormer-GraspNet	94.8	90.0	100.0	96.3	92.3	98.0	100.0	96.8

Grasping Data Set

I use the large-scale grasp data set ACRONYM [38, 166] for training and evaluate it with the YCB data set [20]. ACRONYM is a grasp data set for robot grasp planning based on the physics simulation FleX, which contains 17.7M parallel-jaw grasps, spanning 8872 objects from 262 different categories, each labeled with the grasp result obtained from a physics simulator. The objects from the ACRONYM data set [38] are from ShapeNetSem [153], with the assumption of uniform density and identical friction. I extend the ACRONYM data set by assigning each grasp configuration with a deterministic grasp point. Unlike most state-of-the-art approaches, that train and evaluate the network with the same data set, I use the YCB data set for evaluation. The YCB data set includes objects from daily life, with different shapes, sizes, textures, weights, and rigidities. YCB data set aims at evaluating the generalization of proposed grasp detection network.

Grasping Tasks

The experimental environment is set up inside the PyBullet [27], which contains a Franka Emika robotic manipulator, a Franka Emika parallel gripper, a table, the objects to grasp, and one depth camera. I randomly place a single YCB object on the table by considering the reachability of the robotic manipulator. A depth camera is applied to extract the point cloud for the application using the projection and view matrices. I perform the experiments in two steps. In the first step, I infer the grasp configuration by feeding the point cloud and then apply inverse kinematics to get the joint values for the robotic manipulator. I close the fingers to grasp the object finally.

Baselines

I compare my approaches against two open-sourced baselines: GPD [124] and PointNet-GPD [92]. GPD uses a geometry sampler to sample many grasp candidates and then uses an adapted CAN to evaluate the grasp configuration in terms of grasp score. PointNetGPD replaces the geometry sampler with PointNet++ to learn a grasp quality network. In this work, I follow the default network settings from the original papers.

8.4.2 Quantitative Results

I demonstrate the results for generating the grasp configurations in terms of success rate in Table 8.1, where the success rate is the percentage of successful grasps for grasping a single

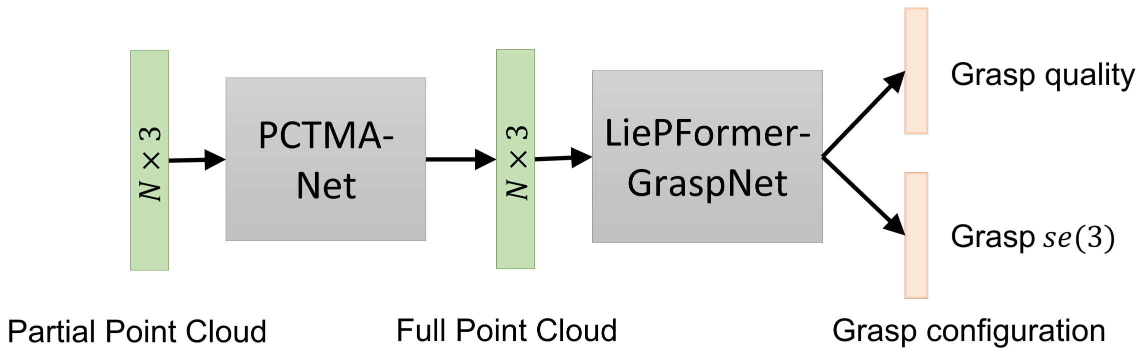


Figure 8.6: The overview of point cloud completion-based grasp planning.

object. The experiment is conducted by randomly translating each object and rotating in the z orientation. The objects vary from big components (pudding box) to small objects (plum). I trained my network with the ACRONYM data set, evaluated it with the whole YCB data set, and summarized part of the results in Table 8.1. From the results, I can conclude that my approach can be generalized to an unseen object, which is not contained in the ACRONYM.

8.4.3 Qualitative Results

In this section, I demonstrate the results generated from my proposed LiePFormer-GraspNet. The results are qualitatively illustrated in Fig. 8.4. LiePFormer-GraspNet will first generate 256 grasp configurations and their grasp qualities, where the number of 256 is a hyperparameter, which can be changed in terms of the GPU memory. Then, the generated grasps will be pruned via collision checking. By the grasp refinement phase, the object is assumed to be placed on a table. The grasp configurations are selected based on their grasp quality and their collision-free property. Furthermore, I demonstrate the results in a PyBullet environment (Fig. 8.5) with different initial robot joint configurations. The execution of a grasp configuration is composed of two steps: in the first step, I select the collision-free grasp configuration with the highest graps quality from my LiePFormer-GraspNet. Then, an inverse kinematics solver is applied for getting the joint configuration. From the qualitative results (Fig.8.4 and 8.5), I can conclude that my proposed LiePFormer-GraspNet can generate diverse grasp configurations for satisfying the various grasp initial configurations, which enables the application of motion planning.

8.4.4 Ablation Study

The skip-connection point transformer encoder block is integrated into the PointNet++ structure for extracting more attractive features for grasping. This section wants to study the effect of the transformer encoder block by using pure PointNet++ as the main skeleton. I compare the results in Table 8.2. The results show that the introduced modification of PointNet++ can help the LiePFormer-GraspNet better regress performance by extracting more valuable features.

8.4.5 Proposed future work

In this work, I assumed I obtain the full view of point cloud object representations, which, however, have only partial point cloud representation due to the single camera in the reality. In my previous work, I proposed a point transformer-based point cloud completion neural work [99], which aims at completing the partial point cloud. Incorporating the point cloud completion neural network and my proposed grasping network will be more attractive and enable more practical applications.

8.5 Conclusion

This work proposes a new grasp detection neural network based on PointNet++ with a hierarchically skip-connection transformer encoder. Two different grasp loss functions, a Gaussian distribution-based quality score loss and a grasp Transform Loss in terms of Lie algebra, are introduced for remedying the discontinuity problem due to rotation loss in form of Euler angles and quality score loss in form of an indicator function. My network is trained with the synthetic data set ACRONYM [38] and also works well in the real-world YCB data set. Furthermore, the experimental results show that my framework can detect diverse grasps with a higher converge on the ground truth grasps and that it can generalize to unknown objects, as the YCB data set [20] is different from ACRONYM [38].

Part V

Conclusion

Chapter 9
Conclusions and Future Work

Until now, most factories are not yet geared to respond quickly to change, adapt promptly to production demands and customer needs, or to quality problems that arise since robotics in the factory are mainly manually configured, calibrated, and managed. Moreover, it requires a great deal of expertise, which significantly limits reproducibility and scale. Besides, the classic methods of instructing robotic programming in the teach-in method are no longer flexible and approaching their limits when facing a complex requirement. Hence, I can conclude that implementing flexible manufacturing established on existing factory infrastructure is costly and time-consuming. An intelligent robotic system, which includes new programming paradigms with advanced machine learning technology and integrates sensor devices to handle new object classes, is an alternative to enable smart and flexible manufacturing. This system directly contrasts the status quo manufacturing has endured for the past 30 years. Furthermore, the rapid advances in artificial intelligence allow virtually mapping plants, processes, and products into a digital twin environment for better quality control. The primary intention of the intelligent robotic system is not to replace human workers. Instead, by reasonably distributing the work of humans and robots, the burden of human workers can be reduced, and unnecessary injuries can be avoided, especially in the human-robot interaction scenario. The whole article focuses on the theme that is how to make robots intelligent, flexible, and practical to adapt to the high-speed changing demand production environment. In this work, I proposed my first step toward the goal to build the intelligent robotic system from the perspective of vision, trajectory planning, and grasp planning. However, several concepts still need to be investigated in greater depth, and many extensions to this work are possible.

The off-the-shelf motion planning algorithm generates a collision-free path that avoids the manually hand-crafted waypoints in the joint space. The path is presented in the form of position without assigning the time of law. Kinodynamic planning improves the pure geometry motion planning by considering the kinematic constraints. However, this approach is proven to be an NP-hard problem and is limited to the velocity constraint. Therefore, I focus on my first effort on trajectory planning to create a smooth trajectory for multiple waypoints. My solution is based on a nonlinear-constraints optimization framework by connecting every two waypoints using the trapezoidal acceleration profile considering kinematic constraints. One key observation is that the standard trajectory module offered by the robotic manufacturer mainly considers the acceleration constraints and uses the spline-based or polynomial-based approach to interpolate or approximate trajectory. I have proposed a passing-through trajectory algorithm that presents several advantages over the state-of-art. Firstly, it supports the jerk constraint. Secondly, it performs better over the most state of the art approach in the straight-line criteria. I have evaluated the proposed algorithm on several examples to prove the importance of those features. I developed an extension version by blending the waypoints to avoid overshooting. The new algorithm exhibits better straight-line performance. The complexity of optimizing trapezoidal acceleration-based trajectory grows exponentially with increasing geometrical waypoints. I innovatively introduced the model predictive control-based optimization strategy by iteratively optimizing the subproblems. Each subproblem is initialized with the previous optimization step to reduce the complexity. As a consequence, the complexity linearly depends on the number of waypoints.

I revisited classic computer vision topics such as object detection, pose estimation, and shape completion and developed my algorithms to complement the state-of-the-art algorithms. In the field of pose estimation, the iterative closest point (ICP) algorithm is among the most famous and commonly used algorithms. The corresponding point search and initializing step are the primary factors affecting the performance, which causes the optimization to easily be stuck into the local point, even worse, stuck in the saddle point. In this work, I proposed to describe the object with the Gaussian Process Implicit Surface (GPIS). Therefore the pose estimation problem can be converted to the function fitting problem. To the best of my knowledge, this is the first work using the GPIS to solve the pose estimation problem. The GPIS function exploits the properties of the signed distance function to build three different manifolds layers, which avoids the corresponding search step and assumes that the points on the same object share the same function value. In the end, the pose estimation problem is finally reformulated as a manifold optimization problem. Through several experiments, I show that compared to a standard ICP based approach, my GPIS based pose estimation approach achieves comparable accuracy with significantly fewer optimization steps and exhibits rotation and translation invariant properties. The object classification problem is also addressed in this work. Due to the inherent problem of point cloud-based deep learning, the rotation invariant property can only be partially satisfied by means of vast amounts of data, which is not applicable in reality. The Fourier transform function decomposes functions depending on space or time into functions depending on spatial or temporal frequency. In a similar vein, each function defined on a sphere's surface can be reformulated as a sum of these spherical harmonics. This work proposed a new object descriptor by representing the point cloud object into the spherical harmonic energy function, which is proved to share the rotation invariant property. Furthermore, I introduced a point graph inter-geometry descriptor to strengthen the proposed spherical harmonic energy descriptor. In contrast to the most spherical harmonic neural network, which introduces additional operators to improve performance, my proposed descriptor is intuitive and straightforwardly applied in the existing network without further modification. The effectiveness of the presented object rotation-invariant descriptor has been validated via several experiments. The shape completion is the last computer vision topic addressed in this work, which attempts at completing the partial view of the point cloud due to the limited angle view. I view the shape completion problem from a different and uncommon perspective. Like the pose estimation algorithm, I assume each object has its corresponding manifold. I developed a PCTMA-Net in which a transformer-based neural network is be exploited to extract a high-dimensional feature of the partial point cloud. From another dimension, the extracted feature will be remapped to the manifold for reconstructing the full view of the point cloud. The PCTMA-Net can be further applied to grasp planning for widening solution space.

Grasp planning is a critical ingredient in the framework of the intelligent robotic system. I proposed two algorithms to enable proper grasping when facing an unknown object. In the first approach, the incorporation of Bayesian optimization (BO) and alternating direction method of multipliers (ADMM) has been used for figuring out the 6D grasp configuration, where I use the Gaussian Process Implicit Surface (GPIS) to represent the grasped object. The object can be extracted from the real world using the 3D sensor or sampled from the CAD model, which is the standard data format in the industry. Furthermore, I separate the optimization of hand palm posture and grasp contact point based on the observation that the optimization of palm posture is independent of the contact point. Therefore, I can update the palm pose and contact point using the BO-ADMM solver in an iterative fashion. An end-to-end transformer-based neural network is proposed in the second algorithm to predict the 6D grasp planning by directly feeding with the raw point cloud. I evaluate these two algorithms with several experiments and demonstrate their effectiveness and efficiency.

The algorithms I presented in this thesis are the first step towards the goal of intelligent robotic systems. There are still several extensions that can be further investigated for future work. I firmly believe that my work is helpful in many robotic and computer vision applications and will bring some inspiration to this field. In the following subsection, some ideas of the direction for future work will be discussed.

9.1 Online trajectory adaption

I have presented two algorithms targeting generating a smooth trajectory that passes through or blends around the waypoints. It is sufficient for the industry application for repetitive work. In the future, the online trajectory adaption with the trapezoidal acceleration profile should be further investigated, especially in the human-robot interaction scenario. I have made slight progress in the work [98] by incorporating a high-dimensional polynomial function and seven-segment profile. It has already significantly reduced time expenditure for optimizing a smooth trajectory. However, it is still not sufficient for the real-time application.

9.2 Reducing the complexity of GPIS based pose estimation

The work I presented about the pose estimation with the Gaussian Process Implicit Surface function is an innovative work by replacing the corresponding point search step, which significantly enhances the optimization efficiency. However, the complexity of GPIS is known as $O(N^3)$, where N is the number of point clouds. Updating a covariance matrix is a significant factor that affects performance. There have already been some works that address the problem by using the sparse data set and reached substantial progress. However, the dataset used in those works typically exhibits no geometry meaning. A further adaption is required for remaining the geometry details.

9.3 Object classification in a more general scenes

I have presented my approach for object detection from point cloud data considering the rotation invariant feature. However, my focus has been on using a single object for object detection, and the spherical harmonic energy function and the point graph inter-geometry descriptor can not be easily directly applied to a scene that may contain multiple objects. A two-stage pipeline incorporating a scene segmentation neural network can be helpful to extend my proposed algorithms to a more general application scenario.

9.4 Combined partial point cloud completion and grasp planning

The point cloud completion neural network presented in this thesis has succeeded in realizing this work's primary goal. The secondary purpose of this neural network is to improve grasp planning. One way is to directly utilize the reconstructed full point cloud to predict a better grasp configuration. Another direction of future work in this area is to decompose the object into a series of primitive shapes, simplifying the grasp planning searching space and significantly increasing efficiency. Moreover, integrating the primitive shape constrain solver in the grasp planning pipeline explains the predicted grasp configuration.

Part VI

Bibliography And Appendix

Bibliography

[1] Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. “Learning representations and generative models for 3D point clouds”. In: *Proceedings of the International Conference on Machine Learning*. PMLR. 2018, pp. 40–49.

[2] Andrearczyk, V., Oreiller, V., Fageot, J., Montet, X., and Depeursinge, A. “Solid Spherical Energy (SSE) CNNs for Efficient 3D Medical Image Analysis”. In: *Proceedings of the Irish Machine Vision and Image Processing Conference* (2019), pp. 37–44. DOI: 10.21427/ccjb-2504.

[3] Ariafar, S., Coll-Font, J., Brooks, D., and Dy, J. “ADMMBO: Bayesian Optimization with Unknown Constraints using ADMM”. In: *Journal of Machine Learning Research* 20.123 (2019), pp. 1–26. URL: http://jmlr.org/papers/v20/18-227.html.

[4] Asada, H. and Kitagawa, M. “Kinematic analysis and planning for form closure grasps by robotic hands”. In: *Robotics and Computer-Integrated Manufacturing* 5.4 (1989). Special Issue Simulation Software for Robotics, pp. 293–299. ISSN: 0736-5845. DOI: 10.1016/0736-5845(89)90003-3.

[5] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. “The Quickhull Algorithm for Convex Hulls”. In: *ACM Trans. Math. Softw.* 22.4 (Dec. 1996), pp. 469–483. ISSN: 0098-3500. DOI: 10.1145/235815.235821.

[6] Barfoot, T. D. *State Estimation for Robotics*. Cambridge University Press, 2017.

[7] Barnett, E. and Gosselin, C. “A Bisection Algorithm for Time-Optimal Trajectory Planning Along Fully Specified Paths”. In: *IEEE Transactions on Robotics* 37.1 (Feb. 2021), pp. 1–15. DOI: 10.1109/TRO.2020.3010632.

[8] Biagiotti, L. and Melchiorri, C. *Trajectory planning for automatic machines and robots*. Springer, 2008.

[9] Bicchi, A. and Kumar, V. “Robotic grasping and contact: A review”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2000, pp. 348–353.

[10] Blanco Claraco, J. L. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Tech. rep. 012010. University of Malaga, Sept. 2010.

[11] *Blender – A 3D Modelling and Rendering Package*. Blender Foundation. 2019. URL: http://www.blender.org.

[12] Blum, M. and Riedmiller, M. A. “Optimization of Gaussian Process Hyperparameters using Rprop”. In: *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Apr. 2013, pp. 339–344.

[13] Bohg, J., Morales, A., Asfour, T., and Kragic, D. “Data-driven grasp synthesis–A survey”. In: *IEEE Transactions on Robotics* 30.2 (2013), pp. 289–309.

[14] Boor, D. *A practical guide to splines*. Springer, 1978.

[15] Borst, C., Fischer, M., and Hirzinger, G. “Grasping the dice by dicing the grasp”. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 4. 2003, 3692–3697 vol.3. DOI: 10.1109/IROS.2003.1249729.

[16] Borst, C., Fischer, M., and Hirzinger, G. “Grasp planning: How to choose a suitable task wrench space”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2004, pp. 319–325.

[17] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[18] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. “Spectral networks and locally connected networks on graphs”. In: *Proceedings of the International Conference on Learning Representations*. 2014.

[19] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age”. In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[20] Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. “Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set”. In: *IEEE Robotics Automation Magazine* 22.3 (2015), pp. 36–52.

[21] Cao, B., Dodds, G., and Irwin, G. “Constrained time-efficient and smooth cubic spline trajectory generation for industrial robots”. In: *IEE Proceedings-Control Theory and Applications* 144.5 (1997), pp. 467–475.

[22] Chang, A. X., Funkhouser, T. A., Guibas, L. J., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015). arXiv: 1512.03012.

[23] Chen, C., Li, G., Xu, R., Chen, T., Wang, M., and Lin, L. “ClusterNet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4994–5002.

[24] Ciocarlie, M. T. “Low-dimensional robotic grasping: Eigengrasp subspaces and optimized underactuation”. PhD thesis. Columbia University, 2010.

[25] Cohen, T. S. and Welling, M. “Transformation properties of learned visual representations”. In: *Proceedings of the International Conference on Learning Representations* (2015).

[26] Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. “Spherical CNNs”. In: *Proceedings of the International Conference on Learning Representations*. 2018.

[27] Coumans, E. and Bai, Y. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2021.

[28] Dai, A., Ruizhongtai Qi, C., and Nießner, M. “Shape completion using 3D-encoder-predictor CNNs and shape synthesis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5868–5877.

[29] Dam, E. B., Koch, M., and Lillholm, M. *Quaternions, Interpolation and Animation*. Tech. rep. MIT, 2000.

[30] Dantam, N. and Stilman, M. "Spherical parabolic blends for robot workspace trajectories". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA, Sept. 2014, pp. 3624–3629. DOI: 10.1109/iros.2014.6943070.

[31] De Boor, C., De Boor, C., Mathématicien, E.-U., De Boor, C., and De Boor, C. *A practical guide to splines*. Vol. 27. springer-verlag New York, 1978.

[32] Denavit, J. and Hartenberg, R. S. "A kinematic notation for lower-pair mechanisms based on matrices". In: *Journal of Applied Mechanics* (1955), pp. 215–221.

[33] Diebel, J. "Representing attitude: Euler angles, unit quaternions, and rotation vectors". In: *Matrix* 58.15-16 (2006), pp. 1–35.

[34] Donald, B., Xavier, P., Canny, J., and Reif, J. "Kinodynamic motion planning". In: *Journal of the ACM* 40.5 (Nov. 1993), pp. 1048–1066.

[35] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[36] Duchi, J., Hazan, E., and Singer, Y. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[37] Duchon, J. "Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces". In: *Constructive Theory of Functions of Several Variables*. Vol. 571. Lecture Notes in Mathematics. Springer, 2006, pp. 85–100.

[38] Eppner, C., Mousavian, A., and Fox, D. "ACRONYM: A Large-Scale Grasp Dataset Based on Simulation". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2021.

[39] Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. "Learning SO(3) equivariant representations with spherical CNNs". In: *Proceedings of the European Conference on Computer Vision*. 2018, pp. 52–68.

[40] Fan, H., Su, H., and Guibas, L. J. "A point set generation network for 3D object reconstruction from a single image". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[41] Farin, G. E. and Farin, G. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[42] Feldmar, J. and Ayache, N. "Rigid and Affine Registration of Smooth Surfaces Using Differential Properties". In: *Proc. of the European Conf. on Computer Vision*. 2005, pp. 397–406.

[43] Ferrari, C. and Canny, J. F. "Planning optimal grasps." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 1992, pp. 2290–2295.

[44] Fitzgibbon, A. W. "Robust Registration of 2D and 3D Point Sets". In: *Image and Vision Computing* 21.13-14 (2003), pp. 1145–1153.

[45] Frazier, P. I. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML].

[46] Gardner, J. R., Kusner, M. J., Xu, Z. E., Weinberger, K. Q., and Cunningham, J. P. "Bayesian Optimization with Inequality Constraints." In: *ICML*. Vol. 2014. 2014, pp. 937–945.

[47] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[48] Gilmore, R. *Lie Groups, Physics, and Geometry: An Introduction for Physicists, Engineers and Chemists*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511791390.

[49] Goldfeder, C., Ciocarlie, M., Dang, H., and Allen, P. K. “The columbia grasp database”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2009, pp. 1710–1716.

[50] Granger, S. and Pennec, X. “Multi-Scale EM-ICP: A Fast and Robust Approach for Surface Registration”. In: *Proc. of the European Conf. on Computer Vision*. 2002, pp. 418–432.

[51] Groueix, T., Fisher, M., Kim, V. G., Russell, B., and Aubry, M. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[52] Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. “PCT: Point cloud transformer”. In: *Computational Visual Media* 7 (2021), pp. 187–199.

[53] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., and Kwok, N. M. “A comprehensive performance evaluation of 3D local feature descriptors”. In: *International Journal of Computer Vision* 116.1 (2016), pp. 66–89.

[54] Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. “SEMANTIC3D.NET: A new large-scale point cloud classification benchmark”. In: *Proc. of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-1-W1. 2017, pp. 91–98.

[55] Han, X., Li, Z., Huang, H., Kalogerakis, E., and Yu, Y. “High-resolution shape completion using deep neural networks for global structure and local geometry inference”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 85–93.

[56] Haschke, R., Weitnauer, E., and Ritter, H. “On-line planning of time-optimal, jerk-limited trajectories”. In: *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2008, pp. 3248–3253.

[57] Hu, T., Han, Z., Shrivastava, A., and Zwicker, M. “Render4Completion: Synthesizing multi-view depth maps for 3D shape completion”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 4114–4122.

[58] Hu, Y.-P. and Sun, T.-C. “Moving a B-Spline Surface to a Curve—A Trimmed Surface Matching Algorithm”. In: *Computer-Aided Design* 29.6 (June 1997), pp. 449–455.

[59] Huang, Z., Yu, Y., Xu, J., Ni, F., and Le, X. “PF-Net: Point fractal network for 3D point cloud completion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7662–7670.

[60] Jaderberg, M., Simonyan, K., Zisserman, A., et al. “Spatial transformer networks”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 2017–2025.

[61] Jang, E., Vijayanarasimhan, S., Pastor, P., Ibarz, J., and Levine, S. “End-to-end learning of semantic grasping”. In: *arXiv preprint arXiv:1707.01932* (2017).

[62] Jeng, K.-Y., Liu, Y.-C., Liu, Z. Y., Wang, J.-W., Chang, Y.-L., Su, H.-T., and Hsu, W. H. *GDN: A Coarse-To-Fine (C2F) Representation for End-To-End 6-DoF Grasp Detection*. arXiv:2010.10695 [cs.RO]. 2020. arXiv: 2010.10695 [cs.RO].

[63] Jiang, Y., Moseson, S., and Saxena, A. “Efficient grasping from RGBD images: Learning using a new rectangle representation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2011, pp. 3304–3311.

[64] Johnson, S. G. *The NLopt nonlinear-optimization package*. 2018. URL: http://ab-initio.mit.edu/nlopt.

[65] Jones, D. R., Schonlau, M., and Welch, W. J. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* 13.4 (Nov. 1998), pp. 455–492. ISSN: 1573-2916.

[66] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation”. In: *arXiv preprint arXiv:1806.10293* (2018).

[67] Kanezaki, A., Matsushita, Y., and Nishida, Y. “RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5010–5019.

[68] Kazhdan, M. and Funkhouser, T. “Harmonic 3D shape matching”. In: *ACM SIGGRAPH Conference Abstracts and Applications*. 2002, pp. 191–191.

[69] Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. “Rotation invariant spherical harmonic representation of 3D shape descriptors”. In: *Proceedings of the Eurographics Symposiumon on Geometry Processing*. 2003, pp. 156–164.

[70] Kennedy J.; Eberhart, R. “Particle Swarm Optimization”. In: *Proceedings of IEEE International Conference on Neural Networks* (1995), pp. 1942–1948.

[71] Kingma, D. P. and Ba, J. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[72] Kirillov Jr, A. *An introduction to Lie groups and Lie algebras*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2008.

[73] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.

[74] Klema, V. and Laub, A. “The singular value decomposition: Its computation and some applications”. In: *IEEE Transactions on Automatic Control* 25.2 (1980), pp. 164–176. DOI: 10.1109/TAC.1980.1102314.

[75] Kondor, R., Lin, Z., and Trivedi, S. “Clebsch-Gordan Nets: a fully fourier space spherical convolutional neural network”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 10117–10126.

[76] Kraft, D. *A Software Package for Sequential Quadratic Programming*. Forschungsbericht DFVLR-FB–88-28. Cologne, Germany: Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln, 1988.

[77] Kröger, T. and Wahl, F. M. “Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events”. In: *IEEE Transactions on Robotics* 26.1 (2010), pp. 94–111. DOI: 10.1109/tro.2009.2035744.

[78] Krull, A., Brachmann, E., Michel, F., Ying Yang, M., Gumhold, S., and Rother, C. “Learning analysis-by-synthesis for 6D pose estimation in RGB-D images”. In: *Proc. of the IEEE Intl. Conf. on Computer Vision*. 2015, pp. 954–962.

[79] Kuhn H. W.; Tucker, A. W. “Nonlinear Programming”. In: *Berkeley Symposium on Mathematical Statistics and Probability* 2 (1951), pp. 481–492.

[80] Kumra, S. and Kanan, C. “Robotic grasp detection using deep convolutional neural networks”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017, pp. 769–776.

[81] Kunz, T. and Stilman, M. “Time-optimal trajectory generation for path following with bounded acceleration and velocity”. In: *Proceedings of Robotics: Science and Systems* (July 2012).

[82] Kushner, H. J. “A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise”. In: *Journal of Basic Engineering* 86.1 (Mar. 1964), pp. 97–106. ISSN: 0021-9223.

[83] Lai, K. and Fox, D. “Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation”. In: *The International Journal of Robotics Research* 29.8 (2010), pp. 1019–1037.

[84] Lange, F. and Albu-Schäffer, A. “Path-Accurate Online Trajectory Generation for Jerk-Limited Industrial Robots”. In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 82–89. DOI: 10.1109/LRA.2015.2506899.

[85] Latombe, J.-C. *Robot motion planning*. Vol. 124. Springer Science & Business Media, 2012.

[86] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[87] Lenz, I., Lee, H., and Saxena, A. “Deep learning for detecting robotic grasps”. In: *The International Journal of Robotics Research* 34.4–5 (2015), pp. 705–724.

[88] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436.

[89] Li, J., Chen, B. M., and Hee Lee, G. “SO-Net: Self-organizing network for point cloud analysis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9397–9406.

[90] Li, M., Hang, K., Kragic, D., and Billard, A. “Dexterous grasping under shape uncertainty”. In: *Robotics and Autonomous Systems* 75 (Jan. 2016), pp. 352–364.

[91] Li, Q., Xiong, R., and Vidal-Calleja, T. “A GMM Based Uncertainty Model for Point Clouds Registration”. In: *Robotics and Autonomous Systems* 91 (2017), pp. 349–362.

[92] Liang, H., Ma, X., Li, S., Görner, M., Tang, S., Fang, B., Sun, F., and Zhang, J. “Pointnetgpd: Detecting grasp configurations from point sets”. In: *Proceedings of the International Conference on Robotics and Automation*. 2019, pp. 3629–3635.

[93] Liang, J., Makoviychuk, V., Handa, A., Chentanez, N., Macklin, M., and Fox, D. “GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning”. In: *CoRL*. 2018.

[94] Lin, C., Chang, P., and Luh, J. “Formulation and optimization of cubic polynomial joint trajectories for industrial robots”. In: *IEEE Transactions on automatic control* 28.12 (1983), pp. 1066–1074.

[95] Lin, J., Rickert, M., and Knoll, A. “6D Pose Estimation for Flexible Production with Small Lot Sizes based on CAD Models using Gaussian Process Implicit Surfaces”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[96] Lin, J., Rickert, M., and Knoll, A. “Deep Hierarchical Rotation Invariance Learning with Exact Geometry Feature Representation for Point Cloud Classification”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9529–9535. DOI: 10.1109/ICRA48506.2021.9561307.

[97] Lin, J., Rickert, M., and Knoll, A. “Grasp Planning for Flexible Production with Small Lot Sizes based on CAD models using GPIS and Bayesian Optimization”. In: *Proceedings of the IEEE International Conference on Automation Science and Engineering*. 2021.

[98] Lin, J., Rickert, M., and Knoll, A. “Parameterizable and Jerk-Limited Trajectories with Blending for Robot Motion Planning and Spherical Cartesian Waypoints”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 13982–13988. DOI: 10.1109/ICRA48506.2021.9561601.

[99] Lin, J., Rickert, M., Perzylo, A., and Knoll, A. “PCTMA-Net: Point Cloud Transformer with Morphing Atlas-based Point Generation Network for Dense Point Cloud Completion”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2021.

[100] Lin, J., Somani, N., Hu, B., Rickert, M., and Knoll, A. “An Efficient and Time-Optimal Trajectory Generation Approach for Waypoints under Kinematic Constraints and Error Bounds”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Madrid, Spain, Oct. 2018, pp. 5869–5876.

[101] Liu, S. and Carpin, S. “A fast algorithm for grasp quality evaluation using the object wrench space”. In: *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. 2015, pp. 558–563. DOI: 10.1109/CoASE.2015.7294138.

[102] Macfarlane, S. and Croft, E. A. “Jerk-bounded manipulator trajectory planning: Design for real-time applications”. In: *IEEE Transactions on Robotics and Automation* 19.1 (2003), pp. 42–52. DOI: 10.1109/tra.2002.807548.

[103] Mahler, J., Pokorny, F. T., Hou, B., Roderick, M., Laskey, M., Aubry, M., Kohlhoff, K., Kröger, T., Kuffner, J., and Goldberg, K. “Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2016, pp. 1957–1964.

[104] Mark, W. S. and Mathukumalli, V. *Dynamics and Control of Robot Manipulators*. Wiley, 1989. ISBN: 047161243X.

[105] Maturana, D. and Scherer, S. “VoxNet: A 3D convolutional neural network for real-time object recognition”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2015, pp. 922–928.

[106] McKinley, S. and Levine, M. “Cubic spline interpolation”. In: *College of the Redwoods* 45.1 (1998), pp. 1049–1060.

[107] Miller, A. T., Knoop, S., Christensen, H. I., and Allen, P. K. “Automatic grasp planning using shape primitives”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2003, pp. 1824–1829.

[108] Miller, A. T. and Allen, P. K. “Graspit! A versatile simulator for robotic grasping”. In: *IEEE Robotics and Automation Magazine* 11 (2004), pp. 110–122.

[109] Mishra, B., Schwartz, J., and Sharir, M. “On the existence and synthesis of multifinger positive grips”. English (US). In: *Algorithmica* 2.1-4 (Nov. 1987), pp. 541–558. ISSN: 0178-4617. DOI: 10.1007/BF01840373.

[110] Mousavian, A., Eppner, C., and Fox, D. “6-DOF GraspNet: Variational grasp generation for object manipulation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2901–2910.

[111] Murali, A., Mousavian, A., Eppner, C., Paxton, C., and Fox, D. "6-DOF Grasping for Target-Driven Object Manipulation in Clutter". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2020, pp. 6232–6238.

[112] Murray, R. M., Li, Z., and Sastry, S. S. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[113] Myronenko, A. and Song, X. "Point Set Registration: Coherent Point Drift". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 32.12 (2010), pp. 2262–2275.

[114] Nagy, Á. and Vajk, I. "Sequential time-optimal path-tracking algorithm for robots". In: *IEEE Transactions on Robotics* 35.5 (Oct. 2019). DOI: 10.1109/tro.2019.2920090.

[115] Nakamura, Y., Hanafusa, H., and Yoshikawa, T. "Task-priority based redundancy control of robot manipulators". In: *The International Journal of Robotics Research* 6.2 (1987), pp. 3–15.

[116] Nguyen, V.-D. "Constructing force-closure grasps". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 1986, pp. 1368–1373.

[117] Ni, P., Zhang, W., Zhu, X., and Cao, Q. *PointNet++ Grasping: Learning an End-to-end Spatial Grasp Generation Algorithm from Sparse Point Clouds*. arXiv:2003.09644 [cs.RO]. 2020. arXiv: 2003.09644 [cs.RO].

[118] Nocedal, J. and Wright, S. J. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006. ISBN: 9780387303031. DOI: 10.1007/978-0-387-40065-5.

[119] Nogueira, J., Martinez-Cantin, R., Bernardino, A., and Jamone, L. "Unscented Bayesian optimization for safe robot grasping". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016, pp. 1967–1972.

[120] Nortje, C. R., Ward, W. O., Neuman, B. P., and Bai, L. "Spherical harmonics for surface parametrisation and remeshing". In: *Mathematical Problems in Engineering* 2015 (2015).

[121] Ott, C. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. Vol. 49. Springer, Jan. 2008. ISBN: 978-3-540-69253-9. DOI: 10.1007/978-3-540-69255-3.

[122] Pan, J., Chitta, S., and Manocha, D. "FCL: A general purpose library for collision and proximity queries". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2012, pp. 3859–3866.

[123] Papazov, C. and Burschka, D. "Stochastic optimization for Rigid Point Set Registration". In: *Proc. of the Intl. Symposium on Visual Computing*. 2009, pp. 1043–1054.

[124] Pas, A. ten, Gualtieri, M., Saenko, K., and Platt, R. "Grasp pose detection in point clouds". In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1455–1473.

[125] Pelossof, R., Miller, A., Allen, P., and Jebara, T. "An SVM learning approach to robotic grasping". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2004, pp. 3512–3518.

[126] Perzylo, A., Rickert, M., Kahl, B., Somani, N., Lehmann, C., Kuss, A., Profanter, S., Beck, A. B., Haage, M., Hansen, M. R., Nibe, M. T., Roa, M. A., Sornmo, O., Robertz, S. G., Thomas, U., Veiga, G., Topp, E. A., Kesslar, I., and Danzer, M. "SMErobotics: Smart Robots for Flexible Manufacturing". In: *IEEE Robotics and Automation Magazine* 26.1 (2019), pp. 78–90. DOI: 10.1109/MRA.2018.2879747.

[127] Perzylo, A., Somani, N., Rickert, M., and Knoll, A. “An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions”. In: *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. Hamburg, Germany, Sept. 2015, pp. 4197–4203. DOI: 10.1109/IROS.2015.7353971.

[128] Pham, H. and Pham, Q.-C. “A new approach to time-optimal path parameterization based on reachability analysis”. In: *IEEE Transactions on Robotics* 34.3 (June 2018), pp. 645–659.

[129] Picheny, V., Gramacy, R. B., Wild, S., and Digabel, S. L. “Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 1443–1451.

[130] Piegl, L. and Tiller, W. *The NURBS book*. Springer Science & Business Media, 1996.

[131] Pomerleau, F., Colas, F., and Siegwart, R. “A Review of Point Cloud Registration Algorithms for Mobile Robotics”. In: *Foundations and Trends in Robotics* 4.1 (2015), pp. 1–104.

[132] Pomerleau, F., Colas, F., Siegwart, R., and Magnenat, S. “Comparing ICP variants on Real-World Data Sets”. In: *Autonomous Robots* 34.3 (2013), pp. 133–148.

[133] Porikli, F. M. “Regression on Lie Groups and Its Application to Affine Motion Tracking”. In: 2016.

[134] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. “PointNet: Deep learning on point sets for 3D classification and segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[135] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. “Volumetric and multi-view CNNs for object classification on 3D data”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5648–5656.

[136] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Proceedings of the International Conference on Neural Information Processing Systems*. Long Beach, California, USA, 2017, pp. 5105–5114.

[137] Qin, Y., Chen, R., Zhu, H., Song, M., Xu, J., and Su, H. “S4G: Amodal Single-view Single-Shot SE(3) Grasp Detection in Cluttered Scenes”. In: *Proceedings of the Conference on Robot Learning*. Ed. by Kaelbling, L. P., Kragic, D., and Sugiura, K. Vol. 100. Proceedings of Machine Learning Research. 2020, pp. 53–65.

[138] Raguram, R., Frahm, J.-M., and Pollefeys, M. “A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus”. In: *Proc. of the European Conf. on Computer Vision*. 2008, pp. 500–513.

[139] Rasmussen, C. E. “Gaussian processes in machine learning”. In: *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[140] Reisert, M. and Burkhardt, H. “Using irreducible group representations for invariant 3D shape description”. In: *Proceedings of the Joint Pattern Recognition Symposium*. Springer. 2006, pp. 132–141.

[141] Ren, S., He, K., Girshick, R., and Sun, J. “Faster R-CNN: towards real-time object detection with region proposal networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2016), pp. 1137–1149.

[142] Rickert, M. and Gaschler, A. “Robotics Library: An Object-Oriented Approach to Robot Applications”. In: *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2017, pp. 733–740. DOI: 10.1109/IROS.2017.8202232.

[143] Riegler, G., Osman Ulusoy, A., and Geiger, A. “OctNet: Learning deep 3D representations at high resolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3577–3586.

[144] Rossmann, W. *Lie groups: an introduction through linear groups*. Vol. 5. Oxford University Press on Demand, 2006.

[145] Ruina, A. and Pratap, R. *Introduction to Statics and Dynamics*. Oxford University Press, 2015.

[146] Runge, C. “Über Empirische Funktionen Und Die Interpolation Zwischen äquidistanten Ordinaten”. In: *Zeitschrift für Mathematik und Physik* 46 (1901), pp. 224–243.

[147] Rusu, R. B. “Semantic 3D object maps for everyday manipulation in human living environments”. In: *KI-Künstliche Intelligenz* 24.4 (2010), pp. 345–348.

[148] Rusu, R. B., Blodow, N., and Beetz, M. “Fast Point Feature Histograms (FPFH) for 3D Registration”. In: *Proc. of the IEEE Intl. Conf. on Robotics and Automation*. 2009, pp. 3212–3217. DOI: 10.1109/robot.2009.5152473.

[149] Rusu, R. B. and Cousins, S. “3D is Here: Point Cloud Library (PCL)”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1–4.

[150] Rusu, R. B., Marton, Z. C., Blodow, N., and Beetz, M. “Persistent point feature histograms for 3D point clouds”. In: *Proceedings of the International Conference on Intelligent Autonomous Systems*. 2008.

[151] Saravanan, R., Ramabalan, S., and Balamurugan, C. “Evolutionary optimal trajectory planning for industrial robot with payload constraints”. In: *The International Journal of Advanced Manufacturing Technology* 38.11 (2008), pp. 1213–1226.

[152] Sarmad, M., Lee, H. J., and Kim, Y. M. “RL-GAN-Net: A reinforcement learning agent controlled GAN network for real-time point cloud shape completion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5898–5907.

[153] Savva, M., Chang, A. X., and Hanrahan, P. “Semantically-Enriched 3D Models for Common-Sense Knowledge”. In: *Proceedings of the CVPR Workshop on Functionality, Physics, Intentionality and Causality*. 2015.

[154] Sederberg, T. W., Anderson, D. C., and Goldman, R. N. “Implicit representation of parametric curves and surfaces”. In: *Computer Vision, Graphics, and Image Processing* 28.1 (1984), pp. 72–84.

[155] Segal, A., Haehnel, D., and Thrun, S. “Generalized-ICP”. In: *Proc. of Robotics: Science and Systems*. Seattle, USA, 2009.

[156] Selig, J. M. “Lie Groups and Lie Algebras in Robotics”. In: *Computational Noncommutative Algebra and Applications*. Ed. by Byrnes, J. Vol. 136. NATO Science Series II: Mathematics, Physics and Chemistry. Springer, 2004, pp. 101–125.

[157] Sertac Karaman, E. F. “Sampling-Based Algorithms for Optimal Motion Planning”. In: *The International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894.

[158] Shoemake, K. “Animating rotation with quaternion curves”. In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. 1985, pp. 245–254.

[159] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. *Modelling, planning and control*. Springer, 2017.

[160] Snoek, J., Larochelle, H., and Adams, R. P. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems*. Ed. by Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. Vol. 25. Curran Associates, Inc., 2012.

[161] Standardization, I. O. for. *Manipulating Industrial Robots, Performance Criteria and Related Test Methods*. International Standard 9283. International Organization for Standardization, Apr. 1998.

[162] Strandberg, M. and Wahlberg, B. “A method for grasp evaluation based on disturbance force rejection”. In: *IEEE Transactions on Robotics* 22.3 (2006), pp. 461–469. DOI: 10.1109/TRO.2006.870665.

[163] Stuelpnagel, J. “On the Parametrization of the Three-Dimensional Rotation Group”. In: *Siam Review* 6 (1964), pp. 422–430.

[164] Stutz, D. and Geiger, A. “Learning 3D shape completion under weak supervision”. In: *International Journal of Computer Vision* 128.5 (2020), pp. 1162–1181.

[165] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. “Multi-view convolutional neural networks for 3D shape recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 945–953.

[166] Sundermeyer, M., Mousavian, A., Triebel, R., and Fox, D. “Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2021.

[167] Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*. MIT press, 1998. DOI: 10.1109/tnn.1998.712192.

[168] Tchapmi, L. P., Kosaraju, V., Rezatofighi, H., Reid, I., and Savarese, S. “TopNet: Structural Point Cloud Decoder”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[169] Tenorth, M., Perzylo, A., Lafrenz, R., and Beetz, M. “The RoboEarth language: Representing and Exchanging Knowledge about Actions, Objects and Environments”. In: *Proc. of the IEEE Intl. Conf. on Robotics and Automation*. May 2012, pp. 1284–1289. DOI: 10.1109/ICRA.2012.6224812.

[170] Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. “Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds”. In: *arXiv preprint arXiv:1802.08219* (2018).

[171] Tieleman, T. and Hinton. “Lecture 6.5-rmsprop: divide the gradient by a running averageof its recent magnitude”. In: *Neural networks for machine learning* (2012).

[172] Todorov, E., Erez, T., and Tassa, Y. “Mujoco: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[173] Turk, G. and Mullins, B. *Large Geometric Models Archive*. https://www.cc.gatech.edu/projects/large_models/. Mar. 2019.

[174] Turk, G. and O’brien, J. F. “Variational implicit surfaces”. In: (1999).

[175] Varley, J., DeChant, C., Richardson, A., Ruales, J., and Allen, P. “Shape completion enabled robotic grasping”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 2442–2447.

[176] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[177] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. "Dynamic graph CNN for learning on point clouds". In: *ACM Transactions on Graphics* 38.5 (2019), pp. 1–12.

[178] Wedderburn, J. H. M. *Lectures on matrices*. Vol. 17. American Mathematical Soc., 1934.

[179] Wen, B., Mitash, C., Ren, B., and Bekris, K. E. "SE(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020, pp. 10367–10373. DOI: 10.1109/IROS45743.2020.9341314.

[180] Wen, X., Li, T., Han, Z., and Liu, Y.-S. "Point cloud completion by skip-attention network with hierarchical folding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1939–1948.

[181] Williams, O. and Fitzgibbon, A. "Gaussian Process Implicit Surfaces". In: *Proc. of the Workshop on Gaussian Processes in Practice*. Bletchley Park, UK, Apr. 2007.

[182] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. "Harmonic networks: Deep translation and rotation equivariance". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5028–5037.

[183] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. "3D ShapeNets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1912–1920.

[184] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes". In: *Proc. of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, USA, 2018.

[185] Xie, H., Yao, H., Zhou, S., Mao, J., Zhang, S., and Sun, W. "GRNet: Gridding residual network for dense point cloud completion". In: *European Conference on Computer Vision*. Springer. 2020, pp. 365–381.

[186] Yang, B., Wen, H., Wang, S., Clark, R., Markham, A., and Trigoni, N. "3D object reconstruction from a single depth view with adversarial learning". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 679–688.

[187] Yang, J., Li, H., Campbell, D., and Jia, Y. "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 38.11 (2016), pp. 2241–2254.

[188] Yang, Y., Feng, C., Shen, Y., and Tian, D. "FoldingNet: Point cloud auto-encoder via deep grid deformation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 206–215.

[189] Yoshikawa, T. "Analysys and control of robot manipulators with redundancy. Robotic Research". In: *The First International Syposium, 1984* (1984), pp. 735–747.

[190] Yoshikawa, T. "Manipulability of robotic mechanisms". In: *The international journal of Robotics Research* 4.2 (1985), pp. 3–9.

[191] Yu, T., Meng, J., and Yuan, J. "Multi-view harmonized bilinear network for 3D object recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 186–194.

[192] Yu, X. and Gen, M. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[193] Yuan, W., Khot, T., Held, D., Mertz, C., and Hebert, M. "PCN: Point Completion Network". In: *Proceedings of the International Conference on 3D Vision (3DV)*. 2018.

[194] Zhang, L. and Trinkle, J. C. "The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2012, pp. 3805–3812.

[195] Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. "Point transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16259–16268.

[196] Zhao, H. "A Fast Sweeping Method for Eikonal Equations". In: *Mathematics of Computation* 74.250 (2005), pp. 603–627.

[197] Zhou, Q.-Y., Park, J., and Koltun, V. "Open3D: A Modern Library for 3D Data Processing". In: (2018). arXiv: 1801.09847.

[198] Zhou, Y. and Tuzel, O. "VoxelNet: End-to-end learning for point cloud based 3D object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499.

List of Figures

List of Tables

Appendix A
Appendix

In this appendix, I give a brief review of some basic concepts from optimization and linear algebra. The treatment is by no means complete and is mainly meant to set out my notation.

A.1 Norms

The standard inner product for two vectors $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n$ is defined as

$$< \mathbf{x}, \mathbf{y} >= \mathbf{x}^{\mathrm{T}}\mathbf{y} = \sum_{i}^{n} x_i y_i \tag{A.1}$$

For a vector $\mathbf{x} \in \mathbb{R}^n$, the following norm is used

$$||\mathbf{x}||_1 \overset{\text{def}}{=} \sum_{i=1}^{n} |x_i| \tag{A.2a}$$

$$||\mathbf{x}||_2 \overset{\text{def}}{=} \left(\sum_{i=1}^{n} x_i^2\right)^{1/2} = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}\right)^{1/2} \tag{A.2b}$$

$$||\mathbf{x}||_\infty \overset{\text{def}}{=} \max_{i=1,\ldots,n} |x_i| \tag{A.2c}$$

The operator $||\cdot||_2$ is also denoted as Euclidean norm or L_2 norm, and I refer $||\cdot||_1$ as L_1 norm, and $||\cdot||_\infty$ as L_∞ norm. Similarly, the matrix norm $\mathbf{A}$ is defined as

$$||\mathbf{A}||_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} \left|\mathbf{A}_{ij}\right| \tag{A.3a}$$

$$||\mathbf{A}||_2 = \sqrt{\lambda_{\max}(\mathbf{A}^{\mathrm{T}}\mathbf{A})} \tag{A.3b}$$

$$||\mathbf{A}||_\infty = \max_{i=1,\ldots,m} \sum_{j=1}^{n} \left|\mathbf{A}_{ij}\right| \tag{A.3c}$$

The Frobenius norm of $\mathbf{A}$ has the following formal:

$$||\mathbf{A}||_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \left|\mathbf{A}_{ij}\right|^2} = \sqrt{\operatorname{trace}(\mathbf{A}^*\mathbf{A})} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)} \tag{A.4}$$

where $\sigma_i(\mathbf{A})$ are the singular values of $\mathbf{A}$, and the trace of a $n \times n$ matrix $\tilde{\mathbf{A}} = \mathbf{A}^*\mathbf{A}$ is formulated as

$$\operatorname{trace}(\tilde{\mathbf{A}}) = \sum_{i}^{n} \tilde{\mathbf{A}}_{ii} = \sum_{i}^{n} \lambda_i \tag{A.5}$$

where λ_i is the eigenvalue of the matrix $\tilde{\mathbf{A}}$. Furthermore, condition number of square non-singular matrix $\tilde{\mathbf{A}}$ is defined by

$$\mathrm{cond}(\tilde{\mathbf{A}}) = ||\tilde{\mathbf{A}}||||\tilde{\mathbf{A}}||^{-1} \tag{A.6}$$

The condition number measures the ratio of the maximum relative stretching to the maximum relative shrinking that the matrix does to any non-zero vectors. Therefore, cond($\tilde{\mathbf{A}}$) is ∞ indicating a singular matrix. A low condition number of a problem is said to be well-conditioned. In contrast, I say the problem is ill-conditioned.

A.2 Optimization in a Nutshell

Optimization may be regarded as the cornerstone of many areas of applied mathematics, computer science, engineering, and a number of other scientific disciplines. Mathematical optimization or mathematical programming is selecting the best element, concerning some criterion, from some set of available alternatives. Optimization modeling is a methodology to define an objective function together with the constraints for a given problem. An optimization problem calculates an objective function's extrema (maxima, minima, or stationary points) over a set of unknown real variables. It is conditional to the satisfaction of a system of equalities and inequalities, collectively termed constraints

- An objective function is a quantitative measure of the system's performance to minimize or maximize. For instance, the time-optimal or the minimal deviation of straight-line trajectory is the goal in the trajectory generation, whereas in fitting experimental data to a model such as a pose estimation or deep learning, I may want to reduce the total deviation of the observed data from the predicted data.
- The variables are the main components of the system. In trajectory generation, the variables may be the kinematic variables or trajectory traveling time, whereas, in the application of data science, the variables would be the model's parameters.
- The constraints are the functions that describe the relationships among the variables and define the allowable values for the variables. In trajectory generation, for example, the kinematic limitations cannot exceed.

The optimization problem can be mathematically formulated in the following as:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad \mathrm{f}(\mathbf{x}) \tag{A.7a}$$

$$\text{s.t} \quad \mathrm{h}_i(\mathbf{x}) = 0, i = 1\ldots, m \tag{A.7b}$$

$$\mathrm{g}_j(\mathbf{x}) \leq 0, j = 1\ldots p \tag{A.7c}$$

where x is the unknown variable, and f(x) is the objective function. The scalar function $\mathrm{g}_i(x)$ and $\mathrm{h}_j(x)$ is the equality, and inequality constraint function, respectively. The unconstrained optimization problem is a special case of (A.7) where $m = p = 0$. Furthermore, the optimization can be categorized as linear or nonlinear dependent on the linearity of the objective function and constraint function. There are many different ways to classify optimization problems. The nonlinear problem can be further divided into the convex and non-convex in terms of convexity property, where the concept of convexity is fundamental in the most practical applications. The function is denoted as a convex function if the following condition is satisfied

$$\mathrm{f}(\alpha x + (1-\alpha)y) \leq \alpha\,\mathrm{f}(x) + (1-\alpha)\,\mathrm{f}(y), \quad \text{for all } \alpha \in [0,1] \tag{A.8}$$

where x, y are two arbitrary points in the convex set S. A problem posses the convex property (A.8) can facilitate the optimization step in the theory and practice. Besides, the found local optimal solution is, in fact, also a global solution. However, the global solution in most cases is far tricky, and the analytical methods are typically not applicable. The numerical approaches can lead to a non-global solution or local solution. The main principle in the optimization algorithm is to determine how to move from the current iterate to the next for improving the performance in term of the direction and moving step length, which derives many different algorithms, such as line search, trust region. I can briefly formulate the line search algorithm in the following (A.9)

$$\min_{\alpha>0} \quad \mathrm{f}(x_k + \alpha p_k) \tag{A.9a}$$

$$x_{k+1} = x_k + \alpha p_k \tag{A.9b}$$

where α indicates the step length in the moving descent direction p_k. At each new point, a new search direction and step length are computed, and the process is repeated. In contrast to the line search algorithm, the trust region approximates the original objective function by constructing a quadratic function within a region

$$\min_{p} \quad m_k(x_k + p), s.t. \|p\|_2 < \triangle_k \tag{A.10a}$$

$$m_k\,(x_k + p) = \mathrm{f}_k + p^T \nabla\, \mathrm{f}_k + \frac{1}{2} p^T \mathbf{B}_k p \tag{A.10b}$$

where $\triangle_k$ is the trust region radius, and $\nabla\,\mathrm{f}_k$ is the gradient at current point, and $\mathbf{B}_k$ is the corresponding hessian matrix or its approximation matrix. The idea behind the trust-region algorithm is to find a proper region radius $\triangle_k$ so that the candidate solution produces a sufficient decrease in f. Otherwise, the radius should be shrunk.

A.2.1 Sequential Quadratic Programming

Most of the problems involved in this thesis are nonlinear and constraints. Sequential quadratic programming (SQP) is one of the most effective methods for nonlinearly constrained optimization. Literally, SQP is an optimization algorithm, which decomposes a problem into a sequence of optimization subproblems, and each subproblem is approximated as a quadratic model around the current iterate. Furthermore, the SQP can be integrated into the line search or trust region framework. Therefore, the main challenge is to formulate the quadratic subproblem to yield a good step for the nonlinear optimization problem. I can rewrite the (A.7) with the Lagrangian equation:

$$\mathcal{L}(x, \lambda, \sigma) = \mathrm{f}(\mathbf{x}) + \sum_{j}^{p} \lambda_j\, \mathrm{h}_j(\mathbf{x}) + \sum_{i}^{m} \sigma_i\, \mathrm{g}_i(\mathbf{x}) \tag{A.11}$$

where $\lambda \in \mathbb{R}^p$ and $\sigma \in \mathbb{R}^m$ are Lagrange multipliers. The first order necessary optimality or Karush-Kuhn-Tucker (KKT) [79] conditions requires the following condition

$$\nabla \mathcal{L}\,(\mathbf{x}^*, \lambda^*, \sigma^*) = \nabla\, \mathrm{f}\,(x^*) + \nabla\, \mathrm{h}\,(x^*)\,\lambda^* + \nabla\, \mathrm{g}\,(x^*)\,\sigma^* = 0 \tag{A.12}$$

The challenge of the SQP is to construct a quadratic programming subproblems at each iterate x_k. An appropriate search direction $\mathbf{p}_k = \mathbf{x} - \mathbf{x}_k$ is worked as a solution to this QP subprogram. I decompose the construction process into two ways. Firstly, an approximation around the objective function f with the Taylor series is executed:

$$\mathrm{f}(\mathbf{x}) = \mathrm{f}(\mathbf{x}_k) + \nabla\, \mathrm{f}(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^{\mathrm{T}} \nabla^2_{xx}\, \mathrm{f}(\mathbf{x} - \mathbf{x}_k) \tag{A.13}$$

and the constraint functions g and h by their local affine approximations:

$$\mathrm{g}(\mathbf{x}) = \mathrm{g}(\mathbf{x}_k) + \nabla\,\mathrm{g}(\mathbf{x} - \mathbf{x}_k) \tag{A.14a}$$
$$\mathrm{h}(\mathbf{x}) = \mathrm{h}(\mathbf{x}_k) + \nabla\,\mathrm{h}(\mathbf{x} - \mathbf{x}_k) \tag{A.14b}$$

Therefore the QP subprograms can be formulated as

$$\min_{\mathbf{p}} \quad \mathrm{f}(\mathbf{x}_k) + \nabla\,\mathrm{f}^{\mathrm{T}}\,\mathbf{p}_k + \frac{1}{2}\mathbf{p}_k^{\mathrm{T}}\nabla_{xx}\,\mathrm{f}\,\mathbf{p}_k \tag{A.15a}$$
$$\text{s.t.} \quad \mathrm{g}(\mathbf{x}_k) + \nabla\,\mathrm{g}^{\mathrm{T}}\,\mathbf{p}_k \leq 0 \tag{A.15b}$$
$$\mathrm{h}(\mathbf{x}_k) + \nabla\,\mathrm{h}^{\mathrm{T}}\,\mathbf{p}_k = 0 \tag{A.15c}$$

The formulation in (A.15) does not consider the KKT conditions or Newton-KKT condition. An alternative approach is to replace the Hessian matrix of objective function with the Hessian matrix of Lagrangian equation [118], and it results in the following formulation:

$$\min_{\mathbf{p}} \quad m_k = \mathrm{f}(\mathbf{x}_k) + \nabla\,\mathrm{f}(\mathbf{x}_k)^{\mathrm{T}}\,\mathbf{p} + \frac{1}{2}\mathbf{p}^T\nabla^2_{xx}\mathcal{L}(\mathbf{x}_k, \lambda_k, \sigma_k)\,\mathbf{p} \tag{A.16a}$$
$$\text{s.t.} \quad \mathrm{g}_i(\mathbf{x}_k) + \nabla\,\mathrm{g}_i(\mathbf{x}_k)^T\,\mathbf{p} \leq 0 \tag{A.16b}$$
$$\mathrm{h}_j(\mathbf{x}_k) + \nabla\,\mathrm{h}_j(\mathbf{x}_k)^T\,\mathbf{p} = 0 \tag{A.16c}$$

The candidate at the iterate x_k is given by x_{k+1}. The construction is repeated until the algorithm converges to a local minimum $x^\star$. As mentioned before, the SQP can be incorporated into the line search or trust-region framework by adding additional constraints in the subprogram, such as $\|\mathbf{p}_k\|_2 \leq \Delta_k$. There exist several challenges in the QP subproblem, such as the convexity of the m_k and the complexity of computing the Hessian matrix $\nabla^2_{xx}\mathcal{L}$. If the Hessian matrix $\nabla^2_{xx}\mathcal{L}$ is positive semi-definite, I can conclude that the (A.16) is a convex QP problem, which shares a similar complexity as the linear programming. In contrast, if the $\nabla^2_{xx}\mathcal{L}$ is an indefinite matrix, it leads to a nonconvex QP problem, which has several stationary points and local minima. The more detailed algorithm refers to [118].

A.2.2 Bayesian Optimization

The approximation for an objective function with the Taylor series can be challenge in some applications, such as data fitting, since the objective functions in those cases are either described as a black box model or too expensive to evaluate. Besides, the convex optimization-based approach can only converge to a local minimum. Finding a global solution for a nonconvex function is still a challenging problem. In the last decades, many researchers have proposed several heuristic approaches to find a better local minimum, such as multiple starting points. However, they can not be guaranteed to share the same performance in all cases. To further mitigate the problem, the concept of global optimization concerning a specific application area is introduced that attempts to find the global minima or maxima of a function or a set of functions on a given set. The global optimization approach using the random variables, denoted as stochastic optimitzation (SO), has attracted a lot of attention in the past years, which is broadly used in data-driven-based applications, such as Simulated annealing (SA) [73], Evolutionary algorithm (EA) [192], Particle swarm optimization (PSO) [70]. Bayesian optimization (BO) [45, 82, 160]. In this thesis, I use Bayesian optimization to find out grasp configurations. Therefore, I will briefly introduce this technology in this section.

Bayesian Optimization, abbreviated as (BO), is a machine learning based approach by employing the concept of Bayer theory and Gaussian process regression, and focusing on the following problem

$$\max_{\mathrm{x}\in A} \mathrm{f}(\mathbf{x}) \tag{A.17}$$

Algorithm 6 Structure of Bayesian Optimization

```
Require: $n_{\text{dim}}, n_{\text{iter}}$
Ensure: $\mathrm{f}_{\text{obj,min}}(\mathbf{x}^{+})$
 1: Get LHS $\mathcal{D}_{0:t-1} = \{(\mathbf{x}_0, y_0), \dots, (\mathbf{x}_{t-1}, y_{t-1})\}$
 2: Fit the Gaussian process Model $p\,(y|\mathbf{x}, \mathcal{D}_{0:t-1})$
 3: Optimized hyper Parameter
 4: for t = 1 to $n_{\text{iter}}$ do
 5:     $\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x}} \mathrm{EI}\,(\mathbf{x}|\mathcal{D}_{0:t-1})$
 6:     Observe a sample $y_t = \mathrm{f}_{\text{obj}}(\mathbf{x}_t) + \epsilon_t$
 7:     Add the new sample $\mathcal{D}_{0:t} = \{\mathcal{D}_{0:t-1}, (\mathbf{x}_t, y_t)\}$
 8:     update Gaussian process model
 9: end for
10: $\mathbf{x}^{+} = \operatorname{argmax}_{\mathbf{x}_i \in \mathbf{x}_{0:t}} \mathrm{f}_{\text{obj}}\,(\mathbf{x}_i)$
11: return $\mathbf{x}^{+}$
```

where A is described as the feasible set, and $\mathrm{f}(\mathbf{x})$ is the objective function, which could be "expensive to evaluate" or lack the known special structures, i.e., the convexity and linearity. Besides, It is best-suited for the Bayesian Optimization in the case that obtaining the first and second derivative of $\mathrm{f}(\mathbf{x})$ is not possible. Therefore, a surrogate for the objective function is used to approximate the behaviors in the local region. In comparison to the trust region approach, which limits the region radius and approximates the objective function with its first and second derivative, the surrogate function in the BO uses the Gaussian process regression function [139] to realize this purpose. The ability to optimize the derivative-free functions makes the BO extremely attractive. The naive structure of Bayesian Optimization is to be described in Alg 6. The BO is mainly built on three components: Sampling, Gaussian Process (GP) Regression, and acquisition function (EI). The sampling strategy is firstly used to explore the feasible set. Whereas the easiest way is to employ a random sample or grid-based sample. Then the Gaussian Process regression function utilizes these samples with their corresponding scores to approximate the objective function and acts as the surrogate function. In the next, The probabilistic information that resides in the model from the acquisition function can be used to determine which sample is the next worthing evaluating. The acquisition function has a much different representation includes the probability of improvement, expected improvement, upper confidence bounds (UCB), Knowledge Gradient, Entropy Search, and Predictive Entropy Search [160]. Maximizing a well-behaved acquisition function can be solved with a numerical optimization technique, such as Newton's Method or quasi-Newton methods. Therefore, the whole Bayesian optimization is described as solving the maximum problem of the acquisition function in an iterative fashion. In summary, Bayesian optimization provides a probabilistically principled tool to achieve a global solution. However, the approach mentioned above does not consider any additional constraints. Similar to the standard nonlinear programming approach, the augmented Lagrange multiplier [46, 129] can also be integrated into the Bayesian optimization framework for considering the equality and inequality constraints. The Bayesian optimization with the constraints is still a challenging problem. The corporate with ADMM [3, 97] in BO attracts a lot of attention in the machine learning application. In my work [97], I attempt to incorporate BO and ADMM for figuring the grasp configuration under consideration of the constraints.

A.2.3 The Optimization Algorithm in the Deep Learning Framework

In recent years, the optimization algorithm in machine learning, especially in the deep learning, achieved a considerable process. The optimization algorithm in deep learning allows a complex model to continuously update its parameters and minimize the loss function as it evaluates on the training dataset. Similar to the approaches mentioned earlier, an objective function for a particular task needs to be designed, and an optimization algorithm attempts to minimize the loss. It is known that almost all the model within the deep learning framework is nonconvex, and no analytical solution is available. There are many challenges in deep learning algorithms, such as the local minima, saddles point, and vanishing gradients. Among these challenges, the problem of vanishing gradients is the most insidious to encounter, which causes optimization to stall. In the reminder of this section, I will briefly introduce some advanced optimization technology within the deep learning framework.

The gradient descent (GD) based approach is fundamental for the all optimization approach to update the model's parameters. In generally, The batch gradient optimization (BGD) can be formally described as

$$\mathrm{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{f}_i(\mathbf{x}) \tag{A.18}$$

where $\mathbf{x}$ is denoted as the parameters in the model, the parameter n is the number of sample contained in a train dataset. The equation (A.18) is the averages of the loss functions for each example in the training dataset. Therefore the gradient of (A.18) can be straightforward derived as

$$\nabla\,\mathrm{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\nabla\,\mathrm{f}_i(\mathbf{x}) \tag{A.19}$$

The computational complexity in (A.19) is growing linearly depending on the number of a set $\mathcal{O}(n)$. It is well known that deep learning models crave for data, therefore, the number of n is typically a hug value. The concept of stochastic gradient descent approach (SGD) is used to improve the computational efficiency. The stochastic usage in the SGD embodies that it reduces the computational cost at each iteration by randomly sampling an index in $[1,\ldots,n]$ instead of using the whole gradient at once. Therefore the step update is reformulated as

$$\mathbf{x} \leftarrow \mathbf{x} - \eta\nabla\,\mathrm{f}_i(\mathbf{x}) \tag{A.20}$$

where η is the learning rate. As a result of SGD, the computational cost is reduced to $\mathcal{O}(1)$. The stochastic nature of gradient introduces much more noise than the average gradient descent algorithm. The uncertainty is affected by the instantaneous gradient $\eta\nabla\,\mathrm{f}_i(\mathbf{x})$ even though the minimum almost arrives. The performance can be improved by introducing the concept of Mini-Batch Gradient Descent, which is an balance between the batch gradient decent and stochastic gradient decent. The mini-batch gradient descent splits the training dataset into multiple small batches where each batch has a size of B, and update the model's parameter with

$$\nabla\,\mathrm{f}_B(\mathbf{x}) = \frac{1}{B}\sum_{i=1}^{B}\nabla\,\mathrm{f}_i(\mathbf{x}) \tag{A.21a}$$

$$\mathbf{x} \leftarrow \mathbf{x} - \eta\nabla\,\mathrm{f}_B(\mathbf{x}) \tag{A.21b}$$

The mini-batch gradient descent can facilitate the computation using the vectorized implementation.

The gradient-based approach highly depends on the gradient value, which shares significant benefits in the case of curvature or noisy gradient. Due to the saddle point in the optimization problem, the optimization step can be stuck. The physical concept of momentum allows to build the inertia in the search space and overcome the oscillation and across the flat points.The inertia indicates the encountered gradient in the previous updates that avoids the problem of the gradient descent approach, which bounces around the search space. Mathematically speaking, the momentum introduces an additional parameter for controlling the past parameter update information's influence on the current update. It can be formulated as

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \nabla \mathrm{f}(\mathbf{x}_t) \tag{A.22a}$$

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_{t+1} - \eta_t \nabla \mathrm{f}(\mathbf{x}_t) \tag{A.22b}$$

Note that $\beta = 0$ can convert the momentum approach to a standard decent gradient algorithm. The value $\nabla \mathrm{f}(\mathbf{x}_t)$ can be chosen as batch, stochastic or mini-batch gradient value.

Until now, I consider the learning rate η in the GD or momentum approach as a constant value, which is manually chosen. The boundary of the learning rate is difficult to statically determined. The algorithm Adagrad [36] is used to adapt the learning rate to the parameters based on the past gradients that have been computed

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \frac{\eta}{\sqrt{\mathrm{Diag}(G_t) + \epsilon}} \odot \nabla \mathrm{f}(\mathbf{x}_t) \tag{A.23a}$$

where the operator $\odot$ indicates the matrix-vector product, and ϵ is used to avoid the singularity. The value in the diagonal of matrix G_t is computed as

$$G_{t,ii} = G_{t-1,ii} + \nabla \mathrm{f}_i^2(\mathbf{x}_t) \tag{A.24}$$

The advantage of using the Adagrad is to eliminate the manual tuning step. However, the accumulation of the squared gradients of $G_{t,ii}$ reduces the learning rate. The learning rate will eventually converge to an infinitely small value, and no additional knowledge can be acquired in the end, which means the process will be stalled at some point. The concept of RMSProb [171] is introduced to mitigate the problem of vanishing the learning rate. The idea behind the RMSProb is to use the leaky average to normalized the the sum of weights of learning rate.

$$G_{t,ii} = \gamma G_{t-1,ii} + (1-\gamma)\nabla \mathrm{f}_i^2(\mathbf{x}_t) \tag{A.25a}$$

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \frac{\eta}{\sqrt{\mathrm{Diag}(G_t) + \epsilon}} \odot \nabla \mathrm{f}(\mathbf{x}_t) \tag{A.25b}$$

The coefficient $\gamma \in [0,1]$ determines how the history can affect the parameter update. Adaptive Moment Estimation (Adam) [71] is another approach to adapt the learning rate and momentum, which combines all the benefits of previous mentioned approaches into one effective optimization. It and has become more and more popular as one of the more robust and effective optimization algorithms. It can be formully described as

$$\mathbf{v}_t \leftarrow \beta_1 \mathbf{v}_{t-1} + (1-\beta_1) \nabla \mathrm{f}(\mathbf{x}_t) \tag{A.26a}$$

$$\mathbf{s}_t \leftarrow \beta_2 \mathbf{s}_{t-1} + (1-\beta_2) \nabla \mathrm{f}(\mathbf{x}_t)^2 \tag{A.26b}$$

Here β_1 and β_2 are nonnegative weighting parameters. $\mathbf{v}_t$ and $\mathbf{s}_t$ are the first and second moment of the gradients, respectively. Using the fact of $\sum_{i=0}^{t} \beta^i = \frac{1-\beta^t}{1-\beta}$, I can re-normalized the $\mathbf{v}_t$ and $\mathbf{s}_t$ as

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_1^t}, \tag{A.27a}$$

$$\hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-\beta_2^t} \tag{A.27b}$$

Then I use the same parameter update strategy as introduced in the RMSProp

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t + \epsilon}} \hat{\mathbf{s}}_t \tag{A.28}$$

Adam's effectiveness and stability make it the preferred optimization method for many algorithms.

Appendix B
IEEE Accepted Version

In this appendix, the following accepted versions of published papers are listed:

- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "Parameterizable and Jerk-Limited Trajectories with Blending for Robot Motion Planning and Spherical Cartesian Waypoints," IEEE International Conference on Robotics and Automation (ICRA) 2021 [98].
- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "6D Pose Estimation for Flexible Production with Small Lot Sizes based on CAD Models using Gaussian Process Implicit Surfaces," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020 [95]
- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "Deep Hierarchical Rotation Invariance Learning with Exact Geometry Feature Representation for Point Cloud Classification," IEEE International Conference on Robotics and Automation (ICRA), 2021 [96]
- **Lin, Jianjie**, Rickert, Markus, Perzylo, Alexander, Knoll, Alois, "PCTMA-Net: Point Cloud Transformer with Morphing Atlas-based Point Generation Network for Dense Point Cloud Completion," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021 [99]
- **Lin, Jianjie**, Rickert, Markus, and Knoll, Alois, "Grasp Planning for Flexible Production with Small Lot Sizes using Gaussian Process Implicit Surfaces and Bayesian Optimization," IEEE International Conference on Automation Science and Engineering (CASE) 2021 [97]

2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)
August 23-27, 2021. Lyon, France

Grasp Planning for Flexible Production with Small Lot Sizes based on CAD models using GPIS and Bayesian Optimization

Jianjie Lin, Markus Rickert and Alois Knoll

2021 IEEE 17th International Conference on Automation Science and Engineering (CASE) | 978-1-6654-1873-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/CASE49439.2021.9551451

Abstract— **Grasp planning for multi-fingered hands is still challenging due to the high nonlinear quality metrics, the high dimensionality of hand posture configuration, and complex object shapes. Analytical-based grasp planning algorithms formulate the grasping problem as a constraint optimization problem using advanced convex optimization solvers. However, these are not guaranteed to find a globally optimal solution. Data-driven based algorithms utilize machine learning algorithm frameworks to learn the grasp policy using enormous training data sets. This paper presents a new approach for grasp generation by formulating a global optimization problem with Bayesian optimization. Furthermore, we parameterize the object shape utilizing the Gaussian Process Implicit Surface (GPIS) to integrate the object shape information into the optimization process. Moreover, a chart defined on the object surface is used to refine the palm pose locally. We introduced a dual optimization stage to optimize the palm pose and contact points separately. We further extend the Bayesian optimization by utilizing the alternating direction method of multipliers (ADMM) to eliminate contact optimization constraints. We conduct the experiments in the graspit! Simulator that demonstrates the effectiveness of this approach quantitatively and qualitatively. Our approach achieves a 95% success rate on various common objects with diverse shapes, scales, and weights.**

I. Introduction

Only a limited number of small and medium-sized enterprises in Europe use robot systems in production, mainly dealing with small lot sizes and requiring a more flexible production process. It is, however, very time-consuming and expensive to adapt a robot system to a new production line, and it requires expert knowledge for deploying such a system, which, however, is not commonly available in shop floor workers [1]. Intuitive programming is currently proposed on the market for accelerating programming and remedying the problems caused by a lack of expert knowledge. Moreover, the service robots use semantic knowledge combined with reasoning and inference to solve the declarative goal. Automatically synthesizing a robot program based on the semantic product, process, and resource descriptions enable automatic adaptation to new processes. In this process, the recognition of objects and parts in the environments is involved, which is typically designed in CAD systems and described via a boundary representation [2]. Due to SMEs' small lot size production, it is not feasible to train the objects over a long period of time by using data-driven approaches. Based on this observation, it will accelerate the deploying time if we grasp the object firstly in a simulator with the CAD models and then transfer the preplanned grasp to the real world with a 6D pose estimation [3]. Dexterous robotic grasping planning has been an active research subject in the robotic community over the past decades. Grasping is essential in many areas, such as industrial factories and household scenarios. There have many different kinds of robotic hands, i.e., traditional parallel-jaw grippers, complex multi-fingered hands, or even vacuum-based end effectors. The goal of grasp planning aims to find a proper contact on the object and an appropriate posture of the hand related to the object to maximize grasp quality. This is a challenging task, especially for multi-fingered hands, due to different kinds of object shapes, the complicated geometric relationship between robotic hands and objects, and the high dimensionality of hand configurations. Grasp planning can be divided into analytic approaches [4] on the one side and empirical or data-driven approaches on the other side [5]. The analytical grasp synthesis approach is usually formulated as a constrained optimization problem over criteria that measure dexterity, equilibrium, and stability and exhibit a certain dynamic behavior. Besides, it requires the analysis of statically-indeterminate grasps [4] and under-actuated systems. The latter describes hands, in which the number of the controlled degrees of freedom is fewer than the number of contact forces, further increasing the complexity of grasp synergies. One common assumption made in analytic methods is that precise geometric and physical models are available to the robot. Furthermore, optimizing grasp quality with constraints based on a convex optimization solver such as SQP can not guarantee to find a good grasp. In contrast to analytic approaches, the empirical or data-driven methods rely on sampling the grasp candidate either from a data set or by first learning a grasp quality and then selecting the best by ranking them according to some specific metric. This work will study how to optimize the palm pose and contact point in the same framework by utilizing a global Bayesian optimization solver under consideration constraints. Gaussian Process Implicit Surface Atlas (GPIS-Atlas) is used to parameterize the diverse shape. Therefore the geometry information can be integrated into the Bayesian optimization framework. Furthermore, GPIS-Atlas has the capability to describe the perfect geometry model in the form of a CAD model or the noisy point clouds [6]. In the work [3], the 6D pose is estimated between an object in the form of a CAD model and the corresponding point clouds taken from an Ensenso Camera. Therefore, for finding an appropriate grasp pose for this object, we can directly apply our Bayesian optimization framework with the CAD model instead of

Jianjie Lin, Markus Rickert and Alois Knoll are with Robotics and Embedded System, Department of Informatics, Technische Universität München, Munich, Germany `jianjie.lin@tum.de`, `{rickert, knoll}@in.tum.de`

working on noisy point clouds. After that, we transform the grasp pose using the 6D pose transformation from [3].

II. Related Work

Multi-fingered hand grasp planning is still challenging due to the high dimensionality of hand structure and complex graspable object shapes. Automatic grasp planning is a difficult problem because of the vast number of possible hand configurations. Several different approaches have been proposed to find an optimal grasp pose over the past decades. Goldfeder et al. [7] introduced a database-backed grasp planning, which uses shape matching to identify known objects in a database with are likely to have similar grasp poses [7]. Ciocarlie et al. presented [8] Eigengrasp planning defines a subspace of a given hand's DOF space and utilizes the Simulated Annealing planner to find an optimized grasp. Miller et al. [9] proposed a primitive shape-based grasp planning which generates a set of grasps by modeling an object as a set of shape primitives, such as spheres, cylinders, cones, and boxes. Pelosso et al. [10] use an approach based on Support Vector Machines that approximate the grasp quality with a new set of grasp parameters. It considers grasp planning as a regression problem by given a feature vector, which should be defined heuristically.

With the continuous success of deep learning vision, researchers utilize deep learning, also in combination with reinforcement learning, to learn a grasp quality directly from an image via large training data sets [11]. Levine et al. [12] used between 6 and 14 robots at any given point in time to collect data in two months and train a convolutional neural network to predict grasp success for a pick-and-place task with a parallel-jaw gripper. Mahler et al. [13] proposed a Dex-Net-based deep learning framework using a parallel-jaw gripper or vacuum-based end effector learn a grasp policy based on millions of grasp experiments. Kalashnikov et al. [14] introduced a scalable self-supervised vision-based reinforcement learning framework to train a deep neural network Q-function by leveraging over 580k real-world grasp attempts. However, the deep learning-based algorithm can only take the 2d image as an input, and the trained neural network cannot be easily transferred to another robotic hand configuration. Varley et al. [15] obtained the geometry representation of grasping objects from point clouds using a 3D-CNN. Ten et.al [16] is the state of the art 6 DOF grasp planner (GPD). Liang et.al. [17] proposed an end-to-end PointNetGPD to detect the grasp configuration from a point sets. Mousavian et al. introduced a 6DOF GraspNet by sampling a set of grasping using a variational autoencoder. In addition to deep learning, the Bayesian optimization-based algorithm in [18] can consider uncertainty in input space to find a safe grasp region by optimizing the grasp quality. Furthermore, it utilizes unscented transformation-based Bayesian optimization (UBO), a popular nonlinear approximation method, to explore the safe region. However, UBO considers only the palm pose optimization without considering the contact point. We present a grasp planning approach in this work, where we combine Bayesian optimization with an analytical approach. We use the Grasp Wrench Space (GWS) [19] as grasp quality metric, which calculates the convex hull over discretized friction cones from the individual contact wrench spaces of all contacts. Due to the GWS metric's complexity, we explore the potential of Bayesian optimization to optimize this highly-nonlinear grasp quality problem. Since the hand posture (hand palm pose) and hand configuration can be considered separately because the finger's contact points on the object surface only depend on the hand posture and forward kinematics of hand joints, we propose a dual-stage approach: In the first stage, we optimize the hand palm pose without considering hand configuration, and in the second stage, we use the result of the first stage and optimize the contact points on the object surface. Our approach optimizes a hand palm pose $\boldsymbol{T} \in \mathrm{SE}(3)$ regarding its grasp quality. For this, we present the rotation in hyperspherical coordinates instead of a rotation matrix or quaternion. We further parameterize the object surface as a Gaussian Process Implicit Surface (GPIS) [6] and use a k-D tree to find the closest point between the current palm pose and object surface. Based on GPIS, we can further compute a chart C and the corresponding normal vector $\boldsymbol{N}_C$ on this nearest point. Utilizing this chart, we can make a local adaption of the palm pose to find a better location concerning the object's surface. In the second stage, we convert the problem of solving constraints between the contact points and the object surface to querying the GPIS given a known contact point. Since the standard framework of Bayesian optimization cannot solve this constraint optimization problem, we use the Alternating Direction Method of Multipliers (ADMM) [20] to assist the contact point optimization by decomposing the whole problem into a set of subproblems.

III. Problem Formulation

In general, to define a grasp, we need two sets of variables: the intrinsic variables to define the hand degrees of freedom (DOF) and the extrinsic variables to define the hand's position relative to the target object. Grasp planning is used to find the optimized contact points and an associated hand configuration to maximize grasp quality. The contact point on the object surface is denoted as $\boldsymbol{c} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n]$, where $\boldsymbol{c}_i \in \mathrm{SE}(3)$, and n is the number of fingers. We will assume that contact happens on the fingertip, and one finger only has one contact on the object surface $\mathcal{O}$. The finger joint of the hand configuration is described as $\boldsymbol{q} = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_m]$, where m is the DOF. Note that some finger joints are under-actuated (passive joint). Therefore the number of finger joints is not equal to the DOF. The pose of the palm is represented by $\boldsymbol{T}_{\text{palm}}(\boldsymbol{R}, \boldsymbol{t})$. We formulate the problem as:

$$\max_{\boldsymbol{c},\boldsymbol{q},\boldsymbol{T}_{\text{palm}}} Q(\boldsymbol{c}, \boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{c} = \mathrm{FK}_{\text{palm2c}}(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) \in \mathcal{O} \tag{1b}$$

$$q_{\min,i} \le q_i \le q_{\max,i}, \qquad i = 1 \cdots m, \tag{1c}$$

where $Q(\boldsymbol{c}, \boldsymbol{q}, \boldsymbol{T}_{\text{palm}})$ is the GWS, which is a 6-dimensional convex polyhedron, the epsilon and volume quality metric introduced by Ferrari and Canny [19]. The epsilon quality is defined as the minimum distance from the origin to any of the hyperplanes defining the boundary of the GWS. In contrast, the volume quality is the volume of GWS. $\mathrm{FK}_{\mathrm{palm2c}}$ is the forward kinematics from the palm pose to the contact points. The formulation (1b) constrains all contact points on the object surface $\mathcal{O}$. Furthermore, a contact is defined as any point where two bodies are separated by less than the contact threshold ϵ_c, but not interpenetrating. In this work, the object surface will be parameterized by using GPIS to easily check if the contact point satisfies the constraint (1b). The problem in (1) is a high-dimensional nonlinear constraint problem, besides the gradient of the objective function and constraints cannot be analytically computed. Further, the convex optimization solver can only find a local minimum. Bayesian optimization is applied to find a near-global optimization solution for these problems. Since the optimization of palm pose is independent of optimization of contact point, we switch between optimizing the palm pose and the contact point.

IV. Bayesian Optimization for Grasp Planning

Bayesian optimization is a global optimization method, which can be used to solve the problem

$$x_{\mathrm{optimized}} = \max_{x \in \mathcal{X}} f_{\mathrm{obj}}(x)\,, \tag{2}$$

where the objective function $f_{\mathrm{obj}}(x)$ is a black-box function or a function which is expensive to evaluate and $\mathcal{X} \subseteq \mathbb{R}^D$ is a bounded domain. We use the Latin Hypercube Sampling (LHS) to get the initial sampling, and save it as data set $\mathcal{D}_{0:t-1} = \left\{ \left(\mathbf{x}_0, y_0\right), \ldots, \left(\mathbf{x}_{t-1}, y_{t-1}\right) \right\}$, and learn a Gaussian process model (GP). The essential step is to choose an appropriate acquisition function. Here, we use Expected Improvement [21], which is defined as

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}\left[\max\left(f_{\mathrm{obj}}(\boldsymbol{x}) - f_{\mathrm{obj}}(\boldsymbol{x}^{+}), 0\right)\right]\,, \tag{3}$$

where $\mathbb{E}$ is the expectation function, $f_{\mathrm{obj}}(\boldsymbol{x}^{+})$ is the best observation with the location $\boldsymbol{x}^{+}$ so far. The $\mathrm{EI}(\boldsymbol{x})$ can be evaluated analytically under the GP model as

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{1}\left(\sigma(\boldsymbol{x})\right)\Big(\left(\mu(\boldsymbol{x}) - f_{\mathrm{obj}}(\boldsymbol{x}^{+}) - \xi\right)\Phi(Z) + \sigma(\boldsymbol{x})\phi(Z)\Big)\,,$$

with $Z = \mathbb{1}\left(\sigma(\boldsymbol{x})\right)\left(\frac{\mu(\boldsymbol{x}) - f_{\mathrm{obj}}(\boldsymbol{x}^{+}) - \xi}{\sigma(\boldsymbol{x})}\right)$, where $\mathbb{1}(x)$ is the indicator function that is equal to 0 for $x \le 0$ and equal to 1 otherwise. The mean $\mu(\boldsymbol{x})$ and standard deviation $\sigma(\boldsymbol{x})$ are defined in the GP posterior at $\boldsymbol{x}$, and Φ and ϕ are the CDF (cumulative distribution function) and PDF (probability density function) of the standard normal distribution. ξ is a parameter which balances between the exploration and exploitation. The objective function in our algorithm for optimizing the grasp contact points is grasp quality which consist of epsilon and volume quality.

$$f_{\mathrm{obj}}(\boldsymbol{x}) = \left(\mathbb{1}(q_\epsilon) q_\epsilon + \lambda\, q_{\mathrm{volume}}\right)\,, \tag{4}$$

where λ is a predefined parameter. The Matérn covariance function ($\nu = 5/2$) is chosen as kernel function for the Gaussian process model in the Bayesian optimization and can be described as: $K_{5/2}(d) = \sigma^2\left(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2}\right)\exp\left(-\frac{\sqrt{5}d}{\rho}\right)$ with hyper parameter σ and length-scale ρ. The parameter d is the distance between two query points. Since we need to optimize the palm pose, which is interpreted as a transformation. We define the distance between two transformation matrices as $\Delta_{\boldsymbol{T}} = \left\|\boldsymbol{t}_1 - \boldsymbol{t}_2\right\| + \left\|\log(\boldsymbol{R}_1^{\mathrm{T}}\boldsymbol{R}_2)\right\|_{\mathrm{F}} / \sqrt{2}$, where the term $\left\|\log(\boldsymbol{R}_1^{\mathrm{T}}\boldsymbol{R}_2)\right\|_{\mathrm{F}} / \sqrt{2}$ is the geodesic distance defined in the Riemann manifold. We use the RProp (Resilient Propagation) [22] for optimizing the hyperparameter.

The object surface in our algorithm is described as a GPIS and every point lying on the surface in the set $\mathcal{X}_{\prime}$ should satisfy the equality constraints $\mathcal{X}_{\prime} = \left\{\mathbf{x} \in \mathbb{R}^3 : f_{\mathrm{GPIS}}(\boldsymbol{x}) = 0\right\}$. Furthermore, the tangent space of each point on these surface will be computed by using

$$\begin{bmatrix} \nabla f_{\mathrm{GPIS}}^{\mathrm{T}}(\boldsymbol{x}_i) \\ \boldsymbol{\Phi}_i^{\mathrm{T}}(\boldsymbol{x}_i) \end{bmatrix} \boldsymbol{\Phi}_i(\boldsymbol{x}_i) = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix}\,, \tag{5}$$

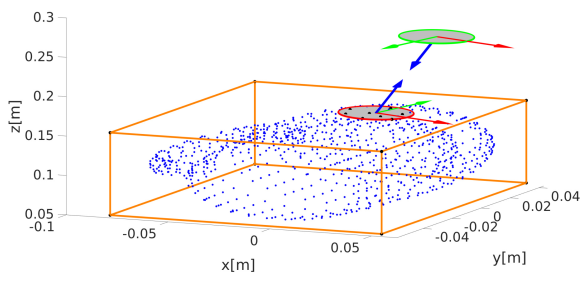
where $\boldsymbol{\Phi}_i(\boldsymbol{x}_i) \in \mathbb{R}^{3\times 2}$ is the basis of tangent space at the location $\boldsymbol{x}$ and $\nabla f_{\mathrm{GPIS}}(\boldsymbol{x}_i) \in \mathbb{R}^{3\times 1}$ is the gradient of implicit function for $\boldsymbol{x}$. By using $\boldsymbol{\Phi}_i(\boldsymbol{x})$, we can map $\boldsymbol{x}_i$ to $\boldsymbol{x}_i'$ with $\boldsymbol{x}_i' = \boldsymbol{x}_i + \boldsymbol{\Phi}_i(\boldsymbol{x}_i)\,\boldsymbol{u}_i$, where $\boldsymbol{u}_i \in \mathbb{R}^{2\times 1}$ is a point in the local coordinate on this chart. The sample value $\boldsymbol{x}_i'$ is shown as black dots in Fig. 1a, the tangent vector on the chart is shown as red and green arrows, while the blue arrow shows the normal vector.

A. Hand Palm Pose Optimization (HPP-Opt)

To optimize the palm pose, we need to take the transformation $\boldsymbol{T}_{\mathrm{palm}}(\boldsymbol{R}, \boldsymbol{t}) \in \mathrm{SE}(3)$ into account, where the rotation matrix $\boldsymbol{R}$ can be interpreted by using a unit quaternion $\boldsymbol{q}$. Since the unit quaternion manifold $\mathcal{M}_{\mathbb{H}}$ is an Riemannian manifold, by virtual equality of $\mathcal{M}_{\mathbb{H}}$ and 4D unit hypersphere $\mathcal{S}^3 = \left\{x \in \mathbb{R}^{3+1} : \|\boldsymbol{x}\| = 1\right\}$, the quaternion $\boldsymbol{q}$ can be represented in the hyperspherical coordinates with ϕ, ψ, θ where ϕ, ψ range over $[0, \pi]$ and θ ranges over $[0, 2\pi)$. Employing hyperspherical coordinates, the constraints of $\boldsymbol{q}$ disappear. Therefore, the palm pose optimization is converted to an unconstrained optimization:

$$\max_{\phi,\psi,\theta,\boldsymbol{t}_{\mathrm{palm}}} f_{\mathrm{obj}}\left(\boldsymbol{q}, \boldsymbol{T}_{\mathrm{palm}}(\boldsymbol{R}(\phi,\psi,\theta), \boldsymbol{t}_{\mathrm{palm}})\right)\,. \tag{6}$$

To reduce the search space, we constrain the palm pose between two bounding boxes, represented by an axis-aligned



(a)

Fig. 1: Visualization of (a) Tangent space on a mug surface

minimum bounding box (AABB), denoted as $\mathcal{X}_{\text{AABB},i}$ with $i \in \{1, 2\}$. The smaller one is shown as an orange cube in Fig. 1a. We define the variable $\boldsymbol{x}_{\text{palm}} = [\boldsymbol{t}, \phi, \psi, \theta]^{\mathrm{T}} \in \mathbb{R}^{6\times 1}$. Sampling a point between two bounding box cannot be formulated mathematically, therefore, we use an ellipsoid $\mathcal{E}_1 = (a_1, b_1, c_1, a_0, b_0, c_0)$ to approximate $\mathcal{X}_{\text{AABB},1}$ and another ellipsoid $\mathcal{E}_2 = (a_2, b_2, c_2, a_0, b_0, c_0)$ to approximate the bigger bounding box $\mathcal{X}_{\text{AABB},2}$. As the result, the palm pose sample $\boldsymbol{t} = [t_x, t_y, t_z]$ can be formulated as

$$\begin{aligned} t_x &= a_0 + r * a_1 \sin(\theta)\cos(\phi) \\ t_y &= b_0 + r * b_1 \sin(\theta)\sin(\phi) \\ t_z &= c_0 + r * c_1 \cos(\theta)\,, \end{aligned}$$

where $m_0 = \frac{k_{\min,1}+k_{\max,1}}{2}$, $m_1 = \frac{k_{\max,1}-k_{\min,1}}{2}$, $m \in \{a, b, c\}$, and $k \in \{x, y, z\}$. The parameter r is a random variable which ranges over $[1, r_{\max}]$, where

$$r_{\max} = \sqrt{\frac{1}{\left(\frac{a_1\sin(\theta)\cos(\phi)}{a_2}\right)^2 + \left(\frac{b_1\sin(\theta)\sin(\phi)}{b_2}\right)^2 + \left(\frac{c_1\cos(\theta)}{c_2}\right)^2}}\,.$$

The parameter θ ranges over $[0, \pi]$ and θ ranges over $[0, 2\pi)$. The problem in (6) can be solved by using Bayesian optimization.The solution found by BO is based on the probability of best grasp distribution. To improve the performance, a local adaption based on GPIS-Atlas is proposed.

1) GPIS-Atlas based local adaption: The first step is to find the closest point from the current palm pose to the object using k-D tree algorithm. Assuming we found the pose $\boldsymbol{t}_{\text{closest}}$, a chart C_i with the center point $\boldsymbol{t}_{\text{closest}}$ is created by solving (5). We denoted the outward unit normal vector of this chart C_i as $\boldsymbol{N}_{\text{closest}}$ and the robot hand posture is designed to orient to the direction of a chart normal $\boldsymbol{N}_C$. We apply the following approaches to get the pose $\boldsymbol{T}$. Assuming that the normal vector of hand in the initial state points to the z-axis $\boldsymbol{n}_z$, the hand is currently in local frame $\mathcal{H}_1$ with rotation matrix ${}^{\mathcal{W}}\boldsymbol{R}_{\mathcal{H}_1}$ with respect to the world frame $\mathcal{W}$. The next hand configuration should point to the normal direction $\boldsymbol{N}_C$ in local frame $\mathcal{H}_{C_i}$, therefore the corresponding rotation matrix ${}^{\mathcal{W}}\boldsymbol{R}_{\mathcal{H}_{C_i}}$ can be interpreted in angle-axis representation $[n_{\text{axis}}, \theta_{\text{axis}}]$ with $\theta_{\text{axis}} = \text{atan2}(\|\boldsymbol{n}_z \times \boldsymbol{N}_C\|, \boldsymbol{n}_z \cdot \boldsymbol{N}_C)$ and $n_{\text{axis}} = \boldsymbol{n}_z \times \boldsymbol{N}_C$. As a result, we can transform the local frame $\mathcal{H}_1$ to $\mathcal{H}_{C_i}$ with the rotation transformation as ${}^{\mathcal{H}_1}\boldsymbol{R}_{\mathcal{H}_{C_i}} = {}^{\mathcal{W}}\boldsymbol{R}_{\mathcal{H}_1}^{\mathrm{T}}\,{}^{\mathcal{W}}\boldsymbol{R}_{\mathcal{H}_{C_i}}$. Furthermore, the translation of the palm pose is defined as $\boldsymbol{t}_{\text{palm}} = \boldsymbol{t}_{\text{closest}} + \lambda \boldsymbol{N}_{\text{closest}}$. This means that the new palm pose $\boldsymbol{x}_{\text{palm}}$ is parallel to the chart C_i with the distance $\|\lambda\|$, as shown in Fig. 1a. The parameter is optimized so that the hand is not colliding with the object. The whole transformation is defined as $\boldsymbol{T}_{\text{palm}} = \boldsymbol{T}(\boldsymbol{I}, \boldsymbol{t}_{\text{palm}})\boldsymbol{T}({}^{\mathcal{W}}\boldsymbol{R}_{\mathcal{H}_{C_i}}, \boldsymbol{0})\boldsymbol{T}(\boldsymbol{R}_z, \boldsymbol{0})$. The transformation $\boldsymbol{T}(\boldsymbol{R}_z, \boldsymbol{0})$ is used to further guarantee no collision. Furthermore, we can define a point set on the chart C_i as $\mathcal{X}_{C_i}$ and randomly choose a sample $\boldsymbol{t}_{\text{sample}} \in \mathcal{X}_{C_i}$ as $\boldsymbol{t}_{\text{closest}}$.

B. ADMM-Based Contact Point Optimization (ADMM-CP-Opt)

The contact point optimization is used to find a set of desired joints $\boldsymbol{q}$ for each finger and the contact points $\boldsymbol{c}$ on the object surface. It can be described as

$$\begin{aligned} \max_{\boldsymbol{c},\boldsymbol{q}} \quad & f_{\text{obj}}(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) && \text{(7a)} \\ \text{s.t.} \quad & \boldsymbol{c} = \text{FK}(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}), && \text{(7b)} \\ & |f_{\text{GPIS}}(\boldsymbol{c}_k)| \le \epsilon_c, \qquad k = 1 \cdots n && \text{(7c)} \\ & q_k \in [q_{\min,k}, q_{\max,k}], \qquad k = 1 \cdots m\,, && \text{(7d)} \end{aligned}$$

where FK calculates the forward kinematics, and n represents the number of contact points on the object surface. The parameter m is the DOF of a hand. A contact point is represented as a transformation $\boldsymbol{T}_{c_i}$. However, each finger has fewer joints than 6, which results in an underestimated system. Consequently, we cannot directly calculate the inverse kinematics based on the contact points, and it is not possible to arbitrarily move the fingertip on the object surface. To relax the constraints, we will not fix the palm pose, but constrain the palm pose on the chart $C_{\text{palm},i}$, which is parallel to chart C_i on the object surface, therefore $\boldsymbol{T}_{\text{palm}} \in C_{\text{palm},i}$. Since the equality constraints cannot be solved by using Bayesian optimization, the Alternating Direction Method of Multipliers (ADMM) based Bayesian optimization [20] is utilized to solve the contact pose optimization problem (7) with the new formulation

$$\max_{\boldsymbol{q}\in\boldsymbol{B}} f_{\text{obj}}(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) + g_c(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}})\,, \tag{8}$$

with $g_c(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) = \mu \sum_{i=0}^{n} \boldsymbol{c}_i(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}})^2$ and $\boldsymbol{c}_i(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) = \left| f_{\text{GPIS}}\Big(\text{FK}_i\big(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}(\boldsymbol{R}, \boldsymbol{t})\big)\Big)\right| - \epsilon_c$. In order to solve (8), ADMM introduces an auxiliary variable $\boldsymbol{z}$, resulting in

$$\begin{aligned} \max_{\boldsymbol{q},\boldsymbol{z}\in\boldsymbol{B}} \quad & f_{\text{obj}}(\boldsymbol{q}, \boldsymbol{T}_{\text{palm}}) + \mathrm{g}_c(\boldsymbol{z}, \boldsymbol{T}_{\text{palm}}) && \text{(9a)} \\ \text{s.t.} \quad & \boldsymbol{q} = \boldsymbol{z}\,. && \text{(9b)} \end{aligned}$$

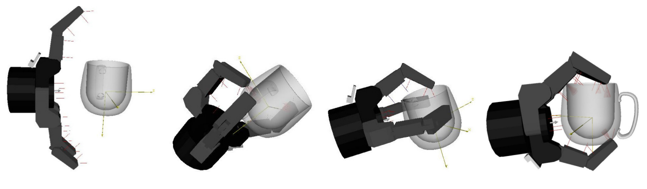
In the following, we neglect $\boldsymbol{T}_{\text{palm}}$ in f_{obj} and g_c. By applying Augmented Lagrangian function for equation (9), a new objective function is formulated as

$$\mathcal{L}_\rho(\boldsymbol{q}, \boldsymbol{z}, \boldsymbol{y}) = f_{\text{obj}}(\boldsymbol{q}) + g_c(\boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{q} - \boldsymbol{z} + \frac{\boldsymbol{y}}{\rho}\right\|_2^2 \tag{10}$$

Therefore, we can solve $f_{\text{obj}}(\boldsymbol{q})$ and $g_c(\boldsymbol{z})$ by alternating over the following sub problems:

$$\begin{aligned} \boldsymbol{q}^{k+1} &= \arg\max_{\boldsymbol{q}}\ f_{\text{obj}}(\boldsymbol{q}) + \frac{\rho}{2}\left\|\boldsymbol{q} - \boldsymbol{z}^k + \frac{\boldsymbol{y}^k}{\rho}\right\|_2^2 && \text{(11a)} \\ \boldsymbol{z}^{k+1} &= \arg\max_{\boldsymbol{z}}\ \mathrm{g}_c(\boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{q}^{k+1} - \boldsymbol{z} + \frac{\boldsymbol{y}^k}{\rho}\right\|_2^2 && \text{(11b)} \\ \boldsymbol{y}^{k+1} &= \boldsymbol{y}^k + \rho(\boldsymbol{q}^{k+1} - \boldsymbol{z}^{k+1})\,. && \text{(11c)} \end{aligned}$$

The optimal condition is defined as $\left\|\boldsymbol{q}^{k+1} - \boldsymbol{z}^{k+1}\right\|_2 \le \epsilon^{\text{primal}}$ and $\left\|\rho(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right\|_2 \le \epsilon^{\text{dual}}$, where ϵ^{primal} and ϵ^{dual} are two predefined optimality tolerances. Each sub problem is solved by using Bayesian optimization.


(a) Mug: Initial (b) Random Alg (c) EigenGrasp (d) HPP-Opt

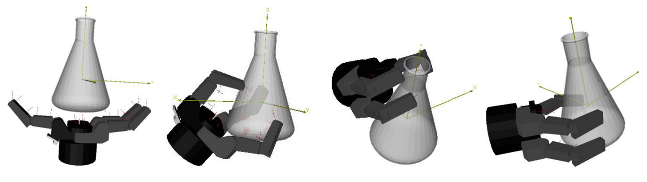

(e) Flask: Initial (f) Random Alg (g) EigenGrasp (h) HPP-Opt

Fig. 2: Comparison of different Grasp planning Algorithm examples for different objects with the multi-fingered hand

TABLE I: Evaluation of different grasp planning algorithm for all data sets. The results is an average of first 20 best Grasp Candidates. A greater value of epsilon and volume means a more stable grasp. The best result is highlighted in *green*

	Algorithms							
	random Alg		EigenGrasp		HPP-Opt		ADMM-CP-Opt	
quality	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}
Mug	0	9.5352e-05	1.7281e-04	7.1981e-04	0.0555	0.0097	0.06	0.0161
Flask	0	3.9603e-05	4.9050e-04	2.0142e-04	0.0107	0.0014	0.0107	0.0026
Phone	0	3.9209e-05	0.0017	4.6942e-04	0.0142	0.0035	0.0142	0.0042
Sphere	0.0042	0.0019	0	0.0011	0.0495	0.0121	0.050	0.0279
Bishop	0	8.1935e-05	0.0012	9.7708e-05	0.0094	0.0011	0.0094	0.0016

TABLE II: The best grasp from 20 grasp candidates. The best result is highlighted in *green*

	Algorithms							
	random Alg		EigenGrasp		HPP-opt		ADMM-CP-Opt	
quality	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}	q_ϵ	q_{volume}
Mug	0	0.0012	0.0065	0.0011	0.1039	0.0151	0.1158	0.0340
Flask	0	0.0004	0.0130	0.0031	0.0449	0.0018	0.0452	0.0019
Phone	0	0.0004	0.0310	0.0007	0.0208	0.0198	0.0335	0.0006
Sphere	0.0844	0.0301	0	0.0086	0.0914	0.0362	0.1676	0.0728
Bishop	0	0.0012	0.0164	0.0004	0.0390	0.0020	0.0444	0.0045

V. Experiment

Simulation results are introduced in this section to verify the effectiveness of our algorithm. The experiment is executed in the platform Graspit! [23] by using Barret hand Barret hand hat three fingers: finger one hat two joints, where the last joint is under-actuated. Finger two and three have one common joint. Each finger has three joints where both fingers have a passive joint. Therefore the Barret hand fingers have totaled 4 DOFs. The hand palm pose is denoted as a 6-dimensional vector. Consequently, the whole Barret hand system has totaled 10 DOFs. The experiment's graspable object is stored in the mesh file The GPIS describes the object by using the mesh triangle verities as the input. Our approach achieves a 95% success rate on various commonly used objects with diverse appearances, scales, and weights compared to the other algorithm. All evaluations were performed on a laptop with a 2.6 GHz Intel Core i7-6700HQ and 16 GB of RAM.

A. Experiment on HPP-Opt

In this section, the algorithm HPP-Opt is compared with other grasp planning. The first grasp planning approach is a simulated annealing grasp planner using an auto grasp quality energy as a search strategy, which behaves like a random grasp planning. The second approach uses an EigenGrasp planner combined with a simulated annealing solver to guide potential quality energy. The Bayesian optimization and simulated annealing are both global optimization solvers, where the latter is a probabilistic technique for approximating the global optimum of a given function. We compare the algorithms with different kinds of shapes. The experiment is conducted in a fixed time of 20 seconds, and we average the first 20 best grasp candidates. The best grasp candidate of mug and flask from whole grasp candidates are visualized in Fig. 2. The initial state of Barret hand and object are randomly defined, shown in Fig. 2a and 2e. It can be seen that the hand configuration selected by random grasp planning is skewed to the object, and contact points on the surface are not properly, and the resulting quality is also worse. The results of EigenGrasp show a better solution where the palm pose is trying to parallel to the object surface. The palm pose selected by HPP-Opt is more reasonable in comparison to the other two algorithms. We show the quantitative result is in table (I). In different kinds of geometry shapes, our algorithm can achieve a much more stable grasp than other algorithms. The epsilon quality achieved by the first approach is almost zero besides in the case of a sphere. The epsilon quality by Eigen grasps a minimal value. On average, the epsilon quality of HPP-Opt is 28.5 times greater than Eigen grasp's epsilon quality. And the HPP-Opt's volume quality is 32.1417 times greater than Eigen Grasp's volume quality. Furthermore, we show the best grasp candidate from the first 20 best grasp candidates in Table II.

B. Experiment of integration HPP-Opt and ADMM-CP-Opt

In section V-A, hand palm pose is optimized based on the Bayesian optimization algorithm combing with local adaption, and hand finger configuration is set based on the function of *AutoGrasp* from Graspit [23]. The principle of *AutoGrasp* is to close each hand finger DOF at a rate equal to a predefined speed factor multiple with its default velocity, and the movement is stopped at the contact point. Therefore ADMM-CP-Opt is used to assist the HPP-Opt to find a better hand finger configuration. The comparison result is visualized in Fig. 3. In the case (3a), fingers two and three are too close and grasp the bottom of the flask. As a consequence, the resulting triangle has a small internal angle. Applying ADMM-CP-Opt splits the fingers two and three by maximizing epsilon and volume quality and makes the resulting triangle closer to the equilateral triangle with a more stable grasp. The same improvement happen in the Fig. 3b - Fig. 3f as well. The solution founded by ADMM-CP-Opt is trying to make the resulting triangle as closer as the equilateral triangle. In Table I, ADMM-CP-Opt improves the volume quality of HPP-Opt. And In table II, ADMM-CP-Opt shows a better grasp than HPP-Opt in most cases under epsilon and volume quality metrics.

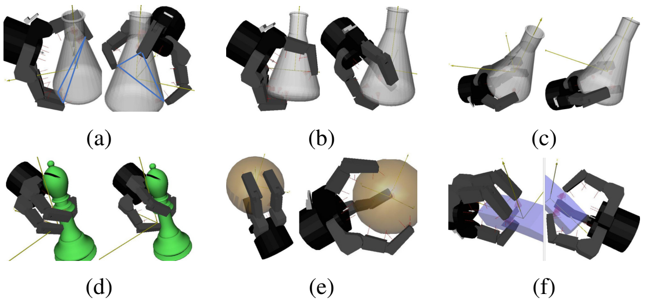


Fig. 3: The comparison result of HPP-Opt and ADMM-CP-Opt. In each sub figure, the left one is result of HPP-Opt, and the right one is improved by using ADMM-CP-Opt

VI. Conclusion and Outlook

We propose a new algorithm for grasp planning with multi-fingered hands by optimizing the hand palm pose and hand finger configuration separately. No initial configuration is required using a global Bayesian optimization solver, which shows superiority over the convex optimization solver. We propose a dual-stage optimization process by considering the independence of the hand palm pose and finger configuration. In the first stage, we utilize a GPIS to describe the graspable object so that collision checking of contact points can be integrated into the optimization framework. Furthermore, the chart on the object surface can be computed using the GPIS, which can be used to explore the local information of objects. Two ellipsoids are used to define the palm pose constraints domain. Relying on the first stage result, we apply an ADMM based Bayesian optimization to optimize the contact points. The whole process will switch between HPP-Opt and ADMM-CP-Opt. We collect the best 20 Grasps, and the final grasp is selected by ranking the grasp candidates under consideration of epsilon and volume quality. In this work, we describe only the object in GPIS. In future work, we can describe the hand part into GPIS so that the collision checking can be converted to a problem by querying the distance between two surfaces. Besides, the robot arm is not considered in the grasping scenario. It will be interesting to integrate the constraints of robot arm manipulability in the objective function.

References

[1] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. G. Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kesslar, and M. Danzer, "SMErobotics: Smart robots for flexible manufacturing," *IEEE Robotics and Automation Magazine*, vol. 26, no. 1, pp. 78–90, 2019.

[2] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Sept. 2015, pp. 4197–4203.

[3] J. Lin, M. Rickert, and A. Knoll, "6D pose estimation for flexible production with small lot sizes based on cad models using gaussian process implicit surfaces," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[4] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000, pp. 348–353.

[5] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis–A survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2013.

[6] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," in *Proceedings of the Gaussian Processes in Practice Workshop*, June 2006.

[7] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 1710–1716.

[8] M. T. Ciocarlie, "Low-dimensional robotic grasping: Eigengrasp subspaces and optimized underactuation," Ph.D. dissertation, Columbia University, 2010.

[9] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003, pp. 1824–1829.

[10] R. Pelossof, A. Miller, P. Allen, and T. Jebara, "An svm learning approach to robotic grasping," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004, pp. 3512–3518.

[11] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine, "End-to-end learning of semantic grasping," *arXiv preprint arXiv:1707.01932*, 2017.

[12] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[13] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 1957–1964.

[14] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[15] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2442–2447.

[16] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[17] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 3629–3635.

[18] J. Nogueira, R. Martinez-Cantin, A. Bernardino, and L. Jamone, "Unscented bayesian optimization for safe robot grasping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 1967–1972.

[19] C. Ferrari and J. F. Canny, "Planning optimal grasps." in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295.

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[21] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, Dec 1998.

[22] M. Blum and M. A. Riedmiller, "Optimization of gaussian process hyperparameters using Rprop," in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Apr. 2013, pp. 339–344.

[23] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, vol. 11, pp. 110–122, 2004.

RightsLink

Sign in/Register

Grasp Planning for Flexible Production with Small Lot Sizes based on CAD models using GPIS and Bayesian Optimization

Requesting permission to reuse content from an IEEE publication

Conference Proceedings:
2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)

Author: Jianjie Lin; Markus Rickert; Alois Knoll

Publisher: IEEE

Date: 23-27 Aug. 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

6D Pose Estimation for Flexible Production with Small Lot Sizes based on CAD Models using Gaussian Process Implicit Surfaces

Jianjie Lin[1], Markus Rickert[1] and Alois Knoll[2]

***Abstract*— We propose a surface-to-surface (S2S) point registration algorithm by exploiting the Gaussian Process Implicit Surfaces for partially overlapping 3D surfaces to estimate the 6D pose transformation. Unlike traditional approaches, that separate the corresponding search and update steps in the inner loop, we formulate the point registration as a nonlinear non-constraints optimization problem which does not explicitly use any corresponding points between two point sets. According to the implicit function theorem, we form one point set as a Gaussian Process Implicit Surfaces utilizing the signed distance function, which implicitly creates three manifolds. Points on the same manifold share the same function value, indicated as $\{1, 0, -1\}$. The problem is thus converted into finding a rigid transformation that minimizes the inherent function value. This can be solved by using a Gauss-Newton (GN) or Levenberg-Marquardt (LM) solver. In the case of a partially overlapping 3D surface, the Fast Point Feature Histogram (FPFH) algorithm is applied to both point sets and a Principal Component Analysis (PCA) is performed on the result. Based on this, the initial transformation can then be computed. We conduct experiments on multiple point sets to evaluate the effectiveness of our proposed approach against existing state-of-the-art methods.**

I. Introduction

While the majority of European companies consists of small and medium-sized enterprises, only a very limited number of them currently uses robot systems in their production. In contrast to larger companies, they mainly deal with small lot sizes and constantly changing production processes. Adapting a robot systems to new products and parts is however very time-consuming and requires expert knowledge in robotics that is not commonly found in shop floor workers [1]. While a number of modern robot systems currently on the market proposes an easier programming concept based on reusable skills, these approaches still require a manual adaptation to new processes. In contrast to this, approaches from service robotics are able to automatically solve declarative goal specifications by using semantic knowledge in combination with reasoning and inference [2].

Synthesizing robot programs based on semantic product, process, and resource descriptions enables an automatic adaptation to new processes and involves handling the recognition of objects and parts in the environment. These parts are typically designed in CAD systems and described via a boundary representation [3]. In order to grasp them with a robot, a full 6D pose estimation is required. Given the small lot size production of SMEs with constantly changing objects, it is not feasible to train object recognition models over a long period of time. Recognizing these parts efficiently by directly using their CAD models during execution is therefore essential in achieving short changeover times. Fig. 1 shows an example of such an assembly use case for a mechanical gearbox together with a point cloud scene captured by the 3D camera sensor attached to the robot.

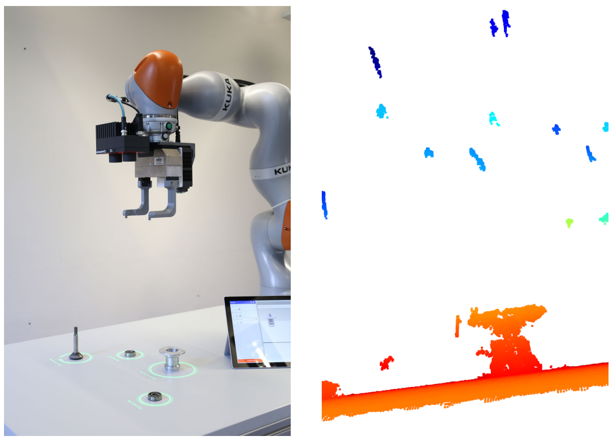

(a) (b)

Fig. 1: Robot setup for assembling a gear box with (a) a lightweight robot and a 3D camera sensor, (b) point cloud scene of mechanical gearbox parts on the table.

Point registration is one of the main approaches in computing the pose transformation by two given point sets and is widely used in MRI/CAT scan alignment [4] and robot manipulation [5]. The problem is especially challenging when two noisy point sets only partially overlap without initial alignment. A standard approach for point registration is based on the Iterative Closest Point (ICP) algorithm [6], [7]. It is interesting due to its intuitive and straightforward implementation. The identification of corresponding points follows a greedy search algorithm that is subjective to local minima and identifies incorrect points for some rotations. Furthermore, the success of ICP heavily relies on a good initial alignment. There are many variants which aim at optimizing the process for correspondence search, such as widening the convergence basin, heuristic global search, relaxed assignments, or distance fields, which however often fail to achieve a better performance and typically follow the principle of point to point (p2p) or point to surface (p2s) registration. Furthermore, with increasingly powerful neural

[1]Jianjie Lin, Markus Rickert are with fortiss, An-Institut Technische Universität München, Munich, Germany `{lin,rickert}@fortiss.org`

[2]Alois Knoll are with Robotics and Embedded System, Department of Informatics, Technische Universität München, Munich, Germany `knoll@in.tum.de`

networks, many researchers started applying deep learning to the problem of computing a pose transformation [8]. However, these approaches still follow the concept of finding the corresponding point and use a RANASAC-based Perspective-n-Point(PnP) algorithm to acquire the 6D pose. In addition, they require a large data set to encode a surrogate task and cannot be transferred to another task efficiently. For a flexible production application however, efficiency is a key factor required in any suitable algorithm.

This work, to the best of our knowledge, is the first one to consider the surface-to-surface (s2s) point registration and to describe one surface as an implicit function by employing Gaussian process regression, also referred to as Gaussian Process Implicit Surfaces (GPIS). We define three manifolds by borrowing the idea of signed distance functions (SDF) [9] with values $\{1, 0, -1\}$. All points on a surface should have the same value of 0. Instead of searching the corresponding points between two different point sets, the goal is now to find a rigid transformation that makes function value zero by transferring the point using this transformation. We evaluate the presented GPIS-based point registration on multiple point sets. In comparison to state-of-the-art algorithms, the presented approach can arrive at or exceed the same accuracy. We prioritize robustness over convergence speed and our approach can achieve more robust results than most of the existing algorithms. The primary advantage in contrast to other algorithms is that our approach does not require finding the corresponding point iteratively. The initialization for the transformation can be calculated based on a FPFH and PCA. The registration problem is efficiently solved with a Lie algebra-based Gaussian Newton solver.

II. Related Work

6D pose estimation is widely used and extensively studied and point registration technology is commonly used to find the spatial relationship between two point sets. Most of the advanced algorithms in this field are based on ICP and several variants exist [10], [7]. The typical work flow for geometric registration consists of two stages: initial (global) alignment and local refinement. Initial alignment is either based on simple Euclidean distance or on more complex sampling-based algorithms that identify matching points by utilizing local geometrical descriptors like Fast Point Feature Histogram (FPFH) [11], [12]. RANSAC [13] can be applied against outliers. In the following, the initial rigid transformation can be estimated by using a least squared method or by using the branch-and-bound framework (BnB) [14], which is a global optimization algorithm. In both cases, finding a good initial alignment can be computationally expensive. After the initial alignment, local refinement is executed by alternating the steps for finding the nearest neighborhood and the steps for updating the transformation based on the greedy search algorithm. This is susceptible to local minima and can only produce an accurate result with a good initialization. Several variants can be used to improve the performance: Fitzgibbon et al. [15] proposed a Levenberg-Marquardt algorithm that uses finite differences to optimize the objective function. Granger et al. [6] applied Expectation-Maximization (EM) principles to consider Gaussian noise, which can improve the robustness of local registration. Li et al. [16] utilize the Gaussian mixture model to model the surface uncertainty so that it increases the robustness of the registration. Myronenko et al.[17] introduced Coherent Point Drift (CPD), which is agnostic as to the used transformation model and similar to GMM takes a probabilistic approach to the alignment of point sets. Chavdar et al. [18] applied stochastic optimization to consider noise robustness, outlier resistance, and optimal global alignment. Yang et al. [14] proposed Globally Optimal ICP (Go-ICP) by using the Branch and Bound framework to derive the lower and upper bound for the error function and to integrate a local ICP in the same frame. Guo et al. [12] introduced the Fast Global Registration (FGR) algorithm that uses a scaled Geman-McClure estimator to describe the error function, optimizes the objective function by using Block Coordinate Descent, and applies FPFH to search the corresponding set before optimization.

All algorithms mentioned above share the requirement of finding the correct corresponding pair. This is followed by using either greedy search or a global optimizer to get the final rigid transformation. With continuous improvement in the field of deep learning, many researchers began to learn the 6D pose directly by using RGB images [19]. However, a large number of point sets is required to learn the surrogate task and the underlying relationship between learned loss function and the pose accuracy is still unclear. Such an approach cannot be easily deployed in a flexible production line. In this paper, we introduce a different approach for GPIS-based point registration (GPIS-S2SPR), that does not need to explicitly use corresponding points. By applying Gaussian Process Implicit Surfaces, we can achieve a more robust performance compared to state-of-the-art algorithms.

III. Problem Formulation

The standard ICP algorithm aims to estimate a rigid transformation $\boldsymbol{T} = \{\boldsymbol{R}, \boldsymbol{t}\}$ between given two point sets $\mathcal{X} = \{\boldsymbol{x}_i\}, i = 0, \ldots, n$ and $\mathcal{Y} = \{\boldsymbol{y}_i\}, i = 0, \ldots, m$, that minimizes the object function

$$E(\boldsymbol{T}) = \min \sum\nolimits_{i=1}^{n} \left(\boldsymbol{x}_i - \boldsymbol{T}\boldsymbol{y}_{\arg\min_{j=0,\ldots,m}\left(\|\boldsymbol{x}_i - \boldsymbol{T}\boldsymbol{y}_j\|^2\right)} \right) \qquad (1)$$

by iteratively finding corresponding points. The objective function in (1) however is a non-convex function and therefore susceptible to local minima.

In this paper, we consider the problem from a different angle: aligning two point sets independent of corresponding points. We model one of the point set as an implicit surface, which can capture the local structure of the surface and then transfer another point set to this implicit surface. We assume that all points on the same surface share the same function value of 0. Then, the point registration problem is turned into finding a transformation that minimizes the objective function

$$E(\boldsymbol{T}) = \frac{1}{2} \sum\nolimits_{j=0}^{m} \left\| f(\boldsymbol{T}\boldsymbol{y}_j, \mathcal{X}) \right\|^2 . \qquad (2)$$

IV. Preliminary Knowledge

A. Gaussian Process Implicit Surfaces

According to the implict function theorem, the implicit function can be formally described as $f(x) = 0$, where f is a scalar function that takes an input $x \in \mathbb{R}^d$ and results in a $d-1$ dimensional manifold $\mathcal{S}$. For point registration, we will constrain d to dimension 3. Many existing methods can be used to describe the implicit surface, e.g., trimmed B-splines [20], transcendental functions, or thin plate splines [21]. In this work, Gaussian Process Implicit Surfaces [22] will be used to describe the 3D mesh surface. Unlike thin plate splines, the Radial Basis Function (RBF) kernel can only describe a local patch whose distribution will rapidly converge to zero when a point is not near the queried point and not on the surface, which however is in contrast to our assumption. As an alternative, thin plate splines will be selected via

$$k_{ij}(r) = 2r^3 - 3Cr^2 + C^3\,, \tag{3}$$

with a maximum radius of C inside the training point cloud sets $\mathcal{X}$. The parameter r is the distance between two points. Gaussian Process Implicit Surfaces can be considered as a standard regression problem, which is expressed as $\boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{0}, \mathbf{K}(\mathcal{X},\mathcal{X}) + \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{\sigma}\right)$, where $\mathcal{N}$ denotes a normal distribution with the mean $\mu(\mathcal{X}) = \boldsymbol{0}$ and variance $\boldsymbol{K}(\mathcal{X},\mathcal{X}) + \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{\sigma}$. The covariance matrix $\boldsymbol{K}(.,.)$ consists of $k_{ij}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $\boldsymbol{\sigma}$ corresponds to the noise.

The goal therefore is to predict the value by evaluating the following formulas for the queried point $\mathbf{y}_j$:

$$f_*(\boldsymbol{y}_j|\mathcal{X}) = \boldsymbol{k}_*^{\mathrm{T}}\left[\boldsymbol{K} + \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{\sigma}\right]^{-1}\boldsymbol{f} = \boldsymbol{k}_*^{\mathrm{T}}\boldsymbol{\alpha} \tag{4}$$

$$V(\boldsymbol{y}_j|\mathcal{X}) = k(\boldsymbol{y}_j,\boldsymbol{y}_j) - \boldsymbol{k}_*^{\mathrm{T}}\left[\boldsymbol{K} + \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{\sigma}\right]^{-1}\boldsymbol{k}_*\,. \tag{5}$$

The kernel function $\boldsymbol{k}_*(\mathcal{X},\boldsymbol{y}_j)$ is used to describe the correlation between source point set $\mathcal{X}$ and target point set $\boldsymbol{y}_j$. The function value $f_*(\boldsymbol{y}_j|\mathcal{X})$ is a prediction of $\boldsymbol{y}_j$ with the corresponding variance, which is expressed as $V(\boldsymbol{y}_j|\mathcal{X})$ and can be used to evaluate the reliability of the predicted value. For simplification, this value is not included inside the objective function. The 3D model points according to SDF are denoted as $\left\{\boldsymbol{X}_i \in \mathcal{X}^0 \mid \boldsymbol{X}_i = \{\boldsymbol{x}_i, \sigma_i, 0\}\right\}$. To aid the training of an implicit function, two additional point sets will be created: $\left\{\boldsymbol{X}_i \in \mathcal{X}^1 \mid \boldsymbol{X}_i = \{\boldsymbol{x}_i, \sigma_i, 1\}\right\}$ lies outside the surface and $\left\{\boldsymbol{X}_i \in \mathcal{X}^{-1} \mid \boldsymbol{X}_i = \{\boldsymbol{x}_i, \sigma_i, -1\}\right\}$ lies inside the surface. How to generate these two additional point sets is presented in [23]. The final training point sets consist of $\mathcal{X} = \mathcal{X}^0 \cup \mathcal{X}^1 \cup \mathcal{X}^{-1} \in \mathbb{R}^{n\times 3}$.

After modeling the training point sets as the GPIS in the sense of a manifold, the objective function E can be further expressed as

$$f_j(\boldsymbol{y}_j|\mathcal{X}) = \boldsymbol{k}_j(\mathcal{X},\boldsymbol{y}_j)^{\mathrm{T}}\boldsymbol{\alpha} = \sum\nolimits_{i=0}^{n} k(\boldsymbol{x}_i,\boldsymbol{y}_j)\alpha_i \tag{6}$$

$$E = \frac{1}{2}\sum\nolimits_{j=0}^{m} f_j^2(\boldsymbol{y}_j|\mathcal{X}) = \sum\nolimits_{j=0}^{m} \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{k}_j\boldsymbol{k}_j^{\mathrm{T}}\boldsymbol{\alpha}\,, \tag{7}$$

where m is the number of points in the target point cloud and n is the number of points in the source point cloud. f_j is the predictive value given by $\boldsymbol{y}_j$ that is equal to zero if the target point lies on the mesh surface. The main benefits of this formulation are that no corresponding points between two point sets are required and that it converts the problem into a standard nonlinear squares problem, which can be solved by standard convex solvers.

B. Lie Algebra for Optimization

The transformation matrix $\boldsymbol{T}$ consists of a rotation matrix $\boldsymbol{R}$ and a translation $\boldsymbol{t}$, which can be interpreted in terms of Lie groups SE(3) [24] with the corresponding Lie algebra $\mathfrak{se}(3)$. It can also be converted to a Lie group by utilizing the exponential map $\boldsymbol{T} = \exp(\boldsymbol{\xi}^\wedge)$, where $\boldsymbol{\xi}^{\mathrm{T}} = \begin{bmatrix}\boldsymbol{\rho}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{1\times 6}$ with $\boldsymbol{\rho} \in \mathbb{R}^3$ and $\boldsymbol{\phi} \in \mathfrak{so}(3)$. In this optimized formulation, the target point set is transferred by $\boldsymbol{T}$ to align the source point set, which is embedded into $f(\mathcal{X})$. The gradient-based optimization uses the Jacobian matrix to search for the minimum solution. We apply the perturbation method for calculating the gradient of $\boldsymbol{T}$ and consider the directional of the derivative of $\boldsymbol{T}$ with respect to the perturbation $\boldsymbol{\epsilon} \in \mathbb{R}^6$, which can be computed as

$$\boldsymbol{T} = \exp\left(\boldsymbol{\epsilon}^\wedge\right)\boldsymbol{T}_{\mathrm{op}} \approx (\boldsymbol{I} + \boldsymbol{\epsilon}^\wedge)\boldsymbol{T}_{\mathrm{op}}\,, \quad \frac{\partial(\boldsymbol{T}\hat{\boldsymbol{y}}_j)}{\partial\boldsymbol{\epsilon}} = (\boldsymbol{T}\hat{\boldsymbol{y}}_j)^{\odot}\,. \tag{8}$$

The operator $(.)^{\odot} : \mathbb{R}^4 \to \mathbb{R}^{4\times 6}$ is defined as $[\boldsymbol{\varepsilon}^{\mathrm{T}}, \eta]^{\mathrm{T}\odot} = \begin{bmatrix}\eta\boldsymbol{I} & -\boldsymbol{\varepsilon}^\wedge \\ \boldsymbol{0}^{\mathrm{T}} & \boldsymbol{0}^{\mathrm{T}}\end{bmatrix}$, where $\boldsymbol{\varepsilon} \in \mathbb{R}^3$ and η is a scalar that maps the vector space to a higher manifold. For simplification, we omit $\boldsymbol{x}_i$ in $k_i(\boldsymbol{x}_i, \boldsymbol{T}\hat{\boldsymbol{y}}_j)$ and use $k_i(\boldsymbol{T}\hat{\boldsymbol{y}}_j)$ instead. By integration of the perturbation formula (8) with the first order Taylor series in kernel function, it can be approximated as

$$\begin{aligned} k_i(\boldsymbol{T}\hat{\boldsymbol{y}}_j) &= k_i\big(\exp(\boldsymbol{\epsilon}^\wedge)\boldsymbol{T}_{\mathrm{op}}\hat{\boldsymbol{y}}_j\big) \approx k_i\big((\boldsymbol{I} + \boldsymbol{\epsilon}^\wedge)\boldsymbol{T}_{\mathrm{op}}\hat{\boldsymbol{y}}_j\big) \\ &\approx k_i(\boldsymbol{T}_{\mathrm{op}}\hat{\boldsymbol{y}}_j) + \left(\frac{\partial k_i}{\partial \boldsymbol{y}_j}\right)^{\mathrm{T}}\bigg|_{\boldsymbol{y}_j = \boldsymbol{T}_{\mathrm{op}}\hat{\boldsymbol{y}}_j} \frac{\partial \boldsymbol{T}\hat{\boldsymbol{y}}_j}{\partial \boldsymbol{\epsilon}}\boldsymbol{\epsilon} \\ &\approx \beta_i + \boldsymbol{\delta_i}^{\mathrm{T}}\boldsymbol{\epsilon}\,, \end{aligned} \tag{9}$$

where $\beta_i = k_i(\boldsymbol{T}_{\mathrm{op}}\hat{\boldsymbol{y}}_j) \in \mathbb{R}$, $\left(\frac{\partial k_i}{\partial \boldsymbol{y}_j}\right)^{\mathrm{T}} \in \mathbb{R}^{1\times 4}$ and $\boldsymbol{\delta_i}^{\mathrm{T}} = \left(\frac{\partial k_i}{\partial \boldsymbol{y}_j}\right)^{\mathrm{T}}\Big|_{\boldsymbol{y}_j = \boldsymbol{T}_{\mathrm{op}}\hat{\boldsymbol{y}}_j}\left((\boldsymbol{T}\hat{\boldsymbol{y}}_j)^{\odot}\right) \in \mathbb{R}^{1\times 6}$. The derivative of the kernel function $\frac{\partial k_i}{\partial \boldsymbol{y}_j}$ in (9) is expressed as

$$\frac{\partial k_i}{\partial \boldsymbol{y}_j} = \frac{\partial k_i}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \boldsymbol{y}_j} = 6(r_{ij} - C)(\boldsymbol{y}_j - \boldsymbol{x}_i)\,.$$

V. GPIS-Based S2S Registration Algorithm

For the partially overlapping situation, the transformation matrix $\boldsymbol{T}$ is considered in the kernel's corresponding distance function with $r = \|\boldsymbol{x}_i - \boldsymbol{R}\hat{y}_j - \boldsymbol{t}\| = \|\boldsymbol{x}_i - \boldsymbol{T}\hat{\boldsymbol{y}}_j\|$. Therefore, the equation (6) is updated as $f_j = \boldsymbol{k}(\mathcal{X}, \boldsymbol{T}\hat{y}_j)^{\mathrm{T}}\boldsymbol{\alpha}$. By combining this with the approximation of the kernel function (9), the objective function (7) can be further simplified as

$$f_j = \sum\nolimits_{i=0}^{n}(\beta_i + \boldsymbol{\delta}_i^{\mathrm{T}}\boldsymbol{\epsilon})\,\alpha_n = \boldsymbol{k}^{\mathrm{T}}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})\,\boldsymbol{\alpha} \tag{10}$$

$$E(\boldsymbol{T}) = \frac{1}{2}\sum\nolimits_{j=0}^{m}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{k}\left(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon}\right)\boldsymbol{k}^{\mathrm{T}}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})\,\boldsymbol{\alpha}\,, \tag{11}$$

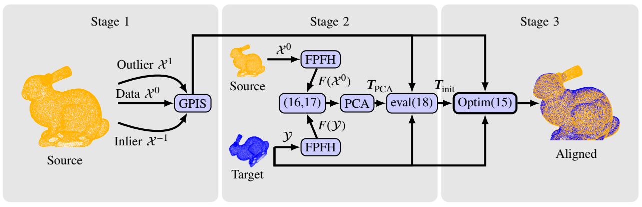


Fig. 2: The algorithm consists of three stages: In the first stage, two additional point sets $\mathcal{X}^1$ and $\mathcal{X}^2$ are created to augment the original point set as $\mathcal{X} = \mathcal{X}^0 \cup \mathcal{X}^1 \cup \mathcal{X}^{-1}$, which is used to form the implicit function GPIS. In stage two, we compute FPFH for each point in the source and target point set and a cross-checking is executed to identify a corresponding group. The PCA is utilized to compute the initial transformation by evaluating the objective function. In the last stage, the alignment is optimized by a convex solver.

with $\boldsymbol{k}^{\mathrm{T}}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon}) = \begin{bmatrix}\beta_1 + \boldsymbol{\delta}_1^{\mathrm{T}}\boldsymbol{\epsilon}, & \cdots, & \beta_n + \boldsymbol{\delta}_n^{\mathrm{T}}\boldsymbol{\epsilon}\end{bmatrix} \in \mathbb{R}^{1\times n}$. As a result, (11) is converted to a nonlinear quadratic equation with the approximated nonlinear kernel function $\boldsymbol{k}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})$ and the argument for this optimized problem is changed from the Lie group $\boldsymbol{T} \in \mathrm{SE}(3)$ to the perturbation variable of the Lie algebra $\boldsymbol{\epsilon} \in \mathfrak{se}(3)$.

A. Gradient of Objective Function

By taking the derivative of $\boldsymbol{J}$ with respect to $\boldsymbol{\epsilon}^{\mathrm{T}}$, we get

$$\begin{aligned}\frac{\partial E(\boldsymbol{T})}{\partial \boldsymbol{\epsilon}^{\mathrm{T}}} &= \sum_{j=0}^{m} \frac{\partial\left(\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{k}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})\right)}{\partial \boldsymbol{\epsilon}^{\mathrm{T}}}\boldsymbol{k}^{\mathrm{T}}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})\boldsymbol{\alpha} \\ &= \sum_{j=0}^{m}\left(\sum_{i=0}^{n}\frac{\partial(\alpha_i\beta_i + \alpha_i\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\delta}_i)}{\partial\boldsymbol{\epsilon}^{\mathrm{T}}}\right)\boldsymbol{k}^{\mathrm{T}}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})\boldsymbol{\alpha} \\ &= \sum_{j=0}^{m}\left(\sum_{i=0}^{n}\alpha_i\boldsymbol{\delta}_i\right)\boldsymbol{k}^{\mathrm{T}}(\boldsymbol{\beta}, \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{\epsilon})\boldsymbol{\alpha} \\ &= \sum_{j=0}^{m}\boldsymbol{\Delta}_j\left(\sum_{i=0}^{n}\beta_i\alpha_i + \boldsymbol{\delta}_i^{\mathrm{T}}\boldsymbol{\epsilon}\alpha_i\right), \qquad (12)\end{aligned}$$

where $\boldsymbol{\Delta}_j$ is defined as $\sum_{i=0}^{n}\alpha_i\boldsymbol{\delta}_i \in \mathbb{R}^{6\times 1}$ and can capture the surface's curvature by summing up all gradients in the kernel function. To get the optimum perturbation $\boldsymbol{\epsilon}^{\star}$ at the current position, the formula $\frac{\partial E(\boldsymbol{T})}{\partial\boldsymbol{\epsilon}^{\mathrm{T}}}$ is forced to be zero, leading to

$$\begin{aligned}\sum_{j=0}^{m}\boldsymbol{\Delta}_j\sum_{i=0}^{n}\boldsymbol{\delta}_i^{\mathrm{T}}\alpha_i\boldsymbol{\epsilon}^{\star} &= -\sum_{j=0}^{m}\boldsymbol{\Delta}_j\sum_{i=0}^{n}\beta_i\alpha_i \\ \boldsymbol{J}\boldsymbol{\epsilon}^{\star} &= -\sum_{j=0}^{m}\boldsymbol{\Delta}_j\sum_{i=0}^{n}\beta_i\alpha_i \qquad (13) \\ \boldsymbol{\epsilon}^{\star} &= -\boldsymbol{J}^{-1}\sum_{j=0}^{m}\boldsymbol{\Delta}_j\sum_{i=0}^{n}\beta_i\alpha_i\,, \qquad (14)\end{aligned}$$

with $\boldsymbol{J} = \sum_{j=0}^{m}\boldsymbol{\Delta}_j\boldsymbol{\Delta}_j^{\mathrm{T}} \in \mathbb{R}^{6\times 6}$. $\boldsymbol{T}$ is therefore updated as

$$\boldsymbol{T}_{\mathrm{op},h} \leftarrow \exp\left((\boldsymbol{\epsilon}^{\star})^{\wedge}\right)\boldsymbol{T}_{\mathrm{op},h-1}, \qquad (15)$$

which captures the local structural manifold by means of the Lie algebra. The optimization process follows the principle of the Gauss-Newton algorithm. We can further adapt $\boldsymbol{J}$ as $\sum_{j=0}^{m}\boldsymbol{\Delta}_j\boldsymbol{\Delta}_j^{\mathrm{T}} + \lambda\,\mathrm{diag}(\boldsymbol{S})$, which is the LM algorithm.

B. Initial Alignment Using PCA and FPFH

A good initial guess for the optimization is important in order to guarantee a good result and run-time. We present a new method for computing the initial alignment by employing the PCA and FPFH [11] algorithms. First, a FPFH is calculated for each point in the two point sets, which are referred to as $F(\mathcal{X}^0)$ and $F(\mathcal{Y})$. We then embed $F(\mathcal{X}^0)$ into a k-d tree $\mathrm{Kd}_{F(\mathcal{X}^0)}$ and a nearest neighbor search is performed for each feature $F(\boldsymbol{y}_j) \in F(\mathcal{Y})$, such that $\mathcal{G}_1$ is a group pair set of the results $\boldsymbol{x}_{i|j}$ for the queries $\boldsymbol{y}_j$:

$$\mathcal{G}_1 = \left\{\{\boldsymbol{y}_j, \boldsymbol{x}_{i|j}\} \mid \mathrm{Kd}_{F(\mathcal{X}^0)}\left(F(\boldsymbol{y}_j)\right),\ \forall \boldsymbol{y}_j \in \mathcal{Y}\right\}. \qquad (16)$$

Subsequently, we embed $F(\mathcal{Y})$ into another k-d tree $\mathrm{Kd}_{F(\mathcal{Y})}$ and perform a nearest neighbor search for each result stored in $\mathcal{G}_1$, such that $\mathcal{G}_2$ is a group pair set of the results $\boldsymbol{y}_{j|i}$ for the queries $\boldsymbol{x}_{i|j}$ that are stored in $\mathcal{G}_1$:

$$\mathcal{G}_2 = \left\{\{\boldsymbol{x}_{i|j}, \boldsymbol{y}_{j|i}\} \mid \mathrm{Kd}_{F(\mathcal{Y})}\left(F(\boldsymbol{x}_{i|j})\right),\ \forall \boldsymbol{x}_{i|j} \in \mathcal{G}_1\right\} \qquad (17)$$

We only keep the subset $\mathcal{G}_1 \cap \mathcal{G}_2$ that contains bidirectional nearest neighbors and refer to these as $\mathcal{X}'$ and $\mathcal{Y}'$. Statistical analysis techniques are then applied to remove any outliers in these groups [25]. The final selected points are grouped together and are denoted as $\mathcal{X}_{\text{group,FPFH}} \in \mathbb{R}^{n_{\text{FPFH}}\times 3}$ and $\mathcal{Y}_{\text{group,FPFH}} \in \mathbb{R}^{m_{\text{FPFH}}\times 3}$. After this, a PCA is used to compute the initial transformation $\boldsymbol{T}_{\text{PCA}}$ between $\mathcal{X}_{\text{group,FPFH}}$ and $\mathcal{Y}_{\text{group,FPFH}}$. Note, that $\boldsymbol{T}_{\text{PCA}}$ has four different possibilities according to the right-hand rule (Fig. 3). By evaluating the formula

$$\boldsymbol{T}_{\text{init}} \leftarrow \min_{k\in\{0,\cdots 3\}}\sum_{j=0}^{m}\left\|f(\boldsymbol{T}_{\text{PCA},k}\hat{\boldsymbol{y}}_j, \mathcal{X})\right\|^2, \qquad (18)$$

we select the transformation matrix $\boldsymbol{T}_{\text{init}}$ that has the smallest function value. The whole process is illustrated in Fig. 2 and the algorithm is summarized in Alg. 1.

VI. Evaluation

In order to compare our algorithm against the state of the art, we evaluated it against other registration algorithms

Algorithm 1 Optimization of transformation matrix by Gauss-Newton/Levenberg-Marquardt algorithm

```
Require: $\mathcal{X}$, $\mathcal{Y}$, $H$
 1: Modeling GPIS $f(\mathcal{X})$                                   ▷ Section IV-A
 2: Compute FPFH features $F(\mathcal{X}^0)$ and $F(\mathcal{Y})$     ▷ Section V-B
 3: Calculate $\mathcal{G}_1$ and $\mathcal{G}_2$                     ▷ (16), (17)
 4: $\{\mathcal{X}_{\text{group,FPFH}}, \mathcal{Y}_{\text{group,FPFH}}\} \leftarrow \text{StaticalRemove}(\mathcal{G}_1 \cap \mathcal{G}_2)$
 5: $\boldsymbol{T} \leftarrow \text{PCA}(\mathcal{X}_{\text{group,FPFH}}, \mathcal{Y}_{\text{group,FPFH}})$   ▷ (18)
 6: $\boldsymbol{T}_{\text{op},0} \leftarrow \arg\min \sum_{j=0}^{m} f(\mathcal{X}, \boldsymbol{T}\boldsymbol{y}_j)$
 7: for $h = 1 : H$ do
 8:     Approximate $k_i(\boldsymbol{y}_j)\ \forall \boldsymbol{y}_j \in \mathcal{Y}$          ▷ (9)
 9:     if Gauss-Newton then
10:         Set $\boldsymbol{J} = \sum_{j=0}^{m}\boldsymbol{\Delta}_j\boldsymbol{\Delta}_j^{\mathrm{T}}$
11:     end if
12:     if Levenberg-Marquardt then
13:         Set $\boldsymbol{J} = \sum_{j=0}^{m}\boldsymbol{\Delta}_j\boldsymbol{\Delta}_j^{\mathrm{T}} + \lambda\,\text{diag}(\boldsymbol{S})$
14:     end if
15:     calculate $\boldsymbol{\epsilon}^{\star} = -\boldsymbol{J}^{-1}\sum_{j=0}^{m}\boldsymbol{\Delta}_j\sum_{i=0}^{n}\beta_i\alpha_i$   ▷ (14)
16:     Update $\boldsymbol{T}_{\text{op,h}}$                                 ▷ (15)
17:     if $\|\boldsymbol{\epsilon}^{\star}\|_F \le \epsilon$ then break
18:     end if
19: end for
20: return $\boldsymbol{T}_{\text{op}^{\star}}$
```

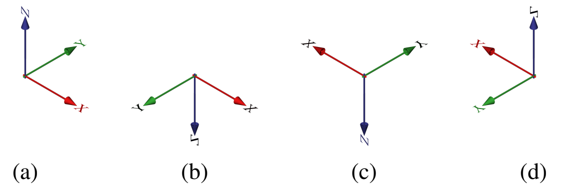


Fig. 3: The four possible coordinate systems for the PCA when computing the initial transformation matrix.

regarding accuracy RMSE and time on the Stanford 3D Scanning Repository's Happy Buddha, Stanford Bunny, and Chinese Dragon [26] as well as Blender's Suzanne model. PCL-ICP [25] is the standard implementation of the ICP algorithm in the Point Cloud Library, which is a local registration algorithm as it relies on a good initial alignment. SAC-IA-ICP [11] employs FPFH to get the initial alignment and then uses ICP to iteratively align the two point clouds. It is therefore considered a global point registration algorithm. GoICP and its variant GoICPT with trimming [14] are global registration algorithms that use the BnB algorithm in their implementations and also support partial overlapping point registration. Global registration RANSAC (Gl.RANSAC) [27] requires no initial alignment. Instead, it utilizes RANSAC for the initialization alignment by searching corresponding points in the FPFH feature space. Fast Global Registration (FGR) is another registration algorithm that utilizes FPFH for searching corresponding points. The algorithm presented in this paper is labeled GPIS-S2SPR. In this section, three different experiments were conducted. In order to reduce the computational burden, the tested point sets were downsampled for every algorithm into small scale numbers (1500–2500) by using voxel filtering. All evaluations were performed on a laptop with a 2.6 GHz Intel Core i7-6700HQ and 16 GB of RAM.

A. RMSE for Random Transformations

For exploring the capability of our algorithm, we evaluated algorithms using the Stanford Bunny point set with 50 290 points without Gaussian noise and only partial overlap (85%). We reduced the number points by applying a voxel grid filter with a size of 0.005 [25]. We ran each algorithm on a set of 40 random transformation matrices (Fig. 4).

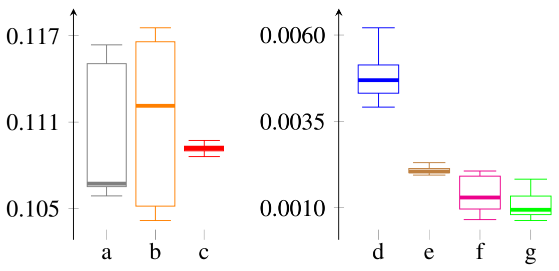


Fig. 4: RMSE for the Stanford Bunny with partial overlap (85%) without Gaussian noise: (a) GoICP, (b) GoICPT (10%), (c) PCL-ICP, (d) Gl.RANSAC, (e) SAC-IA-ICP, (f) FGR, (g) GPIS-S2SPR.

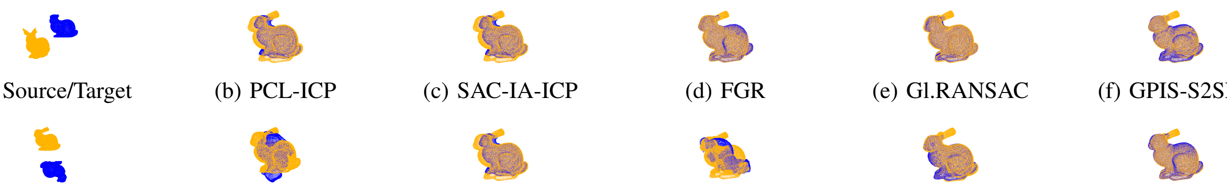
From the results, we can see that GoICP and its variant GoICP with trimming (GoICPT) (10%), as suggested in [14], performed worse in this experiment with median values of 0.110 and 0.108. PCL-ICP was evaluated using an identity matrix as initialization and behaved slightly better with a median value of 0.109 and a smaller variance in comparison to GoICP. Gl.RANSAC showed significant improvement with a median value of 0.005 and SAC-IA-ICP achieved a median value of 0.002. FGR achieved a median value of 0.001. In contrast to the previous algorithms however, its mean value of 0.017 deviated from the median and is much higher. This is due to its lack of robustness in handling large rotational changes, which is further evaluated in the next experiment (Section VI-B). Our algorithm showed the best overall performance, with a median value of 0.001 and a small variance.

B. Rotation and Translation Invariance

Rotation and translation invariance are essential factors for point registration. We conducted two experiments with the Stanford Bunny dataset to evaluate these two properties. For the first experiment, we rotated the source point set 50° around the y axis and translated it with a vector of $[0.1, 0.2, 0.3]$, as illustrated in the first row of Fig. 5. In the second one, we rotated the point set 180° around the z axis without translation, as shown in the second row of Fig. 5. The same initial alignment is used for each algorithm in both cases. We repeated both experiments 40 times. The best results are shown in Fig. 5. PCL-ICP and FGR show entirely different behaviors in these two cases, while they failed with a more than 0.1 RMSE value in the second case. For these two algorithms, we conducted further experiments with different rotations. FGR failed with a high probability for high rotation values and we therefore conclude, that FGR is not rotation invariant. SAC-IA-ICP showed no significant difference in both experiments, achieving roughly the same mean value of 0.009. Gl.RANSAC also showed similar performance in both cases with mean values of 0.003 and 0.006. In this experiment, rotation and translation showed no significant effect in our GPIS-S2SPR approach, with an approximate RMSE of 0.002.

C. Noise and Overlap Robustness

We evaluated the algorithms on all four point sets with the number of points varying from 30 000 to 50 000. Furthermore, we applied three different levels of noise based on a Gaussian distribution with variances set to 0, 0.00025, and 0.0005, respectively. We also evaluated the capability of point registration in a partially overlapping scenario, where only a subset of the points from the source point cloud is used for the target point cloud. Three different overlap factors were used in the experiments: 100%, 85%, and 65%. Furthermore, we used an identity matrix for the initial alignment in each test to maintain identical conditions. Each algorithm was executed 40 times for each configuration, leading to a total of 1440 times for all possible combinations.


(a) Source/Target (b) PCL-ICP (c) SAC-IA-ICP (d) FGR (e) Gl.RANSAC (f) GPIS-S2SPR

(g) Source/Target (h) PCL-ICP (i) SAC-IA-ICP (j) FGR (k) Gl.RANSAC (l) GPIS-S2SPR

Fig. 5: Stanford Bunny together with alignment results of selected algorithms for (a)–(f) 50° rotation around the y axis and small translation, (g)–(l) 180° rotation around the z axis with translation set to zero.

TABLE I: Benchmark results for all algorithms on four different point sets with three levels of Gaussian noise and three different overlap factors. The best RMSE value ϵ for each configuration is highlighted in *green*.

		noise = 0.00000						noise = 0.00025						noise = 0.00050					
		1.00		0.85		0.65		1.00		0.85		0.65		1.00		0.85		0.65	
		ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t	ϵ	t
Bunny [26]	PCL-ICP	0.081	0.5	0.090	0.4	0.067	0.3	0.046	0.4	0.082	0.6	0.046	0.3	0.069	0.4	0.107	0.6	0.073	0.3
	GoICP	0.115	20.3	0.110	20.1	0.097	20.0	0.046	20.1	0.024	20.1	0.063	20.2	0.089	20.1	0.101	20.1	0.046	20.1
	GoICPT	0.100	21.5	0.108	21.5	0.099	21.4	0.106	21.5	0.104	21.7	0.098	21.4	0.103	21.4	0.109	21.7	0.103	21.4
	SAC-IA-ICP	0.001	6.1	0.002	5.6	0.010	7.1	0.001	6.6	0.002	6.1	0.011	8.2	0.001	7.0	0.003	6.4	0.011	7.9
	Gl.RANSAC	0.001	1.6	0.005	1.7	0.005	2.3	0.001	1.7	0.005	1.8	0.005	1.9	0.001	1.7	0.005	1.9	0.005	2.3
	FGR	0.017	0.4	0.009	0.4	0.017	0.3	0.004	0.4	0.007	0.4	0.020	0.3	0.004	0.4	0.007	0.4	0.015	0.3
	GPIS-S2SPR	0.001	0.6	0.001	0.5	0.001	0.5	0.001	0.5	0.001	0.6	0.002	0.5	0.001	0.5	0.002	0.6	0.002	0.7
Suzanne [28]	PCL-ICP	0.134	0.8	0.108	0.8	0.116	0.7	0.049	0.6	0.127	1.3	0.095	0.6	0.131	0.9	0.115	0.6	0.106	0.9
	GoICP	0.089	23.5	0.055	21.6	0.086	22.0	0.089	21.8	0.081	21.8	0.089	21.8	0.064	21.8	0.072	21.7	0.117	21.5
	GoICPT	0.092	23.2	0.059	21.5	0.084	21.5	0.079	21.6	0.071	21.7	0.097	21.7	0.045	21.5	0.091	21.6	0.062	21.5
	SAC-IA-ICP	0.001	11.2	0.005	10.6	0.018	10.3	0.001	11.7	0.006	10.7	0.027	11.4	0.001	12.4	0.006	11.5	0.018	11.1
	Gl.RANSAC	0.014	1.7	0.018	1.7	0.040	2.0	0.016	1.8	0.011	1.8	0.039	2.3	0.013	1.9	0.025	1.9	0.022	2.7
	FGR	0.049	0.6	0.039	0.6	0.060	0.5	0.070	0.6	0.049	0.6	0.061	0.5	0.046	0.7	0.071	0.6	0.052	0.6
	GPIS-S2SPR	0.003	0.8	0.002	1.1	0.002	1.1	0.001	4.3	0.002	2.1	0.002	1.8	0.002	1.3	0.003	1.2	0.002	1.6
Dragon [26]	PCL-ICP	0.087	0.5	0.100	0.3	0.079	0.4	0.090	0.2	0.065	0.3	0.096	0.4	0.071	0.5	0.082	0.5	0.101	0.4
	GoICP	0.017	21.6	0.022	21.7	0.018	21.4	0.034	21.5	0.035	21.7	0.021	21.5	0.018	21.5	0.053	21.5	0.033	21.6
	GoICPT	0.022	21.4	0.014	21.4	0.021	20.0	0.011	19.9	0.009	20.0	0.084	20.1	0.013	20.1	0.044	20.2	0.017	20.2
	SAC-IA-ICP	0.001	5.9	0.003	5.3	0.009	4.8	0.001	6.0	0.003	5.4	0.009	4.9	0.001	6.5	0.004	5.8	0.009	5.2
	Gl.RANSAC	0.001	2.1	0.005	2.6	0.005	2.9	0.001	2.3	0.005	2.5	0.004	2.8	0.001	2.6	0.005	2.5	0.005	2.8
	FGR	0.012	0.5	0.022	0.5	0.024	0.4	0.012	0.5	0.016	0.4	0.017	0.4	0.021	0.5	0.015	0.5	0.013	0.4
	GPIS-S2SPR	0.002	0.7	0.002	0.9	0.002	1.0	0.002	0.8	0.002	0.9	0.003	1.0	0.002	0.7	0.002	1.0	0.003	0.9
Happy Buddha [26]	PCL-ICP	0.094	0.2	0.086	0.4	0.110	0.3	0.095	0.5	0.076	0.2	0.071	0.2	0.124	0.4	0.047	0.4	0.064	0.4
	GoICP	0.043	21.5	0.032	21.4	0.085	21.3	0.075	21.4	0.050	21.3	0.052	21.3	0.051	21.4	0.067	21.5	0.012	21.4
	GoICPT	0.013	20.2	0.025	20.1	0.028	20.2	0.016	20.2	0.031	20.2	0.022	20.1	0.023	20.0	0.031	19.9	0.020	20.0
	SAC-IA-ICP	0.001	5.5	0.007	5.2	0.009	6.3	0.001	6.2	0.014	7.3	0.017	7.4	0.001	6.7	0.011	7.5	0.015	6.2
	Gl.RANSAC	0.002	1.1	0.005	1.3	0.006	1.8	0.002	1.0	0.005	1.4	0.006	1.6	0.002	1.1	0.005	1.5	0.005	1.8
	FGR	0.022	0.4	0.022	0.4	0.019	0.3	0.016	0.4	0.019	0.4	0.015	0.3	0.024	0.4	0.022	0.4	0.021	0.3
	GPIS-S2SPR	0.003	1.0	0.002	0.8	0.002	0.8	0.001	0.6	0.003	0.7	0.002	0.8	0.002	0.7	0.002	0.7	0.003	1.2

TABLE II: Mean RMSE over all noise levels and overlap factors for all point sets and algorithms. The best result is highlighted in *green*.

Data	PCL-ICP	GoICP	GoICPT	SAC-IA-ICP	Gl.RANSAC	FGR	**GPIS-S2SPR**
Bunny	0.073	0.077	0.103	0.005	0.004	0.011	0.001
Suzanne	0.109	0.083	0.076	0.009	0.022	0.055	0.002
Dragon	0.086	0.028	0.013	0.004	0.004	0.017	0.002
Buddha	0.085	0.052	0.052	0.008	0.004	0.020	0.002
Mean	0.088	0.060	0.061	0.007	0.009	0.026	0.002

The results with individual RMSE values ϵ and runtime t for each configuration are listed in Table I. The algorithms GoICP and GoICPT (10% trimming) consistently showed the worst performance in all test cases with regard to the mean value of RMSE and total computation time. The BnB algorithm in these algorithms is very expensive to compute and the constant switch between ICP and BnB was not able to achieve a global optimum solution. PCL-ICP was able to converge very fast but is susceptible to local minima. SAC-IA-ICP achieved the best performance in case of an overlapping factor of 100% with an RMSE of 0.001. However, the RMSE value increased drastically to 0.01 in case of a partial overlap, Gl.RANSAC explores the corresponding points in terms of FPFH and showed similar performance. FGR converged very fast but showed a bad performance in all experiments for the reasons explained in Section VI-B. Our GPIS-S2SPR algorithm does not rely on identifying corresponding points and is therefore stable for different transformations. In this experiment, the RMSE for GPIS-S2SPR is stable in terms of the overlapping factor and noise for all point sets. As it is able to consider noise in its formulation, the RMSE is approximated equal to 0.002 for all noise levels. Table II shows the total RMSE computed over all possible combinations for each point set and algorithm. GPIS-S2SPR showed the best overall performance. In terms of computation time, it is not always the fastest approach, since GPIS is time-consuming due to the computation of an inverse of the covariance matrix with a complexity of $\mathcal{O}(m^3)$.

D. *Gearbox Assembly Application*

We explore the capability of our algorithm by using real point cloud data captured by the system shown in Fig. 1b,

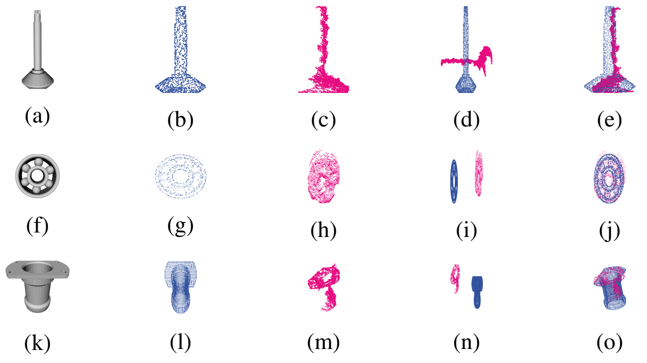


Fig. 6: The first column shows the CAD models of the gearbox components tree, bearing, and pipe. The second one shows the mesh sampled point cloud from the CAD model for each component. The cluster extracted from the point cloud scene (Fig. 1b) is shown in column three. Column four shows the initial source and target point cloud. The alignment results of GPIS-S2SPR are shown in the last column.

where the task is to assemble a mechanical gearbox. In order to grasp the objects on the table with the attached parallel gripper, we require a 6D pose transformation.

The gear box assembly task consists of four parts, a mechanical tree (Fig. 6a), two ball bearings (Fig. 6f), and a mechanical pipe (Fig. 6k). All parts are available in the form of CAD models. For the 6D pose estimation, we apply mesh sampling to acquire a detailed point cloud (PC) for each CAD model, shown in Fig. 6b, 6g, and 6l. The noisy point cloud data shown in Fig. 6c, 6h, and 6m is extracted from the actual data captured by the camera sensor as demonstrated in Fig. 1b. We combine the Euclidean Cluster Extraction and Region growing segmentation from PCL [25] to extract each component from the point cloud scene. The initial position relationship between the source and target point cloud is shown in the fourth column of Fig. 6, where the tree is lying on the table and the pipe is rotated by 90°. As only one 3D camera is used, we can only get a partial view of the objects. The noise added by the camera sensor is not a Gaussian distribution. We evaluate our algorithm with these three components, the results are shown in the last column of Fig. 6. Although the tree and ball bearing are two highly symmetrical components, the algorithm can match the bottom and the upper part with an error of 0.004 In the case of the pipe alignment, the pipe cluster has two disconnected parts and only an approximated 25% of object information is available, which further increases the complexity. Our algorithm can match the objects with an error of 0.009

E. Evaluation with Scanned Datasets

To further verify our algorithm, we evaluate the point sets from semantic-8 [29] and Urban Scenes Velodyne Point Cloud Dataset [30]. The corresponding results are shown in Fig. 7. We compare our algorithm with PCL-ICP in Fig. 7a, where the RMSE of PCL-ICP is 48 times that of our algorithm. In Fig. 7b–7h, each sub-figure consists of two images, where the left one is the initial state, and the right one is the result of point registration. It can be seen that our algorithm can work in different scenarios, such as urban scenes [30] and different kinds of buildings. Furthermore, we evaluate our algorithm with two additional point sets from [27] and Shapenet [31], which are shown in Fig. 8. The source point sets in Fig. 8 are indicated as blue points and the target point sets as orange points. The initial setting for source and target point sets are demonstrated in Fig. 8a, 8b and 8c. From Fig. 8d, 8e, and 8f, we can see that the alignment accuracy is very high in both point sets with an RMSE value of 0.002, 0.0001, and 0.0001, respectively.

VII. Conclusion

We propose a new algorithm for a partially overlapping 3D surface registration algorithm. In this algorithm, we abandon the traditional idea of point to point or point to plane correspondence search to register the points. Instead, we view the 3D surface as a Gaussian Process Implicit Surfaces, which utilizes the signed distance function to describe three manifolds. Furthermore, we convert the point registration as a nonlinear least-squares problem to find a rigid transformation between two point sets. For accelerating the optimization process, we use a Principal Component Analysis (PCA) together with Fast Point Feature Histograms descriptors to compute the initial transformation. Moreover, we derive the Jacobian matrix by applying the Lie algebra perturbation method, which approximated the kernel function with the first-order Taylor series. The whole optimization follows the principle of Gauss-Newton algorithm. By slightly adapting the Jacobian matrix with a damping value, we can convert the algorithm to Levenberg-Marquardt solver. Our approach demonstrated a higher accuracy performance and more robust rotation invariant properties compared to state-of-the-art methods by evaluating diverse experiments.

References

[1] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. G. Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kesslar, and M. Danzer, "SMErobotics: Smart robots for flexible manufacturing," *IEEE Robotics and Automation Magazine*, vol. 26, no. 1, pp. 78–90, 2019.

[2] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth language: Representing and exchanging knowledge about actions, objects and environments," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, May 2012, pp. 1284–1289.

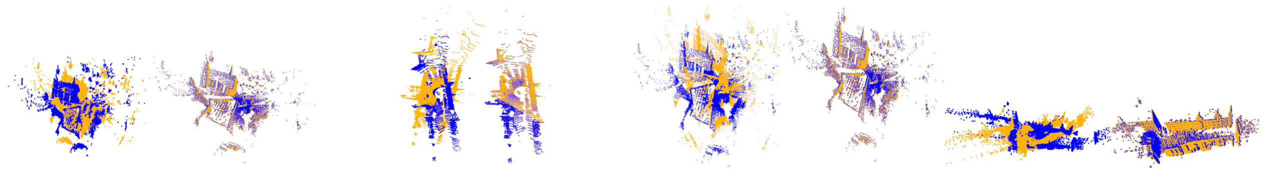
[3] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Hamburg, Germany, Sept. 2015, pp. 4197–4203.

[4] J. Feldmar and N. Ayache, "Rigid and affine registration of smooth surfaces using differential properties," in *Proc. of the European Conf. on Computer Vision*, 2005, pp. 397–406.

[5] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[6] S. Granger and X. Pennec, "Multi-scale EM-ICP: A fast and robust approach for surface registration," in *Proc. of the European Conf. on Computer Vision*, 2002, pp. 418–432.

[7] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. of Robotics: Science and Systems*, Seattle, USA, 2009.



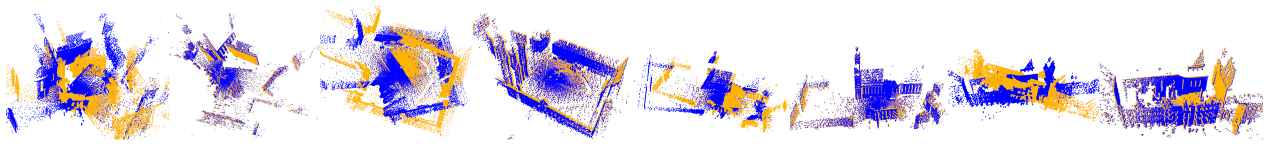


Fig. 7: Comparison between PCL-ICP and GPIS-S2SPR with examples from the large scale point cloud classification dataset Semantic-8 [29] and Urban Scenes Velodyne Point Cloud Dataset [30]. In (a), the alignment result of PCL-ICP is on the left and the result of GPIS-S2SPR on the right. From (b)–(h), left shows the initial pose of source and target point set, while right shows the result of applying GPIS-S2SPR.

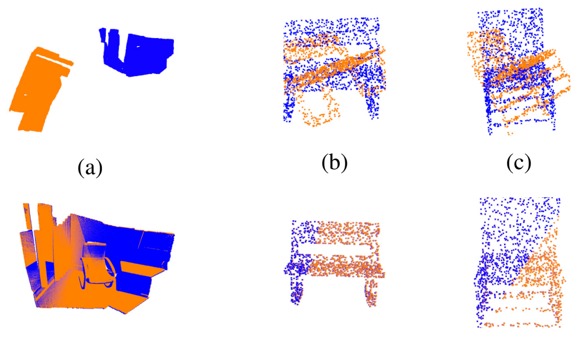


Fig. 8: Evaluation of GPIS-S2SPR with additional scanned point sets from [27] and Shapenet [31]. The first row shows the initial setting for source and target pointsets. The second row shows the alignment results.

[8] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," in *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2015, pp. 954–962.

[9] H. Zhao, "A fast sweeping method for Eikonal equations," *Mathematics of Computation*, vol. 74, no. 250, pp. 603–627, 2005.

[10] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[11] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2009, pp. 3212–3217.

[12] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[13] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus," in *Proc. of the European Conf. on Computer Vision*, 2008, pp. 500–513.

[14] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2241–2254, 2016.

[15] A. W. Fitzgibbon, "Robust registration of 2D and 3D point sets," *Image and Vision Computing*, vol. 21, no. 13-14, pp. 1145–1153, 2003.

[16] Q. Li, R. Xiong, and T. Vidal-Calleja, "A GMM based uncertainty model for point clouds registration," *Robotics and Autonomous Systems*, vol. 91, pp. 349–362, 2017.

[17] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[18] C. Papazov and D. Burschka, "Stochastic optimization for rigid point set registration," in *Proc. of the Intl. Symposium on Visual Computing*, 2009, pp. 1043–1054.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Proc. of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, USA, 2018.

[20] Y.-P. Hu and T.-C. Sun, "Moving a B-spline surface to a curve—A trimmed surface matching algorithm," *Computer-Aided Design*, vol. 29, no. 6, pp. 449–455, June 1997.

[21] J. Duchon, "Splines minimizing rotation-invariant semi-norms in Sobolev spaces," in *Constructive Theory of Functions of Several Variables*, ser. Lecture Notes in Mathematics. Springer, 2006, vol. 571, pp. 85–100.

[22] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," in *Proc. of the Workshop on Gaussian Processes in Practice*, Bletchley Park, UK, Apr. 2007.

[23] M. Li, K. Hang, D. Kragic, and A. Billard, "Dexterous grasping under shape uncertainty," *Robotics and Autonomous Systems*, vol. 75, pp. 352–364, Jan. 2016.

[24] A. Kirillov, Jr, *An introduction to Lie groups and Lie algebras*, ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2008.

[25] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, 2011.

[26] G. Turk and B. Mullins, "Large geometric models archive," https://www.cc.gatech.edu/projects/large_models/, Mar. 2019.

[27] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847 [cs.CV]*, 2018.

[28] *Blender – A 3D Modelling and Rendering Package*, Blender Foundation, 2019. [Online]. Available: http://www.blender.org

[29] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark," in *Proc. of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, 2017, pp. 91–98.

[30] K. Lai and D. Fox, "Object recognition in 3D point clouds using web data and domain adaptation," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1019–1037, 2010.

[31] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "ShapeNet: An information-rich 3D model repository," *arXiv:1512.03012 [cs.GR]*, 2015.

RightsLink

Sign in/Register

6D Pose Estimation for Flexible Production with Small Lot Sizes based on CAD Models using Gaussian Process Implicit Surfaces

Requesting permission to reuse content from an IEEE publication

Conference Proceedings:
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Author: Jianjie Lin; Markus Rickert; Alois Knoll

Publisher: IEEE

Date: 24 Oct.-24 Jan. 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561601

Parameterizable and Jerk-Limited Trajectories with Blending for Robot Motion Planning and Spherical Cartesian Waypoints

Jianjie Lin, Markus Rickert, and Alois Knoll

Abstract— **This paper presents two different approaches to generate a time local-optimal and jerk-limited trajectory with blends for a robot manipulator under consideration of kinematic constraints. The first approach generates a trajectory with blends based on the trapezoidal acceleration model by formulating the problem as a nonlinear constraint and a non-convex optimization problem. The resultant trajectory is locally optimal and approximates straight-line movement while satisfying the robot manipulator's constraints. We apply the bridged optimization strategy to reduce the computational complexity, which borrows an idea from model predictive control by dividing all waypoints into consecutive batches with an overlap of multiple waypoints. We successively optimize each batch. The second approach is a combination of a trapezoidal acceleration model with a 7-degree polynomial to form a path with blends. It can be efficiently computed given the specified blending parameters. The same approach is extended to Cartesian space. Furthermore, a quaternion interpolation with a high degree polynomial under consideration of angular kinematics is introduced. Multiple practical scenarios and trajectories are tested and evaluated against other state-of-the-art approaches.**

I. Introduction

Trajectory generation is a fundamental topic in the robotics community that deals with the calculation of a time-optimal, smooth, jerk-limited, and accurate motion for a well-defined task. Manually programming and optimizing paths for complex robot systems is no longer viable when it comes to flexible production with small lot sizes and multiple robot manipulators, a common use case in small and medium-sized enterprises [1]. In order to quickly adapt to new processes, new paths have to be generated automatically by modern path planning algorithms that are able to calculate complex motions for multiple manipulators in a narrow space. As cycle times should be as short as possible, globally optimal path planning algorithms [2] present a considerable advantage over classical algorithms that are followed by a local optimization step. In order to avoid stopping at every waypoint in a path, supporting various forms of blending is a desired property in a trajectory generation algorithm to further increase the performance of a robot system. Kinodynamic path planning algorithms with velocity information however are proven to be PSPACE hard [3] and therefore lead to a large increase in computation time. Industrial robot controllers and open-source implementations commonly support blending via linear parabolic motions and cubic spline interpolation. Trajectories based on linear parabolic blending, that only limit the acceleration, suffer from infinite jerk around the blend waypoints [4]. Although cubic spline interpolation can improve the smoothness of a path by limiting the jerk, it can result in a more significant deviation of the straight-line movement. This is especially important when calculating trajectories for position-based solutions in robot path planning. Trajectory generation without explicit error bounds in the path deviation can lead to undesired behavior. Deviating too far from the collision-free solution path can result in collisions. Based on these observations, we present two different approaches to generate a trajectory for following multiple waypoints. They support an explicit upper bound in deviation and are jerk-limited around the blended waypoints. In our previous work [4], we consider the situation of performing an accurate motion for a robot manipulator by forcing the trajectory to precisely pass through all waypoints, which are either manually specified or generated via a path planning algorithm. In this work, we extend this to a more general application by considering blending around the waypoints. In contrast to most state-of-the-art blending algorithms, the jerk limitation is followed throughout the trajectory. In the same way as Haschke et al. [5] and Kröger et al. [6], the trapezoidal acceleration profile is used to generate the trajectory between two consecutive waypoints. As stated in [4], the trapezoidal acceleration profile increases the optimization complexity while considering phase synchronization. In comparison to our previous work [4], we relaxed the objective function by introducing two additional weights to control the distribution of acceleration, deceleration, and cruising phases, which reduces the optimization complexity and shows a better performance from the perspective of straight-line movement. We continue to utilize the principle of model predictive control [4] for optimizing all waypoints by decomposing them into many consecutive waypoint batches and bridging each two adjacent batches with an overlapping waypoint. In addition to the optimization approach, we present another new approach that combines the trapezoidal acceleration model with a high-degree polynomial to perform a blending trajectory in joint and Cartesian space. Notably, quaternion interpolation is integrated and extended to a high degree polynomial, which considers the angular jerk and results in a smooth quaternion trajectory.

Jianjie Lin, Markus Rickert, Alois Knoll are with Robotics, Artificial Intelligence and Real-Time Systems, Department of Informatics, Technische Universität München, Munich, Germany `jianjie.lin@tum.de` `{rickert,knoll}@in.tum.de`

II. Related Work

Generating time-optimal and smooth trajectories has been studied extensively for decades in the robotics community.

The proposed trajectory planning techniques are roughly divided into two categories: online and offline planning.

Online real-time trajectory generation is mainly used to deal with unforeseen events and a dynamic and fast modification of the planned trajectory. Macfarlane et al. [7] proposed a jerk-bounded fifth-order polynomial with parabolic blends between two waypoints. Haschke et al. [5] presented an online trajectory planner by considering arbitrary initial kinematics and stopping at each waypoint. Kröger et al. [6] extended the online planner in a more general approach, which can handle arbitrary start and goal states. Lange et al. [8] proposed a path-accurate and jerk-limited online trajectory generation in configuration space. However, this cannot be easily extended to multiple waypoints and it is also not possible to blend the trajectory around the target.

Offline trajectory planning is suitable for a well-defined task, such as assembly or welding applications. The standard trajectory generator utilizes polynomials based on splines such as cubic splines or polynomials of higher degrees to provide a jerk-bound smooth trajectory. B-splines and their extension method [9] are also widely used to generate smooth trajectories. Although polynomial-based and B-spline-based algorithms can generate a smooth trajectory, they cannot fully explore the robot's capabilities and show a significant deviation from a straight line. Pham et al. [10] proposed a new approach based on reachability analysis for the time-optimal path parameterization (TOPP) problem. Similarly, Nagy and Vajk [11] applied a linear programming-based (LP) solver to tackle TOPP. Furthermore, Barnett et al. [12] introduced a bisection algorithm (BA) by extending the dynamic programming approaches to generate a trajectory. However, those algorithms are expensive to perform a trajectory with blends. Kunz et al. [13] proposed a path-following algorithm by adding circular blends that consider the acceleration bounds in joint space. Dantam et al. [14] presented spherical, parabolic blends by using the SLERP function, where no interpolation in a Cartesian pose is considered. These algorithms however do not take jerk limitation into account.

III. Problem Formulation

The goal is to find a time-optimal, jerk-limited, and smooth trajectory that blends an intermediate waypoint without violating kinematic constraints. Furthermore, it is required to minimize the deviation to a straight line in either joint or Cartesian space. The trapezoidal acceleration-based trajectory model [4], also called seven-segment model, has the capability to generate a smooth and jerk-limited trajectory. At segment $h \in [0, \cdots, 6]$, the kinematics are formulated as

$$\begin{aligned} a^k_{i,h+1}(t) &= a^k_{i,h} + j^k_{i,h}\Delta t_{i,h}\,, \\ v^k_{i,h+1}(t) &= v^k_{i,h} + a^k_{i,h}\Delta t_{i,h} + \frac{1}{2} j^k_{i,h}\Delta t^2_{i,h}\,, \\ p^k_{i,h+1}(t) &= p^k_{i,h} + v^k_{i,h}\Delta t_{i,h} + \frac{1}{2} a^k_{i,h}\Delta t^2_{i,h} + \frac{1}{6} j^k_{i,h}\Delta t^3_{i,h} \end{aligned} \quad (1)$$

at the waypoint i in the axis k. The parameter $\Delta t_{i,h}$ is the time difference, defined as $t_{i,h+1} - t_{i,h}$. For a phase synchronization [15] trajectory, position, velocity, acceleration, and jerk in each axis at the same segment should be synchronized. The time evolution of a position is interpreted by a third-order polynomial, which can increase the smoothness of trajectories by bounding the jerk. In this paper, we present two approaches: In the first approach, we extend our previous work, which enforces a precise pass through all waypoints, denoted as TrajOpt-Pass-Joint (TOPJ), to generate a blending trajectory by formulating it as a nonlinear constraint optimization problem, indicated as TrajOpt-Blend-Joint (TOBJ). In the second approach, we combine the trapezoidal acceleration model with a high-dimensional polynomial (7-degree) to generate a blending trajectory both in joint space TrajPoly-Blend-Joint (TPBJ) and Cartesian space TrajPoly-Blend-Cart (TPBC).

A. Blending by Optimization in Joint Space (TOBJ)

A blending trajectory is formulated as a nonlinear constraint optimization problem by applying a nonlinear optimization solver (SQP) [16]. The work presented in this paper introduces a newly designed objective function and additional inequality and equality constraints. We follow the same optimization strategy as introduced in [4].

1) Objective Function: The purpose of the objective function f is to find a trajectory that is optimal in time and moves as linearly as possible in joint space as

$$\begin{aligned} f = \sum_{i=1}^{i=n} \Bigg(& \lambda_3 \Big(\big((\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\lambda_1\big)^2 + \big(\Delta t_{i,\text{cruis}}\lambda_2\big)^2 \Big) \\ & + \lambda_4 \sum_{k=1}^{k=m} (v^k_{i,3})^2 \exp\Big(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2}\Big) \Bigg)\,, \end{aligned} \quad (2)$$

where n is the number of waypoints, m are the degrees of freedom of a robot. The time interval of the acceleration phase is indicated as $\Delta t_{i,\text{acc}}$, the cruising phase as $\Delta t_{i,\text{cruis}}$, and the deceleration phase as $\Delta t_{i,\text{dec}}$. The weight values λ_1 and λ_2 are used to control the distribution of the acceleration/deceleration and cruising phase. The choice of $\big((\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\lambda_1\big)^2 + (\Delta t_{i,\text{cruis}}\lambda_2)^2$ has advantages over the formulation $\big((\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\lambda_1 + \Delta t_{i,\text{cruis}}\lambda_2\big)^2$, which avoids the product of $(\Delta t_{i,\text{acc}} + \Delta t_{i,\text{dec}})\Delta t_{i,\text{cruis}}$, so that acceleration/deceleration phase and cruising phase cannot affect each other. On top of this, λ_3 is used to minimize the whole trajectory time. In this formulation, λ_3 and λ_1/λ_2 conflict with each other. λ_1/λ_2 is used to achieve a straight-line motion, while minimizing the time with λ_3 requires a longer acceleration/deceleration phase that can lead to an overshooting trajectory. In [4], the straight-line deviation bound is added in the objective function. In this work, we relax this constraint by emphasizing a straight-line in joint space. Furthermore, the overshooting is observed with

$$\Delta t_{i,\text{cruis}} = \big((p^k_{i,\text{target}} - p^k_{i-1,0}) - \big(\Delta p^k_{i,\text{acc}} + \Delta p^k_{i,\text{dec}}\big)\big)/v^k_{i,3}\,, \quad (3)$$

where $\Delta p^k_{i,\text{acc}} = \sum_{h=0}^{h=2} p^k_{i,h}(t_{i,h}, v^k_{i,0}, a^k_{i,0})$ and $\Delta p^k_{i,\text{dec}} = \sum_{h=4}^{h=7} p^k_{i,h}(t_{i,h}, v^k_{i,4}, a^k_{i,4}, v^k_{i,7}, a^k_{i,7})$. If the time $\Delta t_{i,\text{cruis}}$ is negative, the reached position overshoots the target. To eliminate this undesired behavior, the cruising phase should be omitted. Since $\Delta p^k_{i,\text{acc}} + \Delta p^k_{i,\text{dec}}$ has a longer distance than $p^k_{i,\text{target}} - p^k_{i-1,0}$, we can reduce the cruising velocity $v^k_{i,3}$ to shorten $\Delta p^k_{i,\text{acc}}$. Besides, the trapezoidal end velocity $v_{i,7}$ influences the position $\Delta p^k_{i,\text{dec}}$, which can be automatically tuned by the optimization solver. We add a regular term $(v^k_{i,3})^2 \exp(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2})$ in the objective function with σ, which controls the decay rate. It can be shown that at $\Delta t_{i,\text{cruis}} \approx 0$, the regular term $(v^k_{i,3})^2 \exp(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2})$ is approximated as $(v^k_{i,3})^2$, and the velocity is reduced by minimizing the objective function. In the case of $\Delta t_{i,\text{cruis}} > 0$, the Gaussian value $\exp(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2})$ is exponentially decayed to zero, which has no effect on the cruising phase.

2) Kinematic Constraints: Instead of passing through the waypoints, we define a blending bound between two consecutive line segments. Furthermore, we set a non-zero velocity and acceleration for each waypoint, except for the first and last waypoint. The constraints are described as

$$\begin{aligned}
p^k(t_{i,h_1}) &= p^k_{i+1,\text{bl,start}},\ h_1 \in [4,\dots,6]\,,\\
p^k(t_{i+1,h_2}) &= p^k_{i+1,\text{bl,end}}\,, h_2 \in [0,\dots,3]\,,\\
v^k(t_{0,0}) &= v^k(t_{n,0}) = 0,\ \Delta t_{i,h} \geq 0\,,\\
a^k(t_{0,0}) &= a^k(t_{n,0}) = a^k(t_{i,3}) = 0\,,\\
|j^k(t_{i,h})| &\leq j^k_{\max}\,, |a^k(t_{i,h})| \leq a^k_{\max}\,, \forall\, h \in [0,\dots,6]\\
|v^k(t_{i,h})| &\leq v^k_{\max}\,, |p^k(t_{i,h})| \leq p^k_{\max}\,, \forall\, h \in [0,\dots,6]\\
p^k_{i,\text{bl,lower}} &\leq p^k_{i+1,\text{bl}} \leq p^k_{i+1,\text{bl,upper}}\,,
\end{aligned} \tag{4}$$

where $p^k_{i+1,\text{bl,start}}$ is the start blending segment position at waypoint $i+1$ in axis k and $p^k_{i+1,\text{bl,end}}$ is the end blending segment position at waypoint $i+1$ in axis k. The variable $p^k_{i,\text{bl,lower}}, p^k_{i,\text{bl,upper}}$ is a lower and upper blending bound, respectively. The corresponding optimized blending waypoint at $i+1$ in axis k is indicated as $p^k_{i+1,\text{bl}}$. The variable h_1 and h_2 are predefined values that depend on the blending percentage. One relaxation of the jerk $\boldsymbol{j}$ constraint [5] is made by allowing double acceleration or deceleration phases: sign($\boldsymbol{j}$) is no longer strictly defined as $[\pm, 0, \mp, 0, \mp, 0, \pm]$ but changed to sign($\boldsymbol{j}$) $= [\pm, 0, \pm, 0, \pm, 0, \pm]$. This relaxation allows reaching the next waypoint without slowing down.

3) Blending Bound Constraints: The blending constraint $\boldsymbol{p}_{i,\text{bl,con}}$ is computed as $\boldsymbol{p}_{i+1} + \hat{\boldsymbol{y}} r\eta$, where $\hat{\boldsymbol{y}}$ is defined as $\frac{\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1}{\|\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1\|}$ with $\hat{\mathbf{y}}_1 = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$ and $\hat{\mathbf{y}}_2 = \frac{\mathbf{p}_{i+2} - \mathbf{p}_{i+1}}{\|\mathbf{p}_{i+2} - \mathbf{p}_{i+1}\|}$. Additionally, $r = l_i/(\tan \alpha_{i+1}/2)$ with $\alpha_{i+1} = \arccos(\hat{\boldsymbol{y}}_1^T \hat{\boldsymbol{y}}_2)$ and $l_i = \min\{\frac{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|}{2}, \frac{\|\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+1}\|}{2}, \frac{\delta \sin(\alpha_{i+1}/2)}{(1 - \cos(\alpha_{i+1}/2))}\}$, where δ is the predefined blending distance from q_{i+1}. η is the percentage value for controlling the blending bound. The deviation $\epsilon_{\text{bl}} = \|\boldsymbol{p}_{i,\text{bl,con}} - \boldsymbol{p}(t_{i,7})\|$ is bound. Utilizing the blending constraints, we have $p^k_{i,\text{bl,lower}} = \min\{p^k_{i,\text{bl,con}}, p^k(t_{i,7})\}$ and $p^k_{i,\text{bl,upper}} = \max\{p^k_{i,\text{bl,con}}, p^k(t_{i,7})\}$.

B. Blending with Polynomial in Joint Space (TPBJ)

The second approach combines the trapezoidal with a high-dimensional polynomial to form a blending path. The blending segment is described as a high-degree polynomial under consideration of initial $f_0 = (p_0, v_0, a_0, j_0)$ and final $f_1 = (p_1, v_1, a_1, j_1)$ kinematic constraints. These require a total of eight coefficients, therefore we utilize a 7-degree polynomial function in one dimension: $f(t) = b_7 t^7 + b_6 t^6 + \cdots + b_2 t^2 + b_1 t + b_0$. The coefficients b_0 - b_3 can be computed using f_0 with $b_0 = p_0$, $b_1 = v_0$, $b_2 = \frac{a_0}{2}$ and $b_3 = \frac{j_0}{6}$. The coefficients b_4 to b_7 depending on f_1 and polynomial time t can be described as

$$\underbrace{\begin{bmatrix} t_1^7 & t_1^6 & t_1^5 & t_1^4 \\ 7t_1^6 & 6t_1^5 & 5t_1^4 & 4t_1^3 \\ 42t_1^5 & 30t_1^4 & 20t_1^3 & 12t_1^2 \\ 210t_1^4 & 120t_1^3 & 60t_1^2 & 24t_1 \end{bmatrix}}_{\boldsymbol{A}_1} \underbrace{\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \end{bmatrix}}_{\boldsymbol{x}_1} = \underbrace{\begin{bmatrix} p_1 \\ v_1 \\ a_1 \\ j_1 \end{bmatrix}}_{\boldsymbol{y}_1} - \underbrace{\begin{bmatrix} 1 & t_1^1 & t_1^2 & t_1^3 \\ 0 & 1 & 2t_1 & 3t_1^2 \\ 0 & 0 & 2 & 6t_1 \\ 0 & 0 & 0 & 6 \end{bmatrix}}_{\boldsymbol{A}_2} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}}_{\boldsymbol{y}_2}\,.$$

Due to this, $\boldsymbol{x}_1(t_1, f_0, f_1) = \boldsymbol{A}_1^{-1}(\boldsymbol{y}_1 - \boldsymbol{A}_2\boldsymbol{y}_2)$ with time matrices A_1, A_2. The polynomial is simplified as

$$f(t, f_0, f_1) = [t^7, t^6, t^5, t^4]\boldsymbol{x}_1(t_1, f_0, f_1) + h_2(t, f_0)\,, \tag{5}$$

where $h_2(t, f_0)$ is described as $\frac{1}{6} j_0 t^3 + \frac{1}{2} a_0 t^2 + v_0 t + p_0$. Firstly, we assume that the initial f_0 and final f_1 kinematic constraints are available. Therefore, $f(t)$ depends only on the time t. To find a polynomial blending trajectory that satisfies all kinematic constraints, we need to verify the extreme point of the polynomial by computing the root of its derivative. For example, the extreme point of jerk can be found at the position where the first derivative of the jerk (snap) is equal to zero. In addition, a n-degree polynomial has at most n real roots. The constraints can be mathematically formulated as

$$|f^{(n-1)}(\lambda(f^{(n)}))\chi_\lambda(\lambda(f^{(n)}))| \leq k_{\max}\,, \tag{6}$$

where $f^{(n)}$ is n-th derivative of f with $n \in [1, 4]$ and the corresponding constraints $k_{\max} \in [p_{\max}, v_{\max}, a_{\max}, j_{\max}]$. The root-finding function $\lambda(f^{(n)})$ for a given polynomial is used to find the extreme value position for $f^{(n-1)}$. $\chi_A(x)$ is the indicator function of A and will be set to one if $x \in [0, t]$, otherwise to zero. Note that if the $\chi_A(x)$ function is not derivable, the gradient-based optimization solver will diverge. To find a time-optimal polynomial trajectory that satisfies initial/final conditions and lies within the kinematic constraints, we iteratively check the constraints (6) by adding a small delta to $t = t + \Delta t$, where in our case Δt is set to 0.001. Furthermore, to obtain the initial f_0 and final f_1 kinematics, we compute a Point to Point (P2P) trapezoidal acceleration profile movement between each two waypoints, which can be in joint space or Cartesian space, with zero initial and end conditions, and the computed traveling time is indicated as $t_{i->i+1}$. After that, we predefine a blending percentage η to set a start blending time $t_{\text{b}_\text{start}} = (1-\eta)t_{i->i+1}$ and end blending time $t_{\text{b}_\text{end}} = \eta t_{i+1->i+2}$ for each two consecutive trajectories. By querying the trapezoidal model at $t_{\text{b}_\text{start}}$ and t_{b_end}, we obtain the kinematics f_0 and f_1.

C. Blending with Polynomial in Cartesian Space (TPBC)

Utilizing the same principle, we extend the algorithm to the Cartesian space, which is widely used in industrial applications. The blending trajectory in Cartesian space requires separate blending for position and orientation. Interpolation of the Cartesian space is executed in the same way as described in Section III-B. For the orientation part, which has to consider a spherical interpolation, we apply the formulation:

$$\boldsymbol{h}(t) = \boldsymbol{h}_i \Delta \boldsymbol{h}(t) = \boldsymbol{h}_i \begin{pmatrix} \boldsymbol{u}(t)\sin(\theta(t)/2) \\ \cos(\theta(t)/2)\,, \end{pmatrix} \tag{7}$$

where $\boldsymbol{h}_i$ is the initial quaternion, and $\Delta\boldsymbol{h}(t)$ transforms the quaternion from $\boldsymbol{h}_i$ to $\boldsymbol{h}(t)$. The Eigen axis between two quaternions is defined as $\boldsymbol{u}(t) = \boldsymbol{\theta}(t)/\,\|\boldsymbol{\theta}(t)\| \in \mathcal{R}^3$ with a rotation angle $\theta(t) = \|\boldsymbol{\theta}(t)\|$. Therefore, the quaternion interpolation depends only on $\boldsymbol{\theta}(t) \in \mathcal{R}^3$. To consider the angular velocity $\boldsymbol{\omega}$, angular acceleration $\dot{\boldsymbol{\omega}}$, and angular jerk $\ddot{\boldsymbol{\omega}}$ at the start and end state for the quaternion blending, the time evolution function $\boldsymbol{\theta}(t)$ is described as: $\boldsymbol{\theta}(t) = \boldsymbol{a}_1(x-1)^7 + \boldsymbol{a}_2 x(x-1)^6 + \cdots + \boldsymbol{a}_8 x^7$ with $x = \frac{t}{t_f - t_0} \in [0, 1]$. Its roots and its derivative are computed at point $x = 0$ and $x = 1$. Based on $\dot{\boldsymbol{h}} = \frac{1}{2}\boldsymbol{h}\boldsymbol{\omega}$, we can derive the relationship between $\boldsymbol{\omega} \in \mathcal{R}^3$ and $\dot{\boldsymbol{\theta}} \in \mathcal{R}^3$ as $\boldsymbol{\omega} = \boldsymbol{u}\dot{\theta} + \sin(\theta)\boldsymbol{w} \times \boldsymbol{u} - (1 - \cos(\theta))\boldsymbol{w}$, where $\boldsymbol{w} = \frac{(\boldsymbol{u}\times\dot{\boldsymbol{\theta}})}{\theta} \in \mathcal{R}^3$ and $\dot{\theta} = \boldsymbol{u}^T\dot{\boldsymbol{\theta}}$ is a scalar value. We further simplify $\boldsymbol{\omega}$ with the skew-symmetric matrix $(\cdot)^\times$ as $\boldsymbol{\omega} = (\boldsymbol{u}\boldsymbol{u}^T - \frac{\sin(\theta)}{\theta}\boldsymbol{u}^\times\boldsymbol{u}^\times - \frac{(1-\cos(\theta))}{\theta}\boldsymbol{u}^\times)\dot{\boldsymbol{\theta}} = \boldsymbol{A}_{\omega,1}\dot{\boldsymbol{\theta}}$. In the same way, we can compute the angular acceleration $\dot{\boldsymbol{\omega}}$ and jerk $\ddot{\boldsymbol{\omega}}$. We combine Cartesian position and quaternion interpolation to form a trajectory with blends.

IV. Experimental Evaluation

We compare the performance of the presented approaches in this paper with the work by Kunz et al. [13] (TO-BAV), Pham et al. [10] (TOPP-RA) and our previous work [4]. The work in [13] generates a trajectory with a designed circular blend under consideration of velocity and acceleration bounds and the work [10] used the reachability-analysis (RA) to solve the time optimal path parameterization (TOPP) problem. Our previous work [4] generates a trajectory by forcing a precise pass through all desired waypoints without stopping. For the evaluation of the motion planning scene, the collision-free paths in the examples are computed using the Robotics Library [17], which is also used for kinematic calculations and simulation. In the evaluation, we set the weights in [4] and TOBJ as $\lambda_1 = 1.2$, $\lambda_2 = 1$, $\lambda_3 = 5000$, and $\lambda_4 = 10$. All evaluations were performed on a laptop with a 2.6 GHz Intel Core i7-6700HQ and 16 GB of RAM.

A. Evaluation of Deviation from Straight-Line in Joint Space

For the first evaluation, we consider a subset of the well-known ISO 9283 [18] cube industrial benchmark as shown in Fig. 1. Here, the robot's end effector has to follow a number of waypoints that are part of a two-dimensional rectangle inside a three-dimensional cube while applying blending. The corresponding results are shown in Fig. 1. The previous work TOPJ [4] in Fig. 1a completed the ISO cube task within 6.26 s by passing through all waypoints and the generated trajectory has to deviate from the straight-line movement to avoid stopping at each waypoint. For improving the quality of the trajectory, TOBJ presented in this work extends the TOPJ by blending around the target position, which results in a better straight-line movement and completed the ISO cube task with a shorter time of 6.1 s due to a shortened path length. The polynomial based algorithm (TPBJ) can further reduce the completion time to 5.94 s by setting a bigger blending circle radius.

B. Comparison between TO-BAV, TOPP-RA and TPBJ

In this section, we compare the algorithm of [10], [13] and our polynomial-based one, as using an extreme jerk value in the optimization-based one may not converge since the gradient value in jerk direction is not at the same order of magnitude with other gradient values in the gradient vector. The results are shown in Figs. 2a to 2o. The first column shows the results from TO-BAV [13], where the velocity arrives at the peak value very quickly with execution time 2.2577 s. The second column illustrates the results of TOPP-RA [10], which is forced precisely to pass through desired waypoints with the traveling time 2.88 s. The other columns show the results of the polynomial-based algorithm with increasing jerk limits, starting from 5, to 100, and finally 10 000 times the maximum velocity value. The respective trajectory travel time reduces from a value of 3.577 s to 2.28 s and our algorithm gradually approaches the profile of [13], while still limiting the jerk. We plot the first and third DOF path without loss of generality, shown in the first row. TOPP-RA produces a trajectory shown in Fig. 2b which has a noticeably bigger straight-line deviation in joint space than other two algorithms. The bigger straight-line deviation can lead to a collision, which is not desirable for the industrial application. From the perspective of velocity and acceleration performance, the velocity profile from TOPP-RA shown in Figs. 2g and 2l is less smooth than TO-BAV and TPBJ and exhibits several vibration points.

C. Evaluation of the Algorithm in a Real Robot Workcell

We evaluate our algorithms on a Universal Robots UR5 robot manipulator by looking at the actual velocity and current measured by the UR5 controller with an update rate of 125 Hz over the native Real-Time Data Exchange protocol. We compare our approaches against the algorithm developed

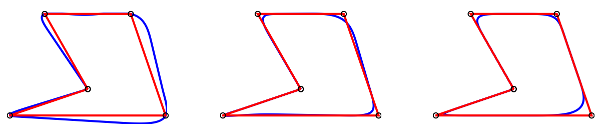


Fig. 1: Comparison of joint space trajectories for the ISO cube scenario. The plots show the first and second DOF of different algorithms. The straight line reference in joint space is shown in *red*, the calculated joint trajectory in *blue* with (a) TOPJ [4], (b) TOBJ, (c) TPBJ.

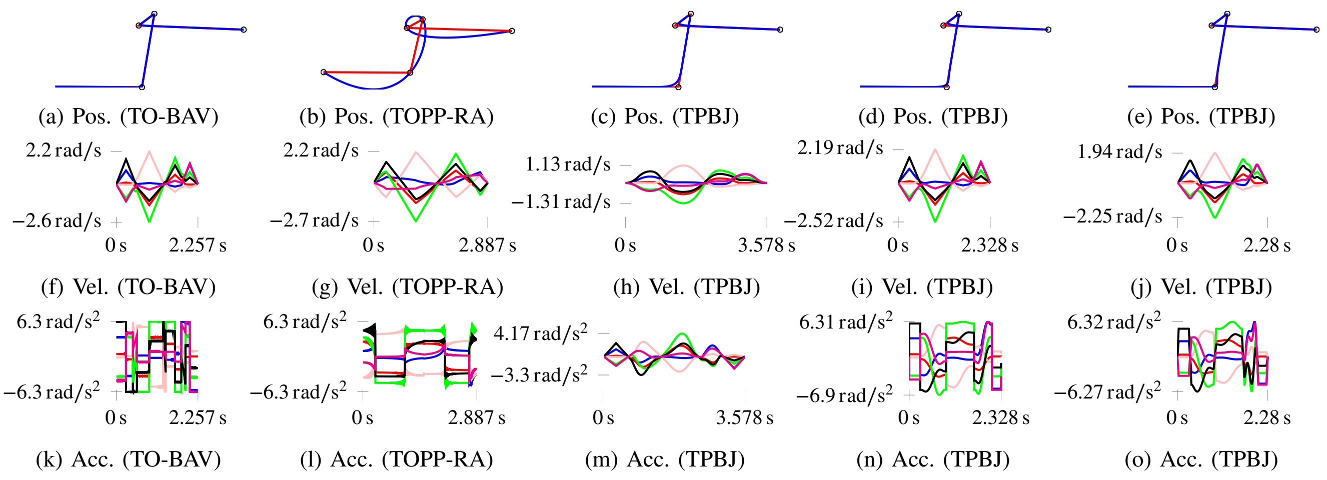


Fig. 2: Comparison of results from TO-BAV [13], TOPP-RA [10] and TPBJ with increasing jerk constraints. (a)–(e) show the plot of first and third DOF with generated trajectory (blue) and target straight-line path (red). (f)–(j) is the velocity profile. (k)–(o) is the acceleration profile. In our approach, TPBJ gradually increases jerk constraints from $j_{\text{max}} = \{5, 100, 10000\}v_{\text{max}}$.

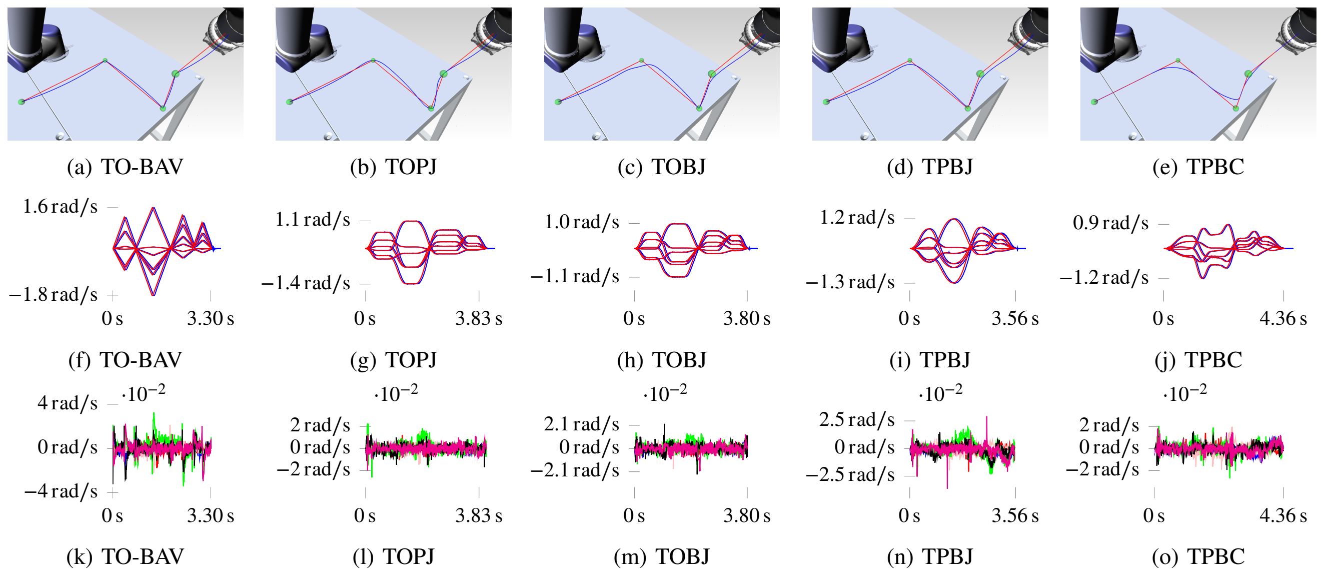
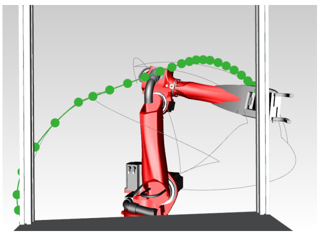


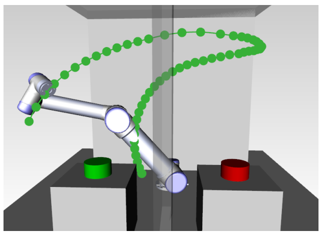
Fig. 3: A comparison of different trajectory profiles for the UR5 example. The individual plots show (a)–(e) position, (f)–(j) velocity with the controller's target (red) and actual (blue) value, and (k)–(o) corresponding velocity differences.

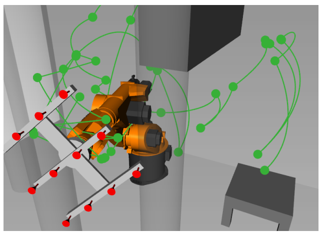
by TO-BAV [13]. The evaluation results are illustrated in Fig. 3. The maximum velocity and acceleration values are identical for all trajectory generators. The only difference is that [13] does not consider any jerk limitation. The remaining algorithms limit the maximum jerk to a value of five times faster than the maximum velocity. We evaluate the computational complexity of each algorithm by running the experiment 40 times. The computed trajectories are visualized in a 3D environment [17] as shown in Figs. 3a to 3e. The blending directly implemented in Cartesian space follows a straight line. The other four algorithms have a similar trajectory performance. It needs be pointed out that the blending mode in [4] is different from the other algorithms, as it can pass precisely through all desired waypoints without stopping. From the perspective of velocity performance, the result from [13] indicates a large gap between desired and real velocity which occurs at each turning point, shown in Fig. 3k, as the robot controller is not able immediately to execute a trajectory that contains an infinite jerk. If the maximum specified acceleration is further increased, the additional burden on the motors may lead to hardware issues and a reduced lifetime. In contrast to this, the algorithm presented in this work considers the jerk limitation. The robot controller can follow the desired waypoints continuously and demonstrates reduced motor current values. Figures 3g and 3h exhibit a clear period of cruising phase in comparison to the other algorithms, as the objective function in the optimization step emphasizes a longer constant velocity

TABLE I: Benchmark results of path planning scenarios. The best results are highlighted in bold and smaller values are better. The maximum blending deviation δ is set to 0.1. The jerk limitation used by TPBJ is set to 100x and 500x of the maximal velocity constraints, denoted as TPBJ1 and TPBJ5, respectively. TOBJ is set to 100x. N_{point} describes the number of waypoint, t_{comp} is the computation time, and t_{tra} is the traveling time. L_{alg} is the traveling length, L_{stra} is the base straight-line length, and we present the percentage.

	Scenario 1 ($N_{\text{point}} = 42$)					Scenario 2 ($N_{\text{point}} = 55$)					Scenario 3 ($N_{\text{point}} = 42$)					Scenario 4 ($N_{\text{point}} = 181$)				
Alg.	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5
t_{comp} [s]	0.23	0.28	54.15	0.15	**0.088**	0.28	**0.186**	65.7	0.65	0.319	0.32	0.45	149.52	0.41	**0.25**	1.68	0.54	231.68	0.70	**0.41**
t_{tra} [s]	4.08	**3.759**	7.09	7.02	5.95	3.67	**3.40**	6.35	6.58	5.88	25.20	**23.95**	26.17	26.32	25.48	11.75	**10.55**	26.48	26.44	18.91
$L_{\text{alg}}/L_{\text{stra}}$	0.999	1.0027	**0.997**	1.0001	0.999	0.999	1.003	0.998	1.0001	**0.998**	0.998	1.174	**0.997**	1.001	**0.997**	**0.996**	1.001	1.001	1.001	0.997


(a) Scenario 1


(b) Scenario 2


(c) Scenario 3

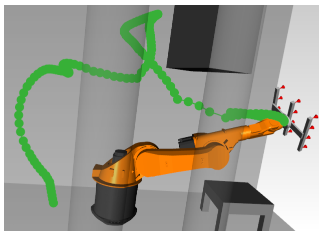

(d) Scenario 4

Fig. 4: Benchmarks on different motion planning scenarios. (a) a Comau Racer 7-1.4 moves to a workstation with a parallel gripper. (b) a UR5 moving between two walls. (c)–(d) a Kuka KR60-3 next to a wall and 3 columns with a vacuum gripper.

phase over the acceleration and deceleration phases. This behavior is desirable in certain industrial robot task such as welding and gluing, as the cruising segments show a smoother overall behavior compared to the other segments.

D. Evaluation of Computation Complexity in Motion Planning Scenarios

We evaluate the algorithms for generating trajectories in different path planning scenarios with an increasing number of waypoints (from 42 to 181) and different point distribution characteristics, as illustrated in Fig. 4. We compare our approaches against TO-BAV, which is currently a standard trajectory generator in the MoveIt! framework, and TOPP-RA. We summarize the results in Table I. Regarding computation time, TPBJ is faster in most scenarios apart from scenario 2. In comparison to TOBJ and TPBJ under the same jerk limitation (100x of maximal velocity constraints), they show a similar performance. However, if the jerk limitation is bigger than 100x, TOBJ has a convergence problem, because jerk and time have huge differences in numerical magnitude. In comparison to TO-BAV and TPBJ with jerk limitation set to 100x and 500x of the maximum velocity constraint, we can conclude that with a higher jerk limitation, TPBJ can significantly reduce the traveling time. This conclusion can be drawn from subsection IV-B as well. The trajectory optimizer TOPP-RA without jerk limitation generates a trajectory with minimal traveling time. From the perspective of straight-line deviation, TOPP-RA generated a trajectory with significant overshooting in scenario 3 with 117.4% path length with respect to a straight-line movement, since TOPP-RA utilizes cubic spline interpolation for path parameterization. The drawbacks of using cubic spline interpolation are shown in [4].

V. Conclusion

In this work, we have extended our previous work by including the capability to blend around the target position. We have presented two different approaches to finding a time-optimal and jerk-limited trajectory. In the first approach, the algorithm follows the same principle as in our previous work by using a bridged optimization procedure, which reduces the computational complexity to a linear complexity with respect to the number of waypoints and degrees of freedom. In contrast to our previous work, we redesigned the objective function and blending constraints to achieve a better straight-line movement in joint space and allow the trajectory to blend around the target position. However, TOBJ has a convergence problem when the jerk constraint exceeds 100x of the maximum velocity constraint due to the huge differences in numerical magnitude between jerk und time. Further improvement will be left to future work by using more advanced optimization strategies. The second approach combines a trapezoidal trajectory with a seven-degree polynomial. In this approach, we compute a point-to-point motion for every two waypoints by using a standard trapezoidal acceleration model. By specifying a blend percentage, the seven-degree polynomial can be used to find a curve segment around the target position. To find a time-optimal trajectory that fully considers all kinematic constraints, we iteratively increase the trajectory time until these constraints are no longer violated for all degrees of freedom. The second approach can be directly extended to the Cartesian space by using the introduced quaternion interpolation algorithm. These two approaches do not suffer from convergence problems and show good performance in our experiments when compared against state-of-the-art approaches without jerk limitations.

REFERENCES

[1] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. G. Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kesslar, and M. Danzer, "SMErobotics: Smart robots for flexible manufacturing," *IEEE Robotics and Automation Magazine*, vol. 26, no. 1, pp. 78–90, Mar. 2019.

[2] E. F. Sertac Karaman, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[3] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, Nov. 1993.

[4] J. Lin, N. Somani, B. Hu, M. Rickert, and A. Knoll, "An efficient and time-optimal trajectory generation approach for waypoints under kinematic constraints and error bounds," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 2018, pp. 5869–5876.

[5] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, Sept. 2008, pp. 3248–3253.

[6] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, Feb. 2009.

[7] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[8] F. Lange and A. Albu-Schäffer, "Path-accurate online trajectory generation for jerk-limited industrial robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 82–89, Jan. 2016.

[9] R. Saravanan, S. Ramabalan, and C. Balamurugan, "Evolutionary optimal trajectory planning for industrial robot with payload constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 38, pp. 1213–1226, Sept. 2008.

[10] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, June 2018.

[11] Á. Nagy and I. Vajk, "Sequential time-optimal path-tracking algorithm for robots," *IEEE Transactions on Robotics*, vol. 35, no. 5, Oct. 2019.

[12] E. Barnett and C. Gosselin, "A bisection algorithm for time-optimal trajectory planning along fully specified paths," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 1–15, Feb. 2021.

[13] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Proceedings of Robotics: Science and Systems*, July 2012.

[14] N. Dantam and M. Stilman, "Spherical parabolic blends for robot workspace trajectories," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, Sept. 2014, pp. 3624–3629.

[15] T. Kröger, "Opening the door to new sensor-based robot applications—the Reflexxes motion libraries," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[16] J. Nocedal and S. Wright, *Numerical optimization.* Springer Science & Business Media, 2006.

[17] M. Rickert and A. Gaschler, "Robotics Library: An object-oriented approach to robot applications," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vancouver, BC, Canada, Sept. 2017, pp. 733–740.

[18] "Manipulating industrial robots, Performance criteria and related test methods," ISO, International Standard 9283, Apr. 1998.

Sign in/Register

Parameterizable and Jerk-Limited Trajectories with Blending for Robot Motion Planning and Spherical Cartesian Waypoints

Conference Proceedings:
2021 IEEE International Conference on Robotics and Automation (ICRA)

Author: Jianjie Lin; Markus Rickert; Alois Knoll

Publisher: IEEE

Date: 30 May-5 June 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Deep Hierarchical Rotation Invariance Learning with Exact Geometry Feature Representation for Point Cloud Classification

Jianjie Lin, Markus Rickert, and Alois Knoll

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561307

Abstract— **Rotation invariance is a crucial property for 3D object classification, which is still a challenging task. State-of-the-art deep learning-based works require a massive amount of data augmentation to tackle this problem. This is however inefficient and classification accuracy suffers a sharp drop in experiments with arbitrary rotations. We introduce a new descriptor that can globally and locally capture the surface geometry properties and is based on a combination of spherical harmonics energy and point feature representation. The proposed descriptor is proven to fulfill the rotation-invariant property. A limited bandwidth spherical harmonics energy descriptor globally describes a 3D shape and its rotation-invariant property is proven by utilizing the properties of a Wigner D-matrix, while the point feature representation captures the local features with a KNN to build the connection to its neighborhood. We propose a new network structure by extending PointNet++ with several adaptations that can hierarchically and efficiently exploit local rotation-invariant features. Extensive experimental results show that our proposed method dramatically outperforms most state-of-the-art approaches on standard rotation-augmented 3D object classification benchmarks as well as in robustness experiments on point perturbation, point density, and partial point clouds.**

I. Introduction

Convolutional neural networks (CNN) [1] have shown tremendous success in image processing due to their translation-invariant capability of detecting local patterns regardless of their position in the image and their ability to process regular data, such as image grids or 3D voxels. However, the more challenging rotation-invariant property is still missing in the designed structure [2]. Data augmentation is a common approach to address this issue. The infinite property of the rotation group howver makes this approach less efficient and comes with a high computational cost. A big neural network with rotation-augmented data is required to generalize the data set. In 3D, geometric irregular data formats such as point clouds increase the difficulty of handling the rotation transformation, while irregular data formats suffer from a permutation problem $N!$. To address this issue and to inherit the benefits of convolutional networks, which can process regular data formats, previous work such as [3], [4] voxelized geometric shapes. [5], [3], [6] proposed a rotation-equivariant network with newly designed spherical convolutional operators. However, the voxelization of 3D geometry induces a trade-off between resolution and computational cost. The pioneering work PointNet used a spatial transformation network to learn an affine transformation, which still did not fulfill the requirement. Inspired by CNNs, which use different receptive fields to aggregate the local features, DGCNN used a dynamic k-nearest neighbors algorithm (KNN) to exploit local information. However, its classification results still suffer a sharp drop in rotation experiments.

For alleviating the issue, we introduce two different rotation-invariant features (RIF). The first one is spherical harmonics [7], which transform the Cartesian pose to the spectral domain by using a non-commutative Fourier analysis methods and are related to the power spectrum in the perspective of signal processing. The second feature can locally describe the geometry relationship by creating a Darboux frame at each object point with a KNN-graph. This geometry point feature is also utilized in the point feature histogram [8] and fast point feature histogram [9]. The rotation-invariant feature aims at separating the rotated point cloud and the network so that the input space is invariant to arbitrary rotation perturbation. Furthermore, we design a new network structure that can hierarchically extract the local features by applying the farthest point sampling strategy. The proposed network structure is composed of RIMapping, PF Abstraction, and Classification blocks. In the RIMapping block, rotation-invariant features are fed to a feature transformation network, which maps the lower level feature to a high level embedding space. Two consequent abstraction layers work on these high-level embedding features. For further exploiting the local geometry information, a fully connected point feature graph is built on each cluster and the resultant features are fed to a point feature transformation. Afterward, a global abstraction layer can aggregate all previous embedding features together to obtain a global feature. The Classification block is a standard fully connected network to classify the objects. We evaluated our proposed network on ModelNet40 with different experimental settings and achieve or exceed most state-of-the-art approaches.

Our primary contributions are two-fold: a) we introduce a novel geometry rotation-invariant feature descriptor, which can globally and locally represent a 3D shape. b) a new rotation-invariant classification network structure is designed, which can efficiently exploit local geometric features.

II. Related Work

With recent good results from deep learning in image-based recognition, 3D visual recognition has also received more attention and rapid development. It benefits from deep learning in extracting and learning geometric features more

Jianjie Lin, Markus Rickert, Alois Knoll are with Robotics, Artificial Intelligence and Real-Time Systems, Department of Informatics, Technische Universität München, Munich, Germany jianjie.lin@tum.de {rickert,knoll}@in.tum.de

efficiently, but the recognition of 3D geometry differs from image-based recognition in many factors. One main aspect are the representation formats, where 3D geometry uses various methods such as point cloud-based representation, implicit surfaces based representation, or volumetric-based representations. These different formats lead to different learning methods. In contrast, the imaged-based representation is interpreted in regular data, where the conventional CNN is designed to handle such regular data. The permutation $N!$ is a common problem in the irregular data format. Based on these observations, previous work seeks to utilize benefits from conventional CNNs by voxelizing the 3D geometric shape [4], [10], [11], [12] or by using multi-view images [11], [13], [14]. However, the trade-off between the resolution and computational cost makes generalization impossible. Most 3D convolutional neural networks sacrifice high resolution to obtain fast calculations to build upon a shallow network. To alleviate the negative impact of accuracy due to resolution, an Octet [15] is proposed by hierarchically partitioning the space using a set of unbalanced octrees to exploit sparse input data.

In contrast to a volumetric representation, PointNet [16] is the first work that directly feeds the point cloud into a set of shared MLP networks and uses the max pool operator to extract global features. It shows a significant improvement in the perspective of 3D shape reasoning and computational cost. PointNet, however, does not extract local information. Follow-up work such as PointNet++ [17] progressively aggregated local features using the farthest point sampling strategy. Moreover, DGCNN [18] introduced a dynamic KNN to build a local graph and aggregated the edge features to obtain a better feature representation. A point-based neural network satisfies many properties, e.g., permutation invariance with a shared MLP and max pool operator and translation equivariance with a relu operator [5]. This network is shown to solve many classical problems such as classification, part segmentation, and instance segmentation. The rotation-invariant property is however still missing in the designed structures. PointNet applies a spatial transformer network [19] to predict an affine transformation matrix. Other work attempts to augment the data set by generating a lot of SO(3) combinations. However, SO(3) is infinite, and data augmentation wastes computational resources and cannot guarantee effectiveness. To alleviate this issue, previous work proposed a rotation-equivalence network structure. [20], [5], [3] designed a spherical-based convolutional operator utilizing the properties of spherical harmonics. [21] proposed tensor field networks, which map point clouds to point clouds under the constraint of SE(3) equivariance by utilizing a spherical harmonics filter. Spherical representations for 3D data are not novel and have been used for retrieval tasks before the deep learning era [7], [22].

Spherical-based CNNs were initially designed for voxelized shapes and suffered a loss of geometric information, as there is no bijection between $\mathbb{R}^3$ and 2-dimensional sphere S^2 [23] as mentioned above. Instead of proposing a new convolutional operator, [23] introduced rigorously rotation-invariant (RRI) features by transforming the point from Cartesian space into an embedding space and showed a good improvement in experiments. However, the RRI features focus only on the local feature using the same dynamic KNN as DGCNN.

III. Geometric RIF Descriptor

Given a set of transformations $\boldsymbol{T}_{g_i} : \mathcal{V} \to \mathcal{V}$ for $g_i \in$ SO(3), a rotation-invariant function $\phi(\cdot)$ has the property

$$\phi(\boldsymbol{T}_{g_1}\boldsymbol{q}) = \phi(\boldsymbol{T}_{g_2}\boldsymbol{q}), \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^3$ is a point in the Cartesian coordinate system. Pioneering works in processing point clouds are PointNet and DGCNN, where EdgeConv from DGCNN and mini-PointNet from PointNet++ utilize the edge feature represented as an implicit geometry feature by considering geometric constraints between points. The edge features $\boldsymbol{x}_i - \boldsymbol{x}_j$ and pose point $\boldsymbol{x}_i$ do not satisfy the property described in (1). Furthermore, edge features under a dynamic KNN can only represent the local geometric context for point clouds in the embedding space. For alleviating this issue, two rotation-invariant descriptors will be introduced, that globally (spherical harmonics descriptor) and locally (point feature descriptor) represent the geometry shape.

A. RI Spherical Harmonics (SH) Descriptor

Definition: Spherical harmonics define an orthonormal basis over the sphere, with the parameterization

$$(x, y, z) = (\sin(\theta)\cos(\varphi), \sin(\theta)\sin(\varphi), \cos(\theta)), \tag{2}$$

where (x, y, z) is a location defined on a unit sphere with colatitude θ and longitude φ and the orthonormal basis function given by Rodrigues' formula can be described as

$$Y_l^m(\theta, \varphi) = K_l^m P_l^m(\cos\theta)e^{im\varphi}, \tag{3}$$

with the normalized constant variable K_l^m and the associated Legendre polynomials P_l^m. The parameters l and m are the spherical harmonic degree and order, respectively. Furthermore, the order should satisfy the constraint $-l \leq m \leq l$. The real spherical harmonics are sometimes known as tesseral spherical harmonics. These functions have the same orthonormality properties as the complex ones above. The harmonics with $m > 0$ are said to be of cosine type and those with $m < 0$ of sine type. The reason for this can be seen by writing the functions in terms of the Legendre polynomials P_l^m with Condon-Shortley phase convention as

$$Y_{lm} = \begin{cases} (-1)^m \sqrt{2} K_{|m|}^l P_l^{|m|}(\cos\theta)\sin(|m|\varphi) & \text{if } m < 0 \\ \sqrt{\frac{2l+1}{4\pi}} P_l^m(\cos\theta) & \text{if } m = 0 \\ (-1)^m \sqrt{2} K_m^l P_l^m(\cos\theta)\cos(m\varphi) & \text{if } m > 0 \end{cases} \tag{4}$$

Moreover, for any rotation matrix $\boldsymbol{R} \in$ SO(3), the rotated SH $Y_l^m(\boldsymbol{R}\cdot)$ can be expressed as a linear combination of other SHs of the same degree l

$$Y_l^m(\boldsymbol{R}\cdot) = \sum_{m'=-l}^{l} \left[D_{\boldsymbol{R}}^{(l)}[m, m']\right]^* Y_l^{m'}, \tag{5}$$

where $\boldsymbol{D}^{(l)}_{\boldsymbol{R}}[m, m'] \in \mathbb{C}^{(2l+1)\times(2l+1)}$ is called the Wigner D-matrix. Note that the Wigner matrices $\boldsymbol{D}^{l}$ are all orthonormal and irreducible representations of SO(3) [24], which considers them as *smallest* representations possible. In accordance with the unitary of $\boldsymbol{D}^{(l)}_{\boldsymbol{R}}$, the energy within a subspace is preserved. Therefore, for any given vector $\boldsymbol{c} \in \mathbb{C}^{2l+1}$, the Wigner D-matrix shows a norm preservation property [25], [26] as $\left\|\boldsymbol{D}^{(l)}_{\boldsymbol{R}}\boldsymbol{c}\right\| = \|\boldsymbol{c}\|$. The theory of spherical harmonics says that any spherical function $\mathrm{f}(\theta, \varphi)$ is decomposed as the sum of its harmonics:

$$f(\theta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{m=l} a_{lm} Y_l^m(\theta, \varphi), \tag{6}$$

with the coefficient a_l^m. Equation (6) can be seen as a kind of Fourier series on the sphere.

*1) **Information loss for a limited bandwidth**:* Since we cannot solve $l \to \infty$, we limit the band l to a constant degree $n_{\mathrm{sh,deg}}$. The information loss is defined as

$$\mathrm{Loss} = \left\| \sum_{l=0}^{n_{\mathrm{sh,deg}}} f_l - \sum_{l=0}^{\infty} f_l \right\|_2 . \tag{7}$$

Furthermore, the numerical solution of coefficients a_l^m can be approximated by using the Monte Carlo integration approach.

$$a_l^m = \frac{4\pi}{n_{\mathrm{sh,deg}}} \sum_{j=0}^{n_{\mathrm{sh,deg}}} f_j\left(\theta_m, \phi_m\right) Y_{l,j}^m\left(\theta_m, \phi_m\right) \tag{8}$$

*2) **SH energy descriptor**:* Polygonal-based surface representations are typically described as Cartesian coordinates (x, y, z). For spherical harmonics, the surfaces are represented by $\mathrm{f}(\theta, \phi)$, therefore the mesh must be transformed into spherical polar coordinates (r, θ, ϕ) about the origin. In this case we define $\mathrm{f}(\theta, \phi) = r$ [27] with the energy spectrum descriptor of spherical harmonics [7]

$$\mathcal{X}_{sh}(f) = \left\{ \|f_0(\theta, \varphi)\|, \|f_1(\theta, \varphi)\|, \dots \right\} \tag{9}$$

with the frequency components

$$f_l = \left[a_{l,-l} Y_l^{-l}, a_{l,-l+1} Y_l^{-l+1} \cdot \cdots, a_{l,l} Y_l^{l}\right] . \tag{10}$$

Utilizing the norm preservation property of Wigner D-matrices [7], [20], [26], we can prove that $\|f_l\|$ is a rotation-invariant descriptor.

B. RI Point Feature (PF) Descriptors

We employ point feature representations to encode the neighborhood's geometrical properties, which provides an overall point density and pose invariant multi-value feature. The surface normal [28] is estimated by using PCA on the k-neighborhood. Furthermore, for each pair $\boldsymbol{p}_s$ and $\boldsymbol{q}_t$ with $\boldsymbol{q}_t \in \mathcal{N}(\boldsymbol{p}_s)$, Darboux frame at $\langle \boldsymbol{p}_s, \boldsymbol{n}_s \rangle$ is defined as

$$\mathbf{u} = \boldsymbol{n}_s, \quad \mathbf{v} = \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \times \mathbf{u}, \quad \mathbf{w} = \mathbf{u} \times \mathbf{v}. \tag{11}$$

The point features descriptor [29] is described as a quadruplet $\langle \alpha, \phi, \theta, d_{\mathrm{st}} \rangle$ with

$$\begin{aligned} d_{\mathrm{st}} &= \|\boldsymbol{p}_t - \boldsymbol{p}_s\|_2, & \alpha &= \mathbf{v} \cdot \boldsymbol{n}_t, \\ \phi &= \mathbf{u} \cdot \frac{(p_t - p_s)}{d_{st}}, & \theta &= \operatorname{atan2}\left(\mathbf{w} \cdot \boldsymbol{n}_t, \mathbf{u} \cdot \boldsymbol{n}_t\right) . \end{aligned} \tag{12}$$

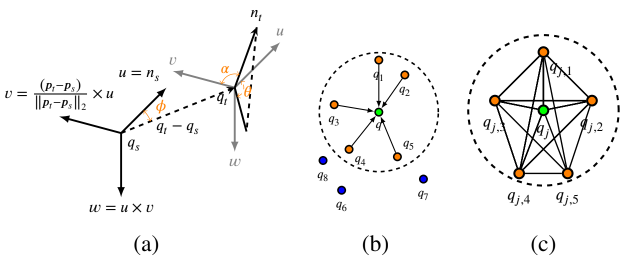


Fig. 1: (a) illustrates the Darboux frame between a point pair. (b) calculates the simplified point features of a source point $\boldsymbol{q}$ with a KNN graph for each point pair. A fully connect point feature (PF) graph (c) is built on a given source point $\boldsymbol{q}_i$.

Furthermore, we augment the distance $r_s = \|\boldsymbol{p}_s\|_2$ to the point feature, therefore, we get a quintuple feature descriptor.

*1) **Proof of rotation-invariant property**:* Given a point set $S = \left\{\boldsymbol{p}_i \mid \boldsymbol{p}_i \in \mathbb{R}^n\right\}_{i=0}^{N-1}$. It is obvious that the L^2 norm is a rotation-invariant operator $\mathbb{R}^n$ to $\mathbb{R}$ due to norm preservation: $\|\boldsymbol{R}\boldsymbol{x}\|_2^2 = \|\boldsymbol{x}\|_2^2$. It can be easily extended that the inner product $\langle \cdot, \cdot \rangle$ between two arbitrary points preserves the rotation-invariant property. In addition, the cross product has the property $R\mathbf{a} \times R\mathbf{b} = R(\mathbf{a} \times \mathbf{b})$ under proper rotations $\boldsymbol{R}$. We define the Darboux frame at $\boldsymbol{q}_i$ as a triple tuple: $\mathcal{O}_i = \langle \boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{w}_i \rangle$. By applying a rotation matrix to the point set, we can get $\boldsymbol{q}_j = \boldsymbol{R}\boldsymbol{q}_i$ with the corresponding Darboux frame $\mathcal{O}_j = \left\langle \boldsymbol{u}_j, \boldsymbol{v}_j, \boldsymbol{w}_j \right\rangle$. As a result, we can conclude that $\mathcal{O}_j = \boldsymbol{R}\mathcal{O}_i = \langle \boldsymbol{R}\boldsymbol{u}_i, \boldsymbol{R}\boldsymbol{v}_i, \boldsymbol{R}\boldsymbol{w}_i \rangle$. The PF descriptor is proven to be rotation-invariant:

$$d_{st,j} = \left\|\boldsymbol{p}_{t,j} - \boldsymbol{p}_{s,j}\right\|_2 = \|\boldsymbol{R}\boldsymbol{p}_{t,i} - \boldsymbol{R}\boldsymbol{p}_{s,i}\|_2 = d_{st,i} = d_{st} \tag{13}$$

$$\alpha_j = \left\langle \boldsymbol{v}_j, \boldsymbol{n}_{t,j} \right\rangle = \left\langle \boldsymbol{R}\boldsymbol{v}_i, \boldsymbol{R}\boldsymbol{n}_{t,i} \right\rangle = \left\langle \boldsymbol{v}_i, \boldsymbol{n}_{t,i} \right\rangle = \alpha_i \tag{14}$$

$$\phi_j = \left\langle \boldsymbol{u}_j, \frac{(\boldsymbol{p}_{t,j} - \boldsymbol{p}_{s,j})}{d_{st}} \right\rangle = \left\langle \boldsymbol{R}\boldsymbol{u}_i, \boldsymbol{R}\frac{(\boldsymbol{p}_{t,i} - \boldsymbol{p}_{s,i})}{d_{st}} \right\rangle = \phi_i \tag{15}$$

$$\begin{aligned} \theta_j &= \operatorname{atan2}\left(\left\langle \boldsymbol{w}_j, \boldsymbol{n}_{t,j} \right\rangle, \left\langle \boldsymbol{u}_j, \boldsymbol{n}_{t,j} \right\rangle\right) \\ &= \operatorname{atan2}\left(\left\langle \boldsymbol{R}\boldsymbol{w}_i, \boldsymbol{R}\boldsymbol{n}_{t,i} \right\rangle, \left\langle \boldsymbol{R}\boldsymbol{u}_i, \boldsymbol{R}\boldsymbol{n}_{t,i} \right\rangle\right) = \theta_i \end{aligned} \tag{16}$$

C. Geometry RIF-Descriptor

The SH-energy and PF descriptors are shown to be rotation-invariant descriptors. The SH-energy descriptor focuses on capturing the global features of the 3D shape and the PF descriptor aims at describing the local features. It is straightforward to concatenate both descriptors and this results in the rotation-invariant feature (RIF)-descriptor at $\boldsymbol{q}_i$

$$\mathcal{X}_{rif,i} = \left[(\mathcal{X}_{sh,i}, \mathcal{X}_{pf,i,0}), \quad \dots, \quad (\mathcal{X}_{sh,i}, \mathcal{X}_{pf,i,k})\right]^{\mathrm{T}}, \tag{17}$$

where $\mathcal{X}_{rif,i} \in \mathbb{R}^{k\times(n_{\mathrm{sh}}+n_{\mathrm{pf}})}$ with point feature descriptor

$$\mathcal{X}_{pf,i,j} = [d_{i,j}, \alpha_{i,j}, \phi_{i,j}, \theta_{i,j}, r_{s,i}] \in \mathbb{R}^{n_{\mathrm{pf}}}, \tag{18}$$

with $n_{\mathrm{pf}} = 5$ and the spherical harmonics energy descriptor

$$\mathcal{X}_{sh,i} = \left[\|f_{0,i}\|, , \dots, \left\|f_{n_{\mathrm{sh,deg}},i}\right\|\right] \in \mathbb{R}^{n_{\mathrm{sh}}}, n_{\mathrm{sh}} = n_{\mathrm{sh,deg}} + 1 . \tag{19}$$

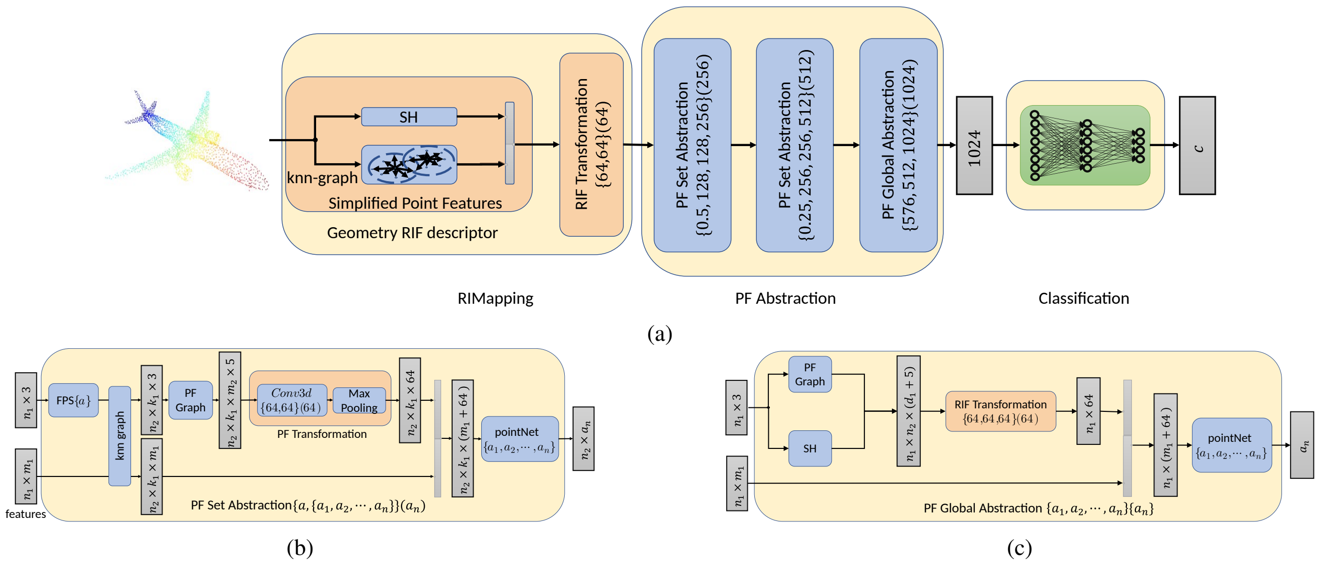

SH
knn-graph
Simplified Point Features
Geometry RIF descriptor
RIF Transformation {64,64}(64)
PF Set Abstraction {0.5, 128, 128, 256}(256)
PF Set Abstraction {0.25, 256, 256, 512}(512)
PF Global Abstraction {576, 512, 1024}(1024)
1024
c
RIMapping
PF Abstraction
Classification
(a)
FPS{a}
knn graph
PF Graph
Conv3d {64,64}(64)
Max Pooling
PF Transformation
pointNet {a1, a2, ⋯, an}
features
PF Set Abstraction{a, {a1, a2, ⋯, an}}(an)
(b)
PF Graph
SH
RIF Transformation {64,64,64}(64)
pointNet {a1, a2, ⋯, an}
PF Global Abstraction {a1, a2, ⋯, an}{an}
(c)

Fig. 2: The model architecture (a) consists of the RIMapping, PF Abstraction, and Classification blocks. In the RIMapping block, a spherical harmonics energy descriptor and simplified point features (SPF) at each point are computed to form a Geometry RIF descriptor, which is fed to a RIF Transformation to extract a high-level feature. In the PF Abstraction Block, we have two PF Set Abstraction layers (b) together with a PF Global Abstraction layer (c) to obtain a final global feature, which is used in a fully-connected Classification network.

IV. Network Architecture

The proposed rotation-invariant network consists of three blocks: RIMapping, PF Abstraction, and Classification, where the latter is a feed-forward network. Our main contributions are on the design of the RIMapping and PF Abstraction blocks.

A. Pre-Alignment with PCA

In an experiment on rotation invariance, we transfer a point cloud set $\boldsymbol{q}$ with an arbitrary rotation matrix $\boldsymbol{R}$, resulting in a rotated clone $\boldsymbol{q}^1 = \boldsymbol{R}\boldsymbol{q}$. According to [22], we pre-align each input $\boldsymbol{q}$ based on the PCA to its principle axes, which are indicated as orthonormal coordinates $\boldsymbol{R}_0$, with the formula of $\boldsymbol{q}_1 = \boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{q}$. It can be shown that the PCA alignment for a rotated clone $\boldsymbol{q}^1$ with corresponding orthonormal coordinates $\boldsymbol{R}_1 = \boldsymbol{R}\boldsymbol{R}_0$ leads to $\boldsymbol{q}_2 = \boldsymbol{R}_1^{\mathrm{T}}\boldsymbol{q}^1 = \boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{q} = \boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{q} = \boldsymbol{q}_1$. Therefore, pre-alignment can reduce the impact of the rotation matrix on the network.

B. RIMapping Block

The RIMapping block consists of the Geometry RIF descriptor and RIF Transformation. For acquiring the rotation-invariant features, we leverage the spherical harmonics and point feature representation with a KNN to enrich geometric features for the point cloud, which is represented as a rotation-invariant descriptor with a size of $\mathbb{R}^{n\times k_1\times(n_{\mathrm{sh}}+n_{\mathrm{pf}})}$ and k_1-neighborhood that can globally and locally manifest a 3D shape. This descriptor provides low-level geometric clues for high-level geometric feature learning, realized with a RIF Transformation. The RIF Transformation layer utilizes a mini PointNet (without input and feature transformation), consisting of a set of shared Conv2d layers with kernel size equal to 1, to extract a global feature employing a max-pooling operator. The output of RIF Transformation is indicated as embedding feature with a size of $\mathbb{R}^{n\times a_n}$. The PF Transformation has the same structure as the RIF Transformation, apart from the different input sizes. Both Transformations intend to aggregate the local details by calculating a weighted average of neighboring features through a shared local fully-connected layer.

C. PF Abstraction Block

*1) **PF Set Abstraction**:* The extracted information from the RIMapping block is still insufficient for the precise classification task, as max-pooling can only describe an outline and some local details could be omitted. To address the problem, we propose the PF Set Abstraction Layer to hierarchically exploit the local features, which consists of the sampling layer, grouping layer, and PointNet layer. PointNet++ inspires PF Set Abstraction. However, there are several significant adaptations inside the grouping layer. In the sampling layer, we use iterative farthest point sampling (FPS) to obtain n_2 points, indicated as $\boldsymbol{P}_i, i \in [0, \cdots, n_2]$. Each point $\boldsymbol{P}_i$ is the center of a local region C_i. In the sequence, a KNN graph is built at point $\boldsymbol{P}_i$ to obtain k_1-neighborhood, indicated as $[\boldsymbol{P}_{0,i}, \boldsymbol{P}_{1,i}, \cdots, \boldsymbol{P}_{k_1,i}]$, with $i \in [0, \cdots, n_2]$. In contrast to PointNet++, which combines the point with the feature from the last layer and works as input for the PointNet layer, we utilize a PF graph to convert a point to a rotation-invariant feature, where PF graph is a fully connected graph (Fig. 1c) and built at each local region C_i. Then, we can get a feature with a size of $\mathbb{R}^{k_1-1\times n_{\mathrm{pf}}}$ for each point $\boldsymbol{P}_{n_k,i}$ in the region C_i. In the end, a new rotation-invariant point feature for all local regions is obtained, indicated as $\boldsymbol{f}_{PF} \in \mathbb{R}^{n_2\times k_1\times k_1-1\times n_{\mathrm{pf}}}$.

Sequentially, we apply the PF Transformation to extract a feature for each local region. We concatenate the previous feature at each center point $\boldsymbol{P}_i$ with the newly extracted feature to form a new feature representation and feed it to the PointNet layer.

*2) **PF Global Abstraction**:* The global abstraction is the successor layer to the PF Set Abstraction layer, which reduces the original input cloud to $\boldsymbol{X}_2 \in \mathbb{R}^{n_1 \times 3}$. We build a PF graph at the reduced point set and concatenate it with its spherical energy descriptor, which leads to a new representation with a size of $\mathbb{R}^{n_1 \times (n_1 - 1) \times (n_{\mathrm{pf}} + n_{\mathrm{sh}})}$. In the sequence, we apply the RIF Transformation for extracting a new feature representation. This new feature will concatenate with the feature from the PF Set Abstraction layer. In the end, a mini PointNet is applied to obtain a global feature.

V. Experiments

We evaluate our approach regarding rotation robustness and compare it with other state-of-the-art methods. We use ModelNet40 [30] as data set for validating the effectiveness of the proposed network structure. ModelNet40 consists of 40 categories in the form of CAD models (mostly human-made). We use the official split with 9843 shapes for training and 2468 for testing. We apply the farthest point sampling algorithm to obtain 1024 points on mesh faces according to the face area and then shift and normalize the point into a unit sphere with centroid on the origin. During training, we use Adam [31] for 200 epochs with an initial learning rate of 10^{-3}. The algorithm is implemented with PyTorch on Linux with one GeForce RTX 2080Ti GPU.

A. Evaluation of Rotation Robustness

For evaluating the property of rotation robustness, we multiply each point cloud from ModelNet40 with a randomly sampled rotation matrix. Based on the same principle as [20], we evaluate our model using three different settings: a) training and testing with azimuthal rotations (z/z), b) training and testing with arbitrary rotations (SO(3)/SO(3)), c) training with azimuth rotations while testing with arbitrary rotations (z/SO(3)). The results are presented in Table I. It can be seen that most networks exhibit a sharp drop in performance in the settings SO(3)/SO(3) and z/SO(3), in particular in the latter one. The network DGCNN [18] shows an outstanding performance in the setting z/z with an accuracy of 92.2 %, while it only achieves 21.1 % in z/SO(3). DGCNN applies the point directly for classification, which changes dramatically when applying a rotation matrix in SO(3). As mentioned before, that point cloud is not a rotation-invariant representation. This is also a common problem for the PointNet-based network. The spherical CNN [20] uses a spherical harmonics-based convolution layer by rasterizing the point cloud, which shows a significant improvement in the setting of z/SO(3). However, the difference between its best performance z/z and z/SO(3) is still significant with a value of 10.3 %. ClusterNet [23] uses the RRI representation together with a cluster abstraction to increase classification performance with a result of 87.1 % in each setting. Note that the result of ClusterNet is directly cited from [23], as the code is not available as open source. Table I demonstrates that our approach achieves the best performance in the settings z/SO(3) and SO(3)/SO(3) with $n_{\mathrm{sh,deg}} = 20, 30$. The difference in the results between each setting is very small, approximately 0.2 %. Based on these observations, we can conclude that our proposed network shows the best performance regarding rotation robustness.

TABLE I: Comparison of rotation robustness on rotation-augmented ModelNet40 benchmark. Our proposed network shows the best performance in the settings z/SO(3) and SO(3)/SO(3). Note, values are given as a percentage.

Method	Input(size)	z/z	z/SO(3)	SO(3)/SO(3)
PointNet [16]	pc (1024×3)	89.2	14.7	83.6
PointNet++ [17]	pc (1024×3)	89.3	28.6	85.0
VoxNet [4]	voxel (30^3)	83.0	-	73.0
RotationNet 20x [14]	voxel (20×224^2)	92.4	20.0	80.0
SO-Net [32]	pc+normal (5000×6)	**92.6**	21.1	80.2
DGCNN [18]	pc (1024×3)	92.2	33.5	81.1
Spherical CNN [20]	voxel 2×64^2	88.9	78.6	86.9
ClusterNet [23]	pc (1024×3)	87.1	87.1	87.1
ours($n_{\mathrm{sh,deg}}$=20)	pc(1024×3)	88.4	88.6	88.5
ours($n_{\mathrm{sh,deg}}$=30)	pc(1024×3)	88.6	**88.7**	**88.8**

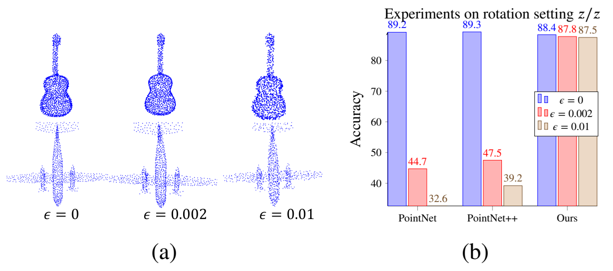


Fig. 3: (a) shows the point cloud with different noise levels and (b) shows the comparison results against point perturbations.

B. Robustness Tests

1) Evaluation of model against point cloud perturbation: For further evaluation of robustness against perturbation, we conducted experiments by adding perturbation at each point. Many studies have shown that deep learning-based networks can be fooled by using an adversarial attack. Following the same principle, we add a perturbation value to each point with $\|\delta\| < \epsilon$, where ϵ is set between 0.002 and 0.01, as shown in Fig. 3a. The results are listed in Fig. 3b. It can be seen that in these two different perturbation levels, our network with $n_{\mathrm{sh,deg}} = 20$ is more robust under perturbation when compared to PointNet and PointNet++.

2) Evaluation of model against point cloud density: The point cloud density also plays an important role in the classification task. In this section, we downsample ModelNet40 to different point densities in the range of 1024 to 128 by using the farthest point sampling strategy (FPS) or random input droupout (DP). The downsampled point clouds are shown in Fig. 4a and the corresponding classification

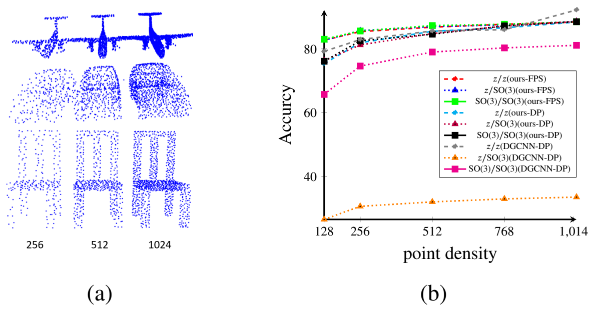


Fig. 4: (a) shows the downsampled point cloud and (b) illustrates the comparison results of different point densities in three rotation settings with FPS and DP.

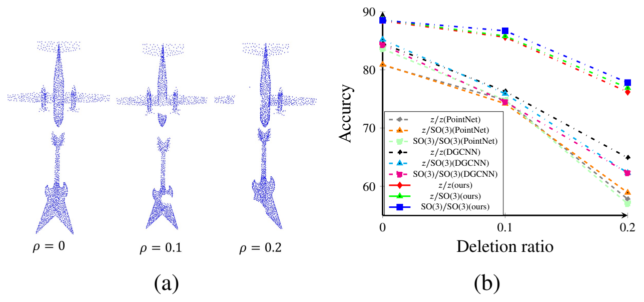


Fig. 5: (a) shows the partial point cloud with the deletion ratio ρ from 0.1 to 0.2 and (b) shows the results.

accuracy is illustrated in Fig. 4b. It is worth noting that our proposed network is very robust against point density changes in these three rotation setups, which decreases the classification accuracy from 88.6 % to 82.7 % by FPS and varies from 88.6 % to 75.4 % by DP. In comparison to DGCNN [18], the results vary from the 92.2 % to 79.2 % in z/z and from 33.5 % to 26.5 % in $z/\mathrm{SO}(3)$. Under the same point density, our model shows no significant change, which further verifies our model's robustness.

3) Evaluation of model against partial point cloud: In reality, we can only get a partial point cloud by using a single stationary camera. To evaluate the partial point cloud classification model, we train our model with a complete point cloud and test against a partial point cloud. The partial point cloud is obtained by first deleting the completed point cloud with a ratio ρ from 0.1 to 0.2 and then using iterative FPS (Fig. 5a). The results are illustrated in Fig. 5b. We compare our model with PointNet and DGCNN under three rotation settings. Training and testing data set are rotated with a PCA algorithm to reduce the impact of arbitrary rotation. From Fig. 5b, we can conclude that our model shows the best performance and far exceeds the other two classification models in all three experiments.

C. Ablation Studies

1) Analysis of architecture design: To evaluate our network architecture's effectiveness, we use PointNet as the baseline and connect our individually designed component to it. Note that we realign all data in this section with

TABLE II: Effectiveness of designed network block.

Method	z/z	$z/\mathrm{SO}(3)$	SO(3)/SO(3)	mean
PointNet [16]	89.20	14.70	83.60	62.40
PCA+PointNet	80.90	80.90	80.80	80.84
RIMapping($n_{\mathrm{sh,deg}}$=20)+PointNet	82.00	83.20	84.50	83.23
ours(without SH)	85.40	85.70	86.20	85.77
ours($n_{\mathrm{sh,deg}}$=20)	88.40	88.60	88.50	88.50

TABLE III: Effectiveness of maximum degree $n_{\mathrm{sh,deg}}$.

$n_{\mathrm{sh,deg}}$	z/z	z/SO(3)	SO(3)/SO(3)
8	88.10	87.60	87.60
15	87.80	88.30	87.80
20	88.40	88.60	88.50
30	88.60	88.70	88.80

PCA (Section IV-A). In Table II, we can see that it shows a significant improvement when compared to the original PointNet in the setting of $z/\mathrm{SO}(3)$ and that the average accuracy rate has increased about 18 %. Furthermore, we analyzed the effectiveness of the RIMapping block by connecting it to PointNet. The results listed in Table II show that the accuracy in $z/\mathrm{SO}(3)$ and SO(3)/SO(3) improved by 2.3 % and 3.7 %. Comparing our proposed network's worst performance shows that our PF Abstraction block helps in improving the final accuracy in all three settings. We also evaluated the effectiveness of the spherical harmonics energy descriptor. The results are listed in Table II. Without the spherical harmonics energy descriptor, the accuracy is worse when compared against our original design. However, it still shows better performance when compared to the PointNet-based network.

2) Effectiveness of maximum degree of spherical harmonics $n_{\mathrm{sh,deg}}$: The spherical harmonics descriptor is an essential aspect of our network. Based on the information loss, a higher degree of spherical harmonics leads to a smaller information loss. However, it will also increase the computational complexity to $\mathcal{O}(n^2)$. For evaluating the effectiveness of the $n_{\mathrm{sh,deg}}$, we vary the degree. The results are listed in Table III and it can be seen that the higher $n_{\mathrm{sh,deg}}$, the better the final classification accuracy.

VI. Conclusion

We presented a rotation-invariant point cloud-based neural network, which utilizes a global spherical harmonics feature and a local points feature to achieve rotation-invariant properties. Furthermore, a new neural network structure is designed, inspired by PointNet++, but with several adaptations such as PF graph and PF Transformation. The network is applied to 3D object classification, but can be extended to part segmentation and instance segmentation. Via several experiments, we have shown that our network can deal with arbitrary input orientations and achieve competitive performance compared to other state-of-the-art approaches on the ModelNet40 data set. Furthermore, our network shows robustness against point perturbations, point density, and partial point cloud.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] T. S. Cohen and M. Welling, "Transformation properties of learned visual representations," *Proceedings of the International Conference on Learning Representations*, 2015.

[3] T. S. Cohen, M. Geiger, J. Khler, and M. Welling, "Spherical CNNs," in *Proceedings of the International Conference on Learning Representations*, 2018.

[4] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 922–928.

[5] R. Kondor, Z. Lin, and S. Trivedi, "Clebsch-Gordan Nets: a fully fourier space spherical convolutional neural network," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 117–10 126.

[6] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5028–5037.

[7] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3D shape descriptors," in *Proceedings of the Eurographics Symposiumon on Geometry Processing*, 2003, pp. 156–164.

[8] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3D point clouds," in *Proceedings of the International Conference on Intellligent Autonomous Systems*, 2008.

[9] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3212–3217.

[10] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[11] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[12] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3D object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 186–194.

[13] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 945–953.

[14] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5010–5019.

[15] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3577–3586.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, pp. 1–12, 2019.

[19] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[20] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning SO(3) equivariant representations with spherical CNNs," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 52–68.

[21] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

[22] M. Kazhdan and T. Funkhouser, "Harmonic 3D shape matching," in *ACM SIGGRAPH Conference Abstracts and Applications*, 2002, pp. 191–191.

[23] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin, "ClusterNet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4994–5002.

[24] R. Gilmore, *Lie Groups, Physics, and Geometry: An Introduction for Physicists, Engineers and Chemists*. Cambridge University Press, 2008.

[25] V. Andrearczyk, V. Oreiller, J. Fageot, X. Montet, and A. Depeursinge, "Solid Spherical Energy (SSE) CNNs for efficient 3D medical image analysis," *Proceedings of the Irish Machine Vision and Image Processing Conference*, pp. 37–44, 2019.

[26] M. Reisert and H. Burkhardt, "Using irreducible group representations for invariant 3D shape description," in *Proceedings of the Joint Pattern Recognition Symposium*. Springer, 2006, pp. 132–141.

[27] C. R. Nortje, W. O. Ward, B. P. Neuman, and L. Bai, "Spherical harmonics for surface parametrisation and remeshing," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[28] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.

[29] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4.

[30] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] J. Li, B. M. Chen, and G. Hee Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9397–9406.

RightsLink

Sign in/Register

Requesting permission to reuse content from an IEEE publication

Deep Hierarchical Rotation Invariance Learning with Exact Geometry Feature Representation for Point Cloud Classification

Conference Proceedings:
2021 IEEE International Conference on Robotics and Automation (ICRA)

Author: Jianjie Lin; Markus Rickert; Alois Knoll

Publisher: IEEE

Date: 30 May-5 June 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636483

PCTMA-Net: Point Cloud Transformer with Morphing Atlas-based Point Generation Network for Dense Point Cloud Completion

Jianjie Lin, Markus Rickert, Alexander Perzylo and Alois Knoll

Abstract— **Inferring a complete 3D geometry given an incomplete point cloud is essential in many vision and robotics applications. Previous work mainly relies on a global feature extracted by a Multi-layer Perceptron (MLP) for predicting the shape geometry. This suffers from a loss of structural details, as its point generator fails to capture the detailed topology and structure of point clouds using only the global features. The irregular nature of point clouds makes this task more challenging. This paper presents a novel method for shape completion to address this problem. The Transformer structure is currently a standard approach for natural language processing tasks and its inherent nature of permutation invariance makes it well suited for learning point clouds. Furthermore, the Transformer's attention mechanism can effectively capture the local context within a point cloud and efficiently exploit its incomplete local structure details. A morphing-atlas-based point generation network further fully utilizes the extracted point Transformer feature to predict the missing region using charts defined on the shape. Shape completion is achieved via the concatenation of all predicting charts on the surface. Extensive experiments on the Completion3D and KITTI data sets demonstrate that the proposed PCTMA-Net outperforms the state-of-the-art shape completion approaches and has a 10% relative improvement over the next best-performing method.**

I. Introduction

The use of point clouds as a format of shape representation has increased in the last years due to the rapid development of 3D acquisition technologies such as Lidar and depth cameras. The limited sensor resolution, occlusion, and camera angles however make it challenging to obtain a point cloud representation of the complete shape of an object. As a result, the acquired raw points are typically sparse, noisy, and miss large regions. On the other hand, complete 3D shapes are essential in vision applications, such as semantic segmentation and SLAM [1]. A complete 3D shape can improve the performance of CAD model-based point registration [2] and enables more flexible grasp planning [3], [4]. In this work, we focus on completing partial 3D shapes that suffer from occlusion and limited sensor resolution.

Previous work [5], [6], [7] principally followed the encoder-decoder paradigm framework by extracting a latent global feature from an incomplete point cloud. Decoders leverage these feature to predict missing regions. Benefiting from PointNet-based [8] feature extractor networks, the task of shape completion made tremendous progress in recent years. However, the extracted global features from PointNet ignore the geometric relationship within the point clouds due to the max-pooling operation. As a result, these approaches suffer from a loss of structural detail in the reconstruction.

The intuitive solution is to make up for the shortcomings of the PointNet by excavating the semantic affinity within the point cloud. Therefore, we propose a novel framework named Point Cloud Transformer with Morphing-Atlas-based Point Generation Network for Shape Completion (PCTMA-Net) to address this problem. The Transformer [9] is a standard framework for natural language processing and has been further extended to vision tasks for image recognition [10], as well as point cloud classification and segmentation [11]. The Transformer follows the encoder-decoder structure and consists of four main modules: input embedding, positional encoding, (self-)attention mechanism, and positional feed-forward. In this work, we apply only the encoder module and neglect the positional encoding module due to the point cloud's irregular nature. The Transformer's central core is the attention mechanism, which can generate refined attention features by leveraging the global context. The attention weight between any two positions is updated by the dot product of query and key vector. The weighted sum of all attention weights is the attention feature. The concept of query, key, and value vector makes it possible to match and learn the global context. The attention feature of each word is related to all input features. Furthermore, the permutation invariant nature of softmax, dot product, and point-wise feed-forward network makes it well-suited for point cloud learning. The offset attention mechanism introduced in [11] uses the idea of the Laplacian matrix to improve the attention performance further. In this work, we replace the original attention design with the offset attention mechanism. The morphing-atlas-based point generation network is the decoder component in our overall structure. The extracted global feature from the Transformer is further utilized to generate the points. An atlas, as defined in topology, consists of a set of charts on a surface. Therefore, we assume that a missing region of the surface can be recovered by a chart. Based on this assumption, we duplicate the Transformer feature and concatenate it with a predefined grid. We utilize the idea of multi-head attention by linearly projecting the concatenated features to learn n_{chart} different features, where each feature is responsible for generating a chart defined on the surface. We quantitatively and qualitatively evaluated the proposed PCTMA-Net on the Completion3D data set and demonstrate a 10% relative improvement over the next best-performing method for the task of shape completion.

Jianjie Lin, Markus Rickert, Alexander Perzylo, Alois Knoll are with Robotics, Artificial Intelligence and Real-time Systems, Department of Informatics, Technische Universität München, Munich, Germany {jianjie.lin, rickret, perzylo, knoll}@in.tum.de

Furthermore, the qualitative evaluation on the KITTI data set shows that our proposed network is able to predict more structural details than other state-of-the-art approaches.

Our contributions are summarized as follows: (1) We propose a novel shape completion framework named Point Cloud Transformer with Morphing-Atlas-based point generation Network for shape completion (PCTMA-Net), which is inherently permutation-invariant and has the capability of learning the global context within the point clouds and preserving structural details. (2) The integration of the concept of an atlas and the multi-head attention mechanism leads to the generation of high-resolution, high-fidelity, and fine-grained shapes. (3) Extensive experiments are conducted on the Completion3D benchmarks, and the KITTI data set, which indicate that the proposed networks remarkably outperforms other competitive methods.

II. Related Work

Shape completion approaches made significant progress in recent years due to the rapid development of deep learning and 3D acquisition technologies. We can roughly categorize the existing work into volumetric-based and multilayer perceptron-based networks from the perspective of network structure and the underlying 3D data representation.

Volumetric-based shape completion: The extension of CNN to 3D convolutional neural networks can be used for dealing with a shape in the volumetric representation [12], [13]. Notable work such as 3D-Encoder-Predictor Networks (3D-EPN) [14] progressively reconstruct the 3D volumetric shape. The work in [15] directly generates the high resolution 3D volumetric shape by combining the global structure with the refinement of local geometry, while [16] introduced a variational auto-encoder to learn a shape prior to inferring the latent representation of complete shapes. GRNet [17] took one step further by introducing Gridding and Gridding Reverse to convert between point clouds and 3D grids. However, a quantization effect is introduced during the transformation of point clouds into a 3D volumetric representation. The computational costs increase cubically to the resolution and therefore make it more challenging to process fine-grained shapes.

Multilayer perceptron (MLP)-based shape completion: Point clouds can be directly obtained by several acquisition techniques. It is much more efficient compared to the voxel-based representation when processing costs are compared. Inspired by PointNet [8] and its successor work [18],[19], several approaches use them for point cloud learning, as the point-wise MLP enables the handling of irregular point clouds and aggregating features using a symmetric function. However, the PointNet network suffers from a loss of structure details. The current state-of-the-art approaches for shape completion such as AtlasNet [6], PCN [20] and Folding-Net [7] use PointNet as their baseline to extract global features and to apply a decoder to predict the missing regions. Unlike PCN and FoldingNet, AtlasNet completes the shape by generating surface elements utilizing the atlas concept. TopNet [5] improves the decoder by using a hierarchical rooted tree. By combining reinforcement learning with an adversarial network, RL-GAN-Net [21] and Render4Completion [13] propose a reinforcement learning agent-controlled GAN to improve the quality and consistency of the generated complete shape. However, most of these studies suffer from information loss on structural details, as they predict the whole point cloud only from a single global shape representation. SA-Net [22] extended these approaches with a skip-attention mechanism to preserve more structural details. PF-Net [23] introduced a point pyramid decoder to generate a shape in different resolution levels.

III. The Architecture of PCTMA-Net

A. Overview

The overall structure of PCTMA-Net is illustrated in Fig 1, which aims to learn a semantic affinity within a partial point cloud by using a Transformer encoder. The complete 3D shape is reconstructed with a morphing-atlas decoder utilizing the extracted feature from the Transformer encoder. We formulate the whole shape completion pipeline as: Given a partial point cloud, indicated as $\mathcal{P}$ with N_{in} points, where each point is represented in 3D coordinates $\mathbf{x} = [x_i, y_i, z_i]$, we first convert this partial point cloud into a feature vector $\mathbf{F}_0$ by a PointNet. The difference to previous work [7], [6], which relies on only the global feature for shape completion, is that we further utilize the Transformer encoder to process the feature to obtain a piece of semantic affinity information for predicting the missing regions. The extracted feature is later fed to the morphing-atlas point generator for completing the shape.

B. Point Cloud Transformer Encoder

The Transformer encoder of PCTMA-Net first transforms an incomplete point cloud to the feature space using an input embedding network. We then feed the extracted feature to $N\times$ stacked encoder layers, where they share a similar philosophy of design as the original paper [9], except for the attention mechanism. The purpose of the encoder layer is to learn a discriminate representation for each point. The encoder can be mathematically formulated in the following: By given a partial point cloud $\mathcal{P} \in \mathcal{R}^{N_{\text{in}}\times d}$ with N_{in} points each having a d-dimensional feature description, an embedding feature $\mathbf{F}_0$ is firstly learned with an input embedding network, indicated as $\mathrm{F}_{\text{embedding}}$. The difference to the embedding network presented in [11] is that we defined $\mathrm{F}_{\text{embedding}}$ as a PointNet followed by a max-pooling operator. As a result, we acquire a d_{model}-dimensional embedding feature $\mathbf{F}_0 \in \mathcal{R}^{d_{\text{model}}}$ instead of $\mathbf{F}_0 \in \mathcal{R}^{d_{\text{model}}\times N_{\text{in}}}$[11]. It will improve the shape completion performance, as the $\mathbf{F}_0$ after max-pool operator can reduce redundant information and make the training more efficient. The global feature $\mathbf{F}_0$ is later fed to $\mathrm{F}_{\text{encoder}_i}$:

$$\mathbf{F}_i = \mathrm{F}_{\text{encoder}_i}(\mathbf{F}_{i-1}),\ i = [1, \ldots, N]. \quad (1)$$

Furthermore, we concatenate the features from each encoder layer and follow up by two cascade LBR layers to form an

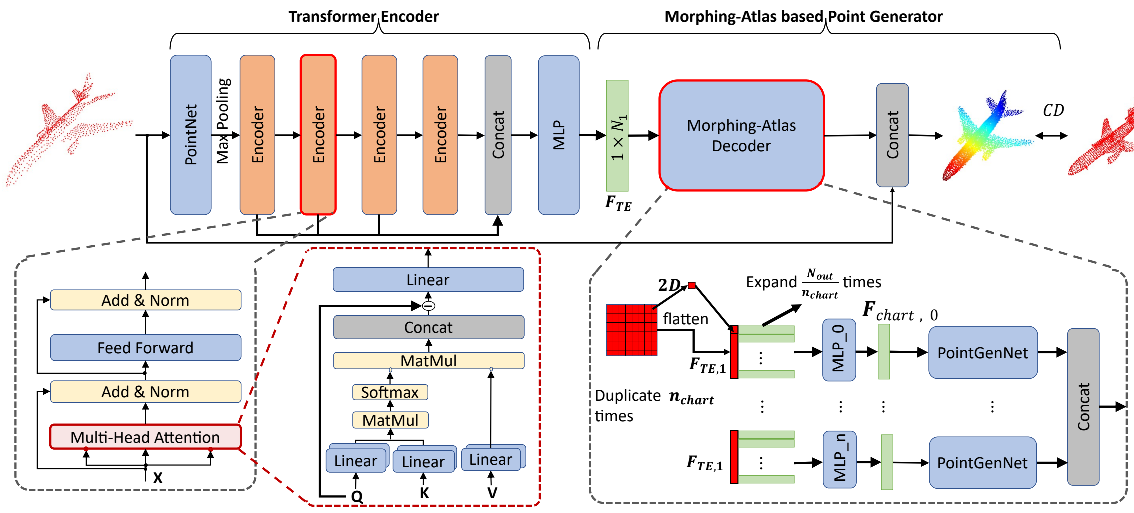


Fig. 1: The overall structure of PCTMA-Net. The whole structure consists of a Transformer encoder and morphing-atlas point generation decoder. The Transformer encoder aims to extract features from the input point clouds by using an $N\times$ stacked encoder layer which consists of an attention mechanism and positional feed-forward network. The morphing-atlas-based surface reconstruction decoder uses multi-chart point generation networks for point cloud completion by concatenating the features from the Transformer encoder and mesh grid.

effective global feature

$$\mathbf{F}_\text{e} = \text{BatchNorm}(\mathbf{F}_i \oplus \cdots \oplus \mathbf{F}_N) \tag{2}$$
$$\mathbf{F}_\text{TE} = \text{LBR}\big(\text{LBR}(\mathbf{F}_\text{e})\big)\,, \tag{3}$$

where $\mathbf{F}_\text{i} \in \mathcal{R}^{d_\text{model}}$, $\mathbf{F}_e \in \mathcal{R}^{N\times d_\text{model}}$ and $\mathbf{F}_\text{TE} \in \mathcal{R}^{d_\text{model}}$. The operator $\oplus$ is denoted as concatenation, and the function LBR represents a linear layer followed by BatchNorm and ReLU operators. The F_{encoder_i} consists of two sub-layers, namely self-attention mechanism and positional forward feedback:

$$\text{F}_{\text{encoder}_i}\big(\mathbf{F}_{i-1}\big) = \text{FFN}_i\big(\text{attention}_i(\mathbf{F}_{i-1})\big), \tag{4}$$
$$\text{FFN}_i(\mathbf{x}) = \text{LBR}_{i,1}\big(\text{LBR}_{i,0}(\mathbf{x})\big) + \mathbf{x}\,. \tag{5}$$

The layer FFN_i is a shared positional forward feedback network comprising two cascaded LBRs with the size of $[d_\text{ff}, d_\text{model}]$, where $d_\text{ff} = 2048$ and $d_\text{model} = 1024$.

a) Offset self-attention mechanism: Self-attention is a mechanism that calculates the semantic relationship between different elements within a sequence of data. In the context of point cloud processing, attention is employed to build weights between every two positions in the feature space. In comparison to k-nearest neighbors algorithms, the attention mechanism has a larger receptive field. Furthermore, the attention mechanism's permutation invariant property makes it suitable for disordered, irregular data representation such as point clouds. The work in [11] proposed the offset attention by utilizing the idea of a Laplacian matrix $L = D - E$, where E is the adjacent matrix E and D is the diagonal matrix. The attention mechanism is adopted as

$$\begin{aligned}\mathbf{F}_\text{sa,out} &= \text{attention}(\mathbf{F}_\text{sa,in}) \\ &= \text{LBR}(\mathbf{F}_\text{sa,in} - \mathbf{F}_\text{sa}) + \mathbf{F}_\text{in}\,.\end{aligned} \tag{6}$$

The remaining part of the attention computation operators still follows the same design as in the original paper [9]. The self-attention feature $\mathbf{F}_\text{sa}$ in (6) concatenates the multi-head attention with the following formulation:

$$\mathbf{F}_\text{sa} = \text{Linear}(\mathbf{F}_{\text{head}_1} \oplus \cdots \oplus \mathbf{F}_{\text{head}_h})\,, \tag{7}$$

where the attention feature at the i-head $\mathbf{F}_{\text{head}_i}, i \in [1,\ldots,h]$ is formulated as

$$\mathbf{F}_{\text{head}_i} = \text{softmax}\Big(\frac{\hat{\mathbf{Q}}\hat{\mathbf{K}}^T}{\sqrt{d_k}}\Big)\hat{\mathbf{V}}\,, \tag{8}$$

with $\hat{\mathbf{Q}} = \text{Linear}(\mathbf{Q})$, $\hat{\mathbf{K}} = \text{Linear}(\mathbf{K})$, $\hat{\mathbf{V}} = \text{Linear}(\mathbf{V})$. The variables $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are projected with a different linear layer, respectively. Following the same principle as the original paper, we set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{F}_\text{sa,in} \in \mathcal{R}^{d_\text{model}}$. We reshape the linear projected query and key as $\hat{\mathbf{Q}}, \hat{\mathbf{K}} \in \mathcal{R}^{d_\text{model}\times 1}$ to obtain the attention weights $\mathbf{A}$ by matrix dot product via $\hat{\mathbf{Q}}\hat{\mathbf{K}}^T$. We normalize $\mathbf{A}$ with $\sqrt{d_k}$ to avoid large values in magnitude, where $d_k = \frac{d_\text{model}}{h}$. The equation in (8) shows, that the self-attention $\mathbf{F}_{\text{head}_i}$ is equal to the weighted sums of the value vector Linear($\mathbf{V}$) using the corresponding attention weights. The multi-head attention mechanism can jointly capture information from different representation subspace at different positions [9]. Therefore, it can efficiently preserve and capture the point cloud's detailed topology and structure for predicting the missing regions in comparison to [5], [6].

C. Morphing-Atlas-Based Point Generation Network

At the first stage, the Transformer encoder extracts a global feature $\mathbf{F}_\text{TE}$ for expressing an incomplete point cloud. We then feed the extracted features into a morphing-atlas-based point generator for predicting continuous and smooth

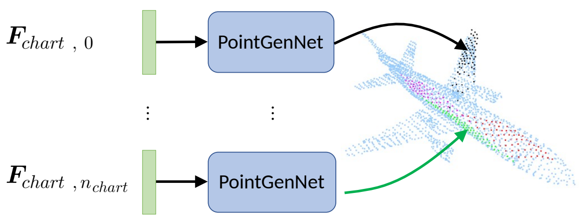


Fig. 2: Visualization of MA-Network

shapes. Atlas [6] is defined in the topology for describing a manifold and an atlas is composed of each chart that, roughly speaking, describes the local region of the manifold. In the context of 3D shapes, the manifold can be considered as a shaped surface. Therefore, we can represent a 3D shape by combing all the charts. Based on the Atlas concept, we define a chart as C_i and let a designed decoder $\mathcal{D}_i$ learn to map a 2D grid to a 3D surface. Furthermore, we introduce a hyper-parameter n_{chart} to control the number of charts defined on a shape to predict a smooth and high-resolution shape. The global feature $\mathbf{F}_{\text{TE}} \in \mathcal{R}^{d_{\text{model}}}$ is duplicated $N_{\text{out}}/n_{\text{chart}}$ times and then concatenated with a mesh grid to describe a new feature, denoted as $\mathbf{F}_{\text{TE},1} \in \mathcal{R}^{(d_{\text{model}}+2)\times(N_{\text{out}}/n_{\text{chart}})}$. It beneficial to linearly project $\mathbf{F}_{\text{TE},1}$ with different learned linear projections. This concept is similar to multi-head attention by allowing the model to obtain the shape features from different representation subspaces at different positions. Therefore, $\mathbf{F}_{\text{TE},1}$ is duplicated n_{chart} times and each $\mathbf{F}_{\text{TE},1}$ is fed to an MLP layer which produces a new hidden code, denoted as $\mathbf{F}_{\text{chart},i} \in \mathcal{R}^{(d_{\text{model}}+2)\times(N_{\text{out}}/n_{\text{chart}})}, i \in [1,\dots,n_{\text{chart}}]$. For each single chart, we feed $\mathbf{F}_{\text{chart},i}$ into a PointGenNetwork (Fig. 2), sharing the same structure as in [6]. All charts are concatenated to form a complete shape.

D. Evaluation Metrics

We apply the Chamfer distance (CD) [24] as a quantitative evaluation metric due to its efficient computation compared to the earth mover's distance [24]. The Chamfer distance measures the mean distance between each point in one point cloud to its nearest neighbor in another point cloud. Let $S_G = [x_i, y_i, z_i]_i^{n_G}$ be the ground truth and $S_R = [x_i, y_i, z_i]_i^{n_R}$ be the reconstructed point by given a partial point cloud. n_G and n_R indicate the number of points in S_G and S_R, respectively. The Chamfer distance d_{CD} of S_G and S_R with $L2$ norm is formulated as

$$d_{\text{CD}} = \frac{1}{n_R}\sum_{x\in S_R}\min_{y\in S_G}||x-y||^2 + \frac{1}{n_G}\sum_{y\in S_G}\min_{x\in S_R}||x-y||^2 . \quad (9)$$

E. Implementation details

We implemented PCTMA-Net in PyTorch, where the model is optimized with an Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, together with a CosineAnnealingLR scheduler. The number of encoder layers used in the Transformer encoder is set to 4, and we follow the original papers by setting the multi heads in the offset attention mechanism to 8. We trained the network on a Linux system with a 2.6 GHz Intel Core i7–6700HQ, 16 GB of RAM, and one Nvidia RTX 2080 Ti GPU.

IV. Experiments

We compare our proposed shape completion algorithm PCTMA-Net with other state-of-the-art approaches on two large scale data sets: Completion3D [5] and KITTI [26]. The Chamfer distance is employed as a metric in the evaluation.

A. Shape Completion on Completion3D Data Set

Completion3D [5] from ShapeNet [27] offers a data set, which consists of 28 974 training samples and 800 point cloud evaluation samples with a point resolution of 2048 for training and validation, respectively. In the comparison, we use different output resolutions and the quantitative results are summarized in Table I. Note that the results of FoldNet [7], SA-Net [22], and PCN [20] are cited from the Completion3D benchmark leaderboard. Table I shows that our PCTMA-Net algorithm outperforms the other methods in 6 out of 8 categories with the overall Chamfer distance of 9.48 for $N_{\text{out}} = 16\,152$ and $n_{\text{chart}} = 32$. The qualitative visualization of completion results shown in Fig. 3 indicates that our approach is able to predict more details. The performance in the quantitative and qualitative evaluations proves the Transformer encoder and the morphing-atlas decoder's effectiveness for predicting and preserving the shape details.

B. Shape Completion on Robustness of Input Resolution

The input resolution can greatly affect the performance of a neural network. In this section, we will study the robustness of input resolution on the different network structures. We downsample the evaluation data set from Completion3D to obtain four levels of input resolutions: 256, 512, 1024, and 2048. The visualization of these four levels of input resolutions is shown in Fig. 4a. All networks are trained on an input resolution of 2048 and output a fixed size of 16 384 points. For point resolutions less than 2048, we follow the principle in PCN [20] to select points from the input randomly and pad the input cloud to raise the number of points to 2048. We evaluate these four levels of input resolution on the Completion3D data set. The quantitative illustration in Fig. 4b indicates that our network has the best robustness and outperforms the other approaches in all four input resolutions experiments.

C. Shape Completion on KITTI data set

For a further study of the application area, we conduct experiments on the KITTI data set [26], which is collected from real-world Velodyne Lidar scans composed of 2401 highly sparse point clouds. Note that the KITTI data set does not include the ground truth in a quantitative evaluation. Therefore, we can only qualitatively visualize the shape completion results. Unlike other work [5], [17], which trains the network with only the car category in ShapeNet [27] and then evaluates the KITTI data set, we use the same trained network as in Section IV-A for evaluation. This evaluation strategy can show the capability of the generalization of

TABLE I: Point completion results on Completion3D with ground truth and input resolution (2048 points) compared using Chamfer distance (CD) with L^2 norm. The results are multiplied by 10^4. In our algorithm (PCTMA-Net), we set meshgrid = 0.05. The best result is highlighted in ***green***, and a lower value is better.

Methods	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Watercraft	Overall
AtlasNet ($N_{\text{out}} = 2048$) [6]	5.82	29.28	11.02	27.11	34.04	19.11	29.27	15.55	21.40
AtlasNet ($N_{\text{out}} = 16384$) [6]	5.50	19.89	9.23	21.17	30.99	15.34	21.67	14.64	17.31
FoldNet [7]	12.83	23.01	14.88	25.69	21.79	21.31	20.71	11.51	19.07
FCN [20]	9.79	22.70	12.43	25.14	22.72	20.26	20.27	11.73	18.22
TopNet ($N_{\text{out}} = 16384$) [5]	5.85	21.27	10.03	20.09	22.98	14.65	24.25	11.78	16.36
PointNetFCAE ($N_{\text{out}} = 2048$) [25]	5.81	21.14	8.95	22.01	33.36	15.81	27.52	14.09	18.59
PointNetFCAE ($N_{\text{out}} = 16384$) [25]	4.00	16.70	6.24	14.63	18.15	10.99	15.77	8.55	11.88
SA-Net [22]	5.27	14.45	7.78	13.67	13.53	14.22	11.75	8.84	11.22
GRNet ($N_{\text{out}} = 2048$) [17]	7.64	24.06	12.02	24.62	28.73	18.85	32.90	12.48	20.16
GRNet ($N_{\text{out}} = 16384$) [17]	3.79	14.86	6.71	12.74	13.73	11.05	15.43	**6.50**	10.60
Ours ($n_{\text{chart}} = 32, N_{\text{out}} = 2048$)	3.60	14.67	7.03	14.04	20.61	10.66	18.01	7.62	12.03
Ours ($n_{\text{chart}} = 128, N_{\text{out}} = 10240$)	**3.16**	13.53	6.58	13.21	12.93	10.29	14.25	6.98	10.11
Ours ($n_{\text{chart}} = 32, N_{\text{out}} = 16152$)	3.38	**13.00**	**6.12**	**12.72**	**11.87**	**9.18**	**12.43**	7.17	**9.48**

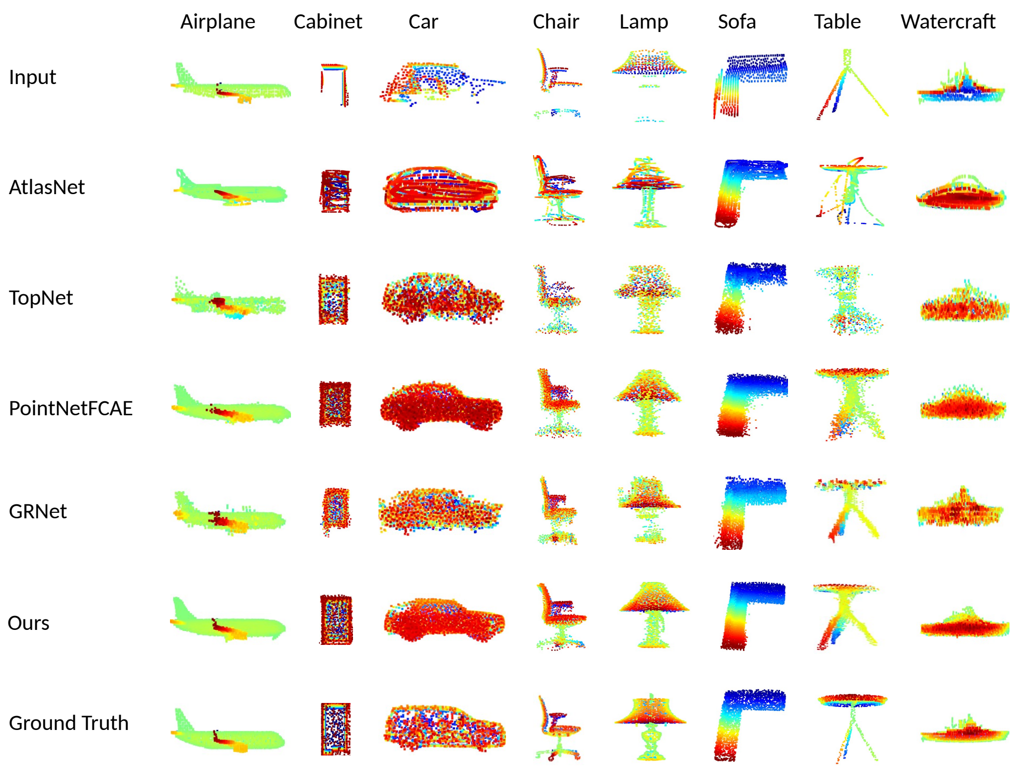
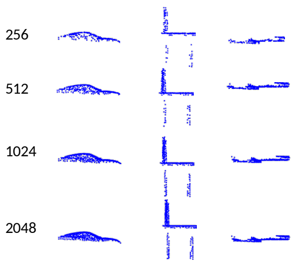


Fig. 3: Visualization of completion results on the Completion3D evaluation set.

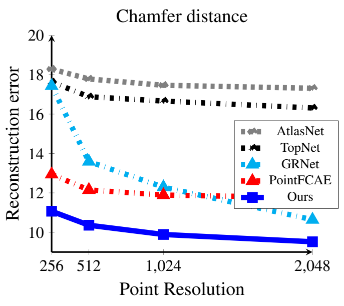
one network. The incomplete point clouds from KITTI have diverse input resolutions and are highly sparse. We use the same strategy as in Section IV-B to lift the number of points to 2048. Besides, we transform the incomplete point cloud by using the 3D bounding boxes to get a point cloud that is distributed between $[-0.5, 0.5]$. The qualitative result illustrated in Fig. 5 indicates that our approach and PointNetFCAE can generate more detailed shape information compared to the other methods.

D. Ablation Studies

In this section, we will study the effectiveness of our designed structure and chosen hyper parameters. All studies are conducted on the Completion3D data set for consistency. Without loss of generality and without special instructions, we set $N_{\text{out}} = 10\,240$ and $n_{\text{chart}} = 32$ in the following



(a)



(b)

Fig. 4: In (a) the point resolution varies from 256, 512, 1024 to 2048. In (b), we compare the proposed approach against other state-of-the-art approaches on the Completion3D benchmarks. Lower values are better.

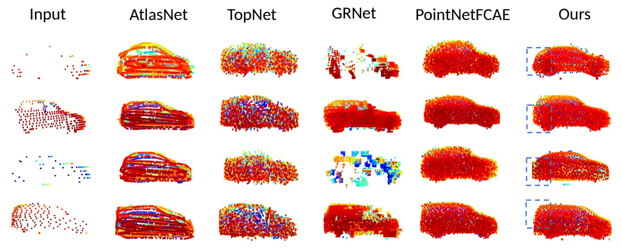


Fig. 5: Qualitative completion results on the LiDAR scans from KITTI. The incomplete input point cloud is extracted and normalized from the scene with its 3D bounding box.

experiment.

*1) **Effect of Transformer encoder:*** The Transformer encoder is the main core used in PCTMA-Net, which has two hyper parameters: the number of encoder layers n_{encoder} and the number of heads h used in the attention mechanism. In this section, we will study the effect on shape completion by varying different combinations of these two parameters. We can conclude from Table II, that we can achieve better shape completion performance with higher numbers of h and n_{encoder}. Taking various factors such as the network parameters into consideration, we set these hyper parameters to $h = 8$ and $n_{\text{encoder}} = 4$.

*2) **Effect of number of charts:*** The hyper parameter n_{char} is used to control the number of charts defined on a shape. In this section, we will study the effectiveness of the number of charts. We summarize the results in Table III. It can be shown, that PCTMA-Net can result in a smaller Chamfer distance with a greater number of charts. However, the parameters of the network will be increased correspondingly, which is shown in the second row of Table III.

*3) **Effect of grid strategy:*** In our proposed morphing-atlas decoder, the pointGenNet maps 2D grids to 3D surfaces. In this section, we will use the plane grid for point generation, which introduces two additional values. We can either randomly sample the value from $[0, 1]$ or use a grid with a predefined grid scale and grid size. The evaluation results on different grid strategies are listed in Table IV. It can be shown, that the mesh grid method shows significantly better performance in comparison to the randomly sampled grid methods. We further study the effectiveness of the grid scale by using the same grid size. The results in Table IV show that the mesh grid scale from 0.05 to 0.5 shares a similar performance.

*4) **Effect of metrics:*** Most existing work employs the Chamfer distance as a loss function due to its efficient computation. The earth mover's distance (EMD) is another option for point clouds and can be formulated as:

$$d_{\text{EMD}}(S_R, S_G) = \frac{1}{|S_G|} \min_{\Phi: S_R \to S_G} \sum_{X \in S_R} ||x - \Phi(x)||_2, \quad (10)$$

where Φ is the bijection function. In this section, we will study the effect on shape completion of different training loss functions. The comparison results in Table V demonstrate, that for a pure EMD loss function, the shape completion value with the metric of CD has the worst performance. The utilization of CD and EMD in the loss function can reduce the Chamfer distance value, and generate a more uniformly distributed point cloud than the pure CD loss function. As EMD uses the bijection function to force the output to have the same density distribution as the ground truth for coping with the linear assignment problem. It hence can generate a point cloud which is more discriminative to local details. However, EMD is much more computationally expensive

TABLE II: The Chamfer distance (CD) on different hyper parameters in the Transformer encoder.

n_{encoder}	2		4		6	
h	4	8	4	8	4	8
CD ($\times 10^4$)	10.86	10.41	10.59	10.21	10.69	10.21

TABLE III: The Chamfer distance (CD) on the number of charts.

n_{chart}	8	32	128
CD ($\times 10^4$)	10.45	10.21	10.11
parameter ($\times 10^6$)	52.75	93.26	258.73

TABLE IV: The Chamfer distance (CD) on different grid types. In Meshgrid (k), k indicates the grid scale.

Grid type	Rand grid	Meshgird (0.5)	Meshgird (0.05)
CD ($\times 10^4$)	11.36	10.25	10.21

TABLE V: The Chamfer distance (CD) on different loss functions.

Loss function	EMD	CD+EMD	CD
CD ($\times 10^4$)	16.12	10.45	10.21

TABLE VI: The Chamfer distance (CD) on different point generators. We abbreviate our Encoder as TE and connect to different algorithm point generators.

Methods	TE-FoldNet	TE-TopNet	TE-AtlasNet
CD ($\times 10^4$)	13.22	13.49	11.36

with approximately $\mathcal{O}(n^2)$, where n is the number of point cloud, compared to CD.

*5) **Effect of point generator:*** In this section, we study the effect of different point generators on shape completion, introduced in FoldNet [7], TopNet [5], by attaching them to our Transformer encoder. The results are summarized in Table VI. All of these three networks have improved to some degree by using the Transformer encoder. FoldNet shows an improvement from 19.07 to 13.22, TopNet improved from 16.36 to 13.49, and the performance of AtlasNet improved from 17.31 to 11.36.

V. Conclusion

We propose a novel network named PCTMA-Net for point cloud completion. Through its encoder-decoder structure, PCTMA-Net can effectively capture features of local regions for predicting missing shape parts. The utilization of the concept of an atlas further helps the network to reconstruct a smooth shape with a predefined number of charts. We conducted extensive experiments on the Completion3D and KITTI data sets to validate our proposed network structure's effectiveness. Via the experiments, we can conclude that our approach outperforms other state-of-the-art approaches on these two large data sets.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] J. Lin, M. Rickert, and A. Knoll, "6D pose estimation for flexible production with small lot sizes based on CAD models using gaussian process implicit surfaces," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[3] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2442–2447.

[4] J. Lin, M. Rickert, and A. Knoll, "Grasp planning for flexible production with small lot sizes based on CAD models using GPIS and bayesian optimization," 2021.

[5] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "TopNet: Structural point cloud decoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry, "AtlasNet: A papier-mâché approach to learning 3D surface generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[11] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point cloud transformer," *arXiv preprint arXiv:2012.09688*, 2020.

[12] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3D object reconstruction from a single depth view with adversarial learning," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 679–688.

[13] T. Hu, Z. Han, A. Shrivastava, and M. Zwicker, "Render4Completion: Synthesizing multi-view depth maps for 3D shape completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 4114–4122.

[14] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3D-encoder-predictor CNNs and shape synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877.

[15] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, "High-resolution shape completion using deep neural networks for global structure and local geometry inference," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 85–93.

[16] D. Stutz and A. Geiger, "Learning 3D shape completion under weak supervision," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1162–1181, 2020.

[17] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "GRNet: Gridding residual network for dense point cloud completion," in *European Conference on Computer Vision*. Springer, 2020, pp. 365–381.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2017, pp. 5105–5114.

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions On Graphics*, vol. 38, no. 5, pp. 1–12, 2019.

[20] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *Proceedings of the International Conference on 3D Vision (3DV)*, 2018.

[21] M. Sarmad, H. J. Lee, and Y. M. Kim, "RL-GAN-Net: A reinforcement learning agent controlled GAN network for real-time point cloud shape completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5898–5907.

[22] X. Wen, T. Li, Z. Han, and Y.-S. Liu, "Point cloud completion by skip-attention network with hierarchical folding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1939–1948.

[23] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "PF-Net: Point fractal network for 3D point cloud completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7662–7670.

[24] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[25] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2018, pp. 40–49.

[26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[27] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," *CoRR*, vol. abs/1512.03012, 2015.

RightsLink

Sign in/Register

PCTMA-Net: Point Cloud Transformer with Morphing Atlas-based Point Generation Network for Dense Point Cloud Completion

Requesting permission to reuse content from an IEEE publication

Conference Proceedings:
2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Author: Jianjie Lin; Markus Rickert; Alexander Perzylo; Alois Knoll

Publisher: IEEE

Date: 27 Sept.-1 Oct. 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW