

TECHNISCHE UNIVERSITÄT MÜNCHEN
TUM School of Computation, Information and Technology

A Hybrid Perception Framework for Analysis and Modeling of Human-Object Manipulations

Juan Carlos Ramírez de la Cruz

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. Gordon Cheng

Prüfer der Dissertation: 1. Prof. Dr. Ing. Darius Burschka
2. Hon.-Prof. Dr. Michael Suppa

Die Dissertation wurde am 23.10.2023 bei der Technischen Universität
München eingereicht und durch die TUM School of Computation,
Information and Technology am 11.07.2024 angenommen.

A mis padres.

A mi padre Carlos Ramírez, gracias por ser como eres.

A mi madre Marlene de la Cruz, la mujer y ser humano más grande que he conocido.

A mis hermanas Nidia y Alejandra, por estar ahí, siempre cuando las necesito.

A mi abuelita Panchita, gracias.

Los amo, y lo saben.

To my parents.

To my father Carlos Ramírez, thank you for being the way you are.

To my mother Marlene de la Cruz, the greatest woman and human being I have ever known.

To my sisters Nidia and Alejandra, you are always there whenever I need you.

To my grandmother Panchita, thank you.

I love you all, and you know that.

Abstract

For an autonomous robotic system to interact or collaborate efficient- and successfully with its immediate environment, it is required to devise the appropriate strategies of cooperation. The level of engagement, either by interaction or understanding, as well as the degree of dexterity and finesse on the task performance and completion depends not only on the system equipment e.g., sensors and actuators, but also on how the acquired information is organized and interpreted, this is, how the environment is perceived. The challenge increases by observing non-static or dynamic environments where the system has to cope not only with noise, inherent sensory misreadings or incomplete data, but also with continuously modified scenarios where the system has to guarantee the stated level of commitment by updating and obtaining a consistent representation of the world and coherent prediction of the expectations.

While the sensing capabilities of current commodity, hybrid sensors allow an increasingly more rapid and reliable acquisition of compound data e.g., spatial and visual information, there still exists the demand in complex scenarios, where multiple (potential) events evolve at different times and parts of the scene, to complete the captured information and extract the most essential and valuable features and attributes, and at the same time, to organize and present it in a more human-understandable manner.

In this work we propose a hybrid perception framework that associates the visual and spatial information coming from the sensors with the functional and operative aspects of the observed scene. For this, the identification of back- and foreground scene areas allows the further labeling and characterization of potential, independent dynamic structures -or actors- inside the scene, where a-priori or gained-by-experience knowledge can help to complete their local- and currently observed properties and features, like 3D shape, appearance, motions, functionalities, affordances, etc; which are to be attached to these back- and foreground entities. As a result we obtain a hybrid representation of the world that augments the captured sensory geometry with functional information and that is able to expand the *prediction horizons* of tracked-actors motions from purely kinematic information to a higher level of discernment, like actions or activities. External and specialized agents of higher perception levels can take advantage of the hybrid framework for a deeper scene analysis and modelling of spatio-temporal events. In the presented work we expose the methods, procedures and principles the proposed framework is built upon and show some examples of practical applications. As a complete integration of the framework an external agent for the analysis of *human-object-background* interactions is implemented and a new dynamic model for object manipulation is presented.

Zusammenfassung

Damit ein autonomes Robotersystem effizient und erfolgreich mit seiner unmittelbaren Umgebung interagieren oder zusammenarbeiten kann, müssen entsprechende Kooperationsstrategien entwickelt werden. Das Maß des *Engagements*, entweder durch Interaktion oder Verständnis, sowie der Grad der Geschicklichkeit und Finesse bei der Durchführung und Fertigstellung von Aufgaben hängt nicht nur von der Systemausrüstung wie beispielsweise Sensoren und Aktuatoren ab, sondern auch davon, wie die gewonnenen Informationen organisiert und interpretiert werden, d.h. wie die Umgebung wahrgenommen wird. Das wird umso komplizierter, wenn nicht-statische oder dynamische Umgebungen untersucht werden, in denen das System nicht nur mit Ungenauigkeiten, inhärenten sensorischen Fehlinterpretationen oder unvollständigen Daten zurechtkommen muss, sondern auch mit sich ständig verändernden Szenarien, in denen das System das angegebene Maß an Engagement gewährleisten muss, indem es eine gleichbleibende Darstellung der Welt und eine stimmige Vorhersage von Erwartungen aktualisiert und bezieht.

Während die Erfassungsmöglichkeiten aktueller handelsüblicher, hybrider Sensoren eine zunehmend schnellere und zuverlässigere Erfassung zusammengesetzter Daten wie räumliche und visuelle Informationen ermöglichen, besteht in komplexen Szenarien, in denen sich mehrere (potenzielle) Ereignisse zu verschiedenen Zeiten und in verschiedenen Teilen der Szene abspielen, nach wie vor der Bedarf, die erfassten Informationen zu vervollständigen und die wichtigsten und wertvollsten Merkmale und Eigenschaften zu extrahieren und gleichzeitig in einer für den Menschen verständlicheren Weise zu organisieren und zu präsentieren.

In dieser Arbeit schlagen wir ein hybrides Framework vor, das die von den Sensoren stammenden visuellen und räumlichen Informationen mit den funktionalen und operativen Aspekten der beobachteten Szene verknüpft. Zu diesem Zweck ermöglicht die Identifizierung von Hinter- und Vordergrundbereichen der Szene die weitere Kennzeichnung und Charakterisierung potenzieller, unabhängiger dynamischer Strukturen oder Akteure innerhalb der Szene, wobei a priori oder durch Erfahrung gewonnene Kenntnisse dabei helfen können, ihre lokalen und aktuell beobachteten Eigenschaften und Merkmale wie 3D-Form, Aussehen, Bewegung, Funktionalität, Leistungen usw. zu vervollständigen, die diesen Hinter- und Vordergrundentitäten zugeordnet werden sollen. Als Ergebnis erhalten wir eine hybride Darstellung der Welt, die die erfasste sensorische Geometrie um funktionale Informationen erweitert und in der Lage ist, die Vorhersagehorizonte der Bewegungen der verfolgten Akteure von rein kinematischen Informationen auf eine höhere Unterscheidungsebene, wie z.B. Aktionen oder Aktivitäten, auszudehnen. Externe und spezialisierte Agenten höherer Wahrnehmungsebenen können die Vorteile des hybriden Rahmens für eine tiefere Analyse der Szene und die Modellierung raum-zeitlicher Ereignisse nutzen. In der vorliegenden Arbeit werden die Methoden, Verfahren und

Prinzipien erläutert, auf denen das vorgeschlagene Framework aufgebaut ist und einige Beispiele für praktische Anwendungen gezeigt. Als vollständige Integration des Frameworks wird ein externer Agent für die Analyse von Interaktionen zwischen Menschen, Objekten und Hintergrund implementiert und ein neues dynamisches Modell für die Objektmanipulation vorgestellt.

Acknowledgments

One of the first things one notices being in Germany for first time is how long or peculiar the compound German words can be by adding (with some grammatical rules) a word right after the other; one of those words was to me *Doktorvater*, whose meaning I was comprehending and getting clearer during this work and years. Prof. Darius Burschka, I do not have the right words to express all my gratitude for all these years, thank you for admitting me to your research team, for your guidance, for your vocation of teaching and above all, for being that accessible and excellent human being. I also want to thank to the *Consejo Nacional de Ciencia y Tecnología* CONACYT and *Deutscher Akademischer Austauschdienst* DAAD for the guidance and support to make this project possible. There are other many persons in particular I had the fortune to meet and that I would like to thank for one or other reason, in chronological order: Oliver Ruepp, Elmar Maier, Chavdar Papazov and Susanne Petsch, thank you all for that first German-international cultural (*shock*) interchange. I would like also to thank to my ex-colleagues in DLR (Deutsches Zentrum für Luft- und Raumfahrt – German Aerospace Center), specially to Dr. Michael Suppa, thank you for having me accepted in your team and gave me the opportunity to collaborate with those world class people and projects; to Alexander Schaub for the work together at the ROMO vision system and to all the guys of the ROMO *Rudel*; to Michael Panzirsch, the best officemate ever. And thank you all guys for the pleasure of having met you: Tin Muskardin, Andre Coelho, Simrad Singh, Sven Schmitd, Georg and Jonseok. To Herr Hao Xing, Andre Costinescu and Peter Gawronski, and of course, Marquitos and Verenita.

“If you wish to make an
apple pie from scratch,
you must first invent the
universe“
— Carl Sagan

Agradecimientos

Una de las primeras cosas que uno observa estando por primera vez en Alemania es lo largas o peculiares que pueden ser las palabras compuestas en Alemán sólo al añadir (con algunas reglas gramaticales) palabra tras palabra; una de esas palabras para mí fué *Doktorvater*, cuyo significado lo fui entendiendo durante este trabajo y años. Prof. Darius Burschka, no tengo las palabras correctas para expresar toda mi gratitud por todos estos años, gracias por admitirme a tu grupo de investigación, por tu guía, por tu vocación de enseñar y sobre todo, por ser ese accesible y excelente ser humano. Quiero también agradecer al *Consejo Nacional de Ciencia y Tecnología* CONACYT y al *Deutscher Akademischer Austauschdienst* DAAD por la guía y el soporte para hacer posible este proyecto. Hay algunas otras personas en particular las cuales tuve la fortuna de conocer y que me gustaría agradecer por una u otra razón, en orden cronológico: Oliver Ruepp Elmar Maier, Chavdar Papazov y Susanne Petsch, gracias por ese primer intercambio (*shock*) cultural Alemán-Internacional. Me gustaría también agradecer a mis ex-colegas del DLR (Deutsches Zentrum für Luft- und Raumfahrt – Agencia Aeroespacial Alemana), especialmente al Dr. Michael Suppa, gracias por haberme aceptado en tu equipo y darme la oportunidad de colaborar con personas y en proyectos de clase mundial; a Alexander Schaub por el trabajo conjunto en el sistema ROMO de visión y a todos los colegas de la ‘*manada*’ ROMO; a Michael Panzirsch, el mejor compañero de oficina. Y gracias a todos ustedes por el placer de haberlos conocido: Tin Muskardin, Andre Coelho, Simrad Singh, Sven Schmitd, Georg Jonseok. Al Señor Hao Xing, Andre Costinescu y Peter Gawronski, y claro a Marquitos y Verenita.

“The absence of evidence
is not the evidence of
absence”
— Carl Sagan

Contents

Dedication	iii
Abstract	v
Zusammenfassung	vii
Acknowledgments	ix
Agradecimientos	xi
Contents	xiii
List of Figures	xvii
List of Tables	xxi
1 Introduction	1
Overview	1
1.1 Motivation	3
1.2 External-Agent Concept	6
1.3 Challenges	7
1.4 Related Work	8
1.4.1 Environment Representation	9
1.4.2 Human-Action Analysis	11
1.5 Contributions	18
1.6 Thesis Outline	19
2 Perception Fundamentals	21
Overview	23
2.1 Stereo Camera Rig	25
2.1.1 Stereo Camera Principle	25
2.1.2 3D Stereo Reconstruction	26
2.2 RGB-Depth Sensors	27
2.2.1 Sensor Data Set	28
3 The Geometric Layer	29
Overview	31
3.1 Structure and Overview	33
3.2 3D Scene Representation	34
3.2.1 Octree Data Structure	34

3.2.2	Storage and Mapping	34
3.2.3	Background Detection	35
3.2.4	Foreground Detection	37
3.3	Location Areas	40
3.4	3D Object Completion	41
3.4.1	Object Fusion	41
3.4.2	Z-buffered Re-projection and Data Association	43
3.4.3	Object Maintenance	45
3.5	Foreground Motion: Non-Static Environments	48
3.5.1	Motion Detection	49
3.5.2	Ego-Motion Estimation	49
3.5.3	Independent Object Estimation	50
3.6	Foreground Actor Tracking: Dynamic Environments	51
3.6.1	Visual Object Tracking	51
3.6.2	Hand Tracking: by Detection and Kalman Filter	52
3.6.3	Hand Tracking: by Skeleton Detection	53
3.6.4	Optical-Flow Estimation	54
4	The Abstraction Layer	59
	Overview	61
4.1	Semantic Structure	63
4.1.1	Knowledge Container, Atlas and Object Representation	64
4.1.2	Prediction Horizons	65
4.2	Dynamic Model for Manipulations	66
4.2.1	Action and Activity	67
4.2.2	Plugin Scheme	68
4.2.3	Fore- and Background Interaction Actors	68
4.2.4	Motion Representation	69
4.2.5	Manipulation Modelling	71
4.3	Dynamic Model for Autonomous Driving	74
4.3.1	Perception System	74
4.3.2	Segmentation and Mapping	74
4.3.3	Autonomous Parking Maneuver	76
4.3.4	Pedestrian Segmentation and Path Prediction	76
5	Results - Geometry Layer	79
	Overview	81
5.1	Sensor Setups	83
5.1.1	Stereo Cameras	83
5.1.2	RGB-D Sensor	83
5.2	Octree-DS Performance	84
5.2.1	Building and Storing Times	85
5.2.2	3D Scene Mapping Test	85
5.3	3D Object Completion	87
5.3.1	Z-Buffered Fusion & Confidence Value	87
5.3.2	Blob State Propagation under Partial Visibility	88
5.4	Visual Estimation of Independent Foreground Motions	90
5.4.1	Ego- & Independent-Motion in Non-Static Scenes	90
5.4.2	3D Object Completion in Non-Static Scenes	91

Summary	93
6 Results - Abstraction Layer	95
Overview	97
6.1 Identification and Analysis of Object Manipulation Actions	99
6.1.1 Identification of Object from Data in the Semantic Structure	99
6.1.2 Identification of Motion from Data in the Dynamic Model	104
Summary	114
7 Conclusions	115
Overview	115
7.1 Summary	117
7.2 Future Works	118
Bibliography	121

List of Figures

1.1	Object-Centric Perspective in Robot Perception Analysis of a Scene	6
1.2	General Hybrid-Model Architecture and supporting Functionalities	8
1.3	Placing of Framework’s main Topics inside the Robotics and Computer Vision Areas	9
2.1	Examples of Stereo Camera Setups: Parallel and Convergent	25
2.2	Stereo Camera Setup	26
2.3	Example of a 3D Reconstructed Image with our Stereo Camera Setup	27
2.4	Kinect’s Three Basic Types of Images	27
2.5	Kinect RGB-Depth Overlapped Image	28
3.1	Structure of the Geometric Layer	33
3.2	Exemplary Images of the Geometry-Layer Data Flow	34
3.3	Octree DS: Geometric and Graph Representation	35
3.4	Octree Computational Complexity for Accessing and Spatial Performance	36
3.5	BFS and DFS Graph Traversing Algorithms	37
3.6	Implementation of a 3D Octree-based Storage and Mapping	37
3.7	Geometrical Detection of Background Regions	38
3.8	3D Clustering at different Octree Resolutions	38
3.9	Examples of PCA Encapsulation of 3D Blobs	38
3.10	Outline of 3D Scene Segmentation Processes	39
3.11	Example of Location Areas in a Cooking Scene	40
3.12	Blob Map: Intersection of two 3D-Blob Sets captured at times (k) and ($k + 1$)	41
3.13	Data Alignment: Data Re-projection for Object Fusion	42
3.14	Data Association: Example of Data Point Matching	43
3.15	2D Z-buffered re-projection Images of two 3D Blobs	44
3.16	The Four cases of Confidence-Value (CV) assignment for Object Maintenance	47
3.17	Exemplary Non-static Scene for Visual Estimation of Independent Foreground Motions	48
3.18	Example of 2D Detection of Independent Motion in Non-Static Scenes	49
3.19	Exemplary Scene for Feature-matched Flows of Inliers and Outliers for Camera Ego- and Object Motion Estimates	51
3.20	OpenCV and YoLo Object Detection Examples	52
3.21	Visual 3D Model-based Object Tracking: Plane Square and BSpline-based Contour	53
3.22	Visual Appearance-based Object Tracking: Particle Filter (PF)	54

3.23	Hand Tracking by Detection	54
3.24	Hand Tracking by Kalman Filter (KF)	55
3.25	OpenPose Key Points and Detection of Body-Skeleton, Fingers and Face.	55
3.26	Augmented OP Skeleton of Hand-Blob Key Points	56
3.27	Examples of OP Fragmented and Failed Detections	56
3.28	Sparse and Dense Optical Flow Examples	56
3.29	Object Tracking Paths and Optical Flow Detection	57
4.1	Semantic Structure of the Abstraction Layer	63
4.2	Object Container and Object Representation in the framework	64
4.3	Exemplary Scenes for Prediction Horizons	66
4.4	Examples of an Object with two different Functionalites (Manipulations) and Two Objects with similar Manipulations	67
4.5	Plugin Scheme for Dynamic Model of Object Manipulations	69
4.6	Detection and Segmentation of Fore- and Background Actors in some Experimental Scenarios	69
4.7	Motion Representation: 3D Path and Rotation Angles	70
4.8	Projection of 3D OF Vectors onto the Plane defined by H_z -axis at Hand Centroid for Estimation of Hand Rotations	71
4.9	3D Geometric Projection of a Point onto a Plane	71
4.10	Hand Centroid Projections onto Interaction Planes	72
4.11	Examples of PCA and PLN Encapsulations	73
4.12	Manipulation Modelling: PCA Encapsulation, Motion and Histograms Profiles of Projected Manipulation Patterns	73
4.13	The ROboMObil - RoMo	75
4.14	360° Stereo Reconstruction and 3D Segmentation around RoMo	76
4.15	RoMo's Autonomous Parking Maneuver	77
4.16	RoMo's interface for visualizations of 3D Pedestrian segmentation, path prediction and 2D mapping	78
5.1	Guppy F-080C Camera	83
5.2	Stereo Camera Images of 3D Object Segmentation Process	83
5.3	Kinect RGB-Depth Overlapped and Warped Images	84
5.4	Exemplary RGB-D Images for Processing and Analysis of Scenes in the Framework	85
5.5	Building and Storing Performance of Implemented Octree DS	85
5.6	Exemplary Mapping Sequence for Characterization of Octree-DS Performance	86
5.7	Building Time and Number of Nodes/Leaves vs Voxel Size.	86
5.8	Scene Sequence for 3D Object Completion and Confidence Value Assignments	87
5.9	ICP Fitting of Fused Blob Points with different Confidence Values (CV) to a 3D Model	89
5.10	Blob State Propagation under Partial Visibility	89
5.11	Pioneer 3-DX	90
5.12	Test Scene for Visual Estimation of Independent Foreground Motions	90
5.13	Confidence Value and 3D Object Completion in Non-Static Scene	92

5.14	Percentage Distribution of cv Fused Points obtained during 3D Object Completion in Non- and Static Scenes	93
6.1	Framework-Augmented Images of Chopping/Writing-Scenes Experiments	101
6.2	Manipulation Pattern and Motion Profiles of ' <i>Chopping with a Knife</i> ' vs ' <i>Writing with a Pen</i> '	102
6.3	Manipulation Pattern and Motion Profiles of ' <i>Eating with a Fork</i> ' vs ' <i>Drinking with a Glass</i> '	103
6.4	Framework-Augmented Images of Eating/Drinking-Scenes Experiments	104
6.5	Framework supporting functions for WB scenes	105
6.6	Manipulation Modeling of Actions on WB: ' <i>writing</i> ' vs ' <i>erasing</i> '	106
6.7	Augmented Framework Images of Screwdriver Experiments	107
6.8	Manipulation Modelling of Actions with a Screwdriver: ' <i>Tightening</i> ' vs ' <i>Poking</i> '	108
6.9	Representation of the Rotational Hand Motion Component during ' <i>tightening with a screwdriver</i> '	110
6.10	Representation of the Rotational Hand Motion Component during ' <i>poking with a screwdriver</i> '	111
6.11	Framework Augmented Images of Hammer-Scenes Experiments	112
6.12	Manipulation Modelling of Actions with a Hammer: ' <i>Hitting vs Pulling</i> '	113
7.1	Modeling and Labelling of Activities	119

List of Tables

1.1	Perception Layers of the Hybrid Framework: Geometry- and Abstraction-Layers	4
5.1	Test Results of Octree-DS Mapping Sequence at different Voxel Sizes .	86
5.2	Confidence-Value Evaluation for 3D Object Completion of a Cereal-Box Blob	88
5.3	Confidence-Value Evaluation for 3D Object Completion of a Pop-Corn Box Blob	88
5.4	Test Results of Pioneer Ego-Motion Estimation in Non-Static Environment	91
5.5	Test Results of Object-Motion Estimation in Non-Static Environment	91
5.6	Confidence-Value Evaluation for 3D Object Completion of a Figurine in Non-Static Environment	93
6.1	Object Identification despite Similar Motion Patterns	99
6.2	Motion (functionality) Identification for each Object in Used	100

Chapter 1

Introduction

Sections

Overview	1
1.1 Motivation	3
1.2 External-Agent Concept	6
1.3 Challenges	7
1.4 Related Work	8
1.4.1 Environment Representation	9
1.4.2 Human-Action Analysis	11
1.5 Contributions	18
1.6 Thesis Outline	19

Overview

Before presenting and describing the fundamental concepts and methods that support our proposed hybrid-model framework, we first present in this introductory chapter the concepts and foundations that motivate and encourage to materialize the presented approach and research into this work.

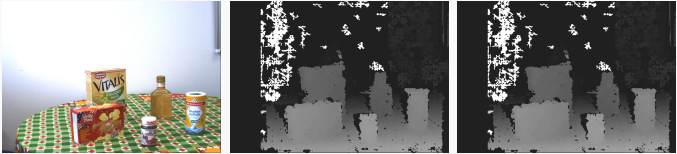
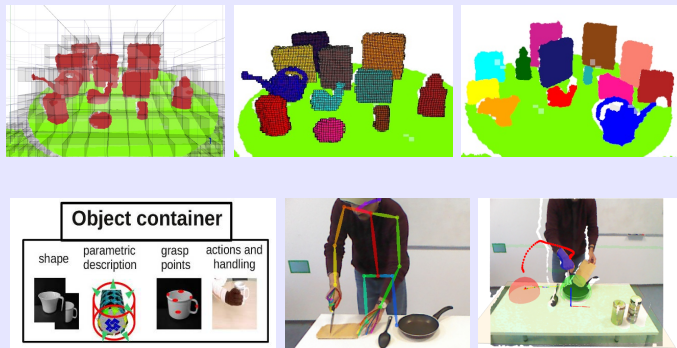
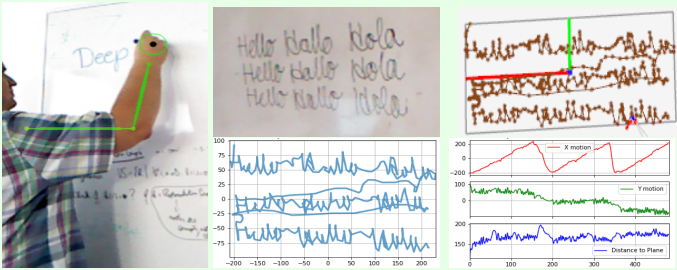
1.1 Motivation

Today's challenges in robotics perception systems demand a huge amount of knowledge comprising numerous and diversified databases ranging from daily-life knowledge to specialized expertise in the fields of science and technology. Although current sensory imaging devices become cheaper with steady increasing robustness allowing a rapid and reliable perception of two- and three-dimensional data from the environment, the sensory information they offer is confined to a paired visual and reconstructed spatial information of the world inside a limited field of view; Moreover, non-static environments or relative motion between the sensors and the structures inside the scope increases the distortions and inconsistencies between the measured and the collated data, demanding a continuous update and refinement for a consistent and coherent world representation. In order to maintain the expected level of engagement either by interaction or understanding at the given task an enhancement of the sensory data must be accomplished. For this, the robotic system is required to re-arrange, complete and associate plausible knowledge to the captured, limited information. Type and amount of the added and attached knowledge indicate the level of perception on the scene.

Biological as well as artificial systems shape the captured information according to their learning or experience, expectations, attention, etc; therefore, they both process and respond differently to the sensory input, which is commonly noisy, incomplete, changing or varying, e.g., we never see or observe fully an object, yet we are able to infer and complete its occluded dimensions, we are also capable to read or decipher an unknown and poorly-understandable handwriting; in audition, we can complete sentences or react to one's own name in noisy crowds. In artificial systems we can consider in first place systems that possess very elemental faculties of perception, i.e., they simply react or are triggered by raw or conditioned sensory data, e.g., simple ambulatory robots like some vacuum cleaners or lawnmowers, only turn randomly to change the path when hitting some physical constraints; similarly, audio or visual devices can react to a fixed level of noise, certain frequencies, or respond to certain colors, intensity of light or motion flow, respectively; Probably the most simple trait of high perception for a robotic mobile system is the ability to navigate around autonomously in a mostly-static environment, this would imply, the capacity to identify physical constraints, avoid them, and eventually correct the path at any possible contact with them; A more perceptive system would detect and classify relevant salient visual features from the environment for self-orientation; In the same manner, systems endowed with even higher level of perception would capture additional functions or attributes from the observed environment, for example, in exploration or grasping systems: the former would search and evaluate specific scene attributes to decide the next steps to execute or the trajectory to follow, the latter would detect and recognize objects and define possible points or areas on them for a stable, proper grasping or handling.

Although in all previously mentioned examples the systems might have captured the same sensory information the level of perception increased respectively; Analogous to biological systems, this could also indicate that perception in artificial systems could be also considered as a subjective aspect since the amount or type of knowledge influence how the environment is perceived.

Table 1.1: Perception layers of the hybrid framework: Geometry- and Abstraction-Layers

Sensor Level: captured raw data is <i>conditioned</i> /adjusted for further processing.	
<p>Input Data visual (2D images), spatial (3D point cloud)</p> <p>Processes signal conditioning: filtering, warping, alignment, etc.</p> <p>Output Data aligned, warped 2/3D images</p>	
Geometric Layer: data is re-arranged, identified and augmented with functional information and attributes	
<p>Input Data calibrated 2/3D images</p> <p>Processes 3D segmentation and clustering, 2/3D object detection/completion, visual tracking, back-and foreground areas and person skeleton detection, etc.</p> <p>Output Data knowledge-augmented geometry: Location areas, segmented 3D object candidates, motion parameters, etc;</p>	
Abstraction Layer: augmented data, recollected and organized in the geometric layer, is utilized and interpreted by external agents of perceptions (plugins)	
<p>Input Data Knowledge containers for motion (traces), locations, physics.</p> <p>Processes Analysis and interpretation of data for modelling, labelling of events, eventually also inference mechanisms</p> <p>Output Data Labeling, recognition, categorization of: actions, activities, gestures, events, behaviour, etc.</p>	

Following the definition of **perception** as the organization, identification and interpretation of sensory data to represent and understand the received information of the environment, we organize the layers of the framework in a hierarchical perception concept of data abstraction and analysis, where besides the two main layers that compound our hybrid framework, the geometric and abstraction layers, an extra stage at sensor level was included to cover and illustrate better the bottom-up data processing in the framework. The **sensor level**, like in the electronics field, is considered the *signal-conditioning* stage where the data is adjusted to fulfill the minimal requirements or basic format to be further processing; as explained later in this work this stage comprises mainly a image calibration, i.e., a *warping* to align the 2D and 3D image data buffers. As also shown in Table 1.1, the **geometric layer** is the processing level where the spatial information is essentially re-ordered, identified and augmented, i.e., the individual, segmented structures are assigned with plausible functional or operational knowledge that is provided by an Atlas, which is a database where a-priori, object-class information of possible scene elements can be stored and acts equivalent to the knowledge gained by learning or experience in humans; processes like 3D segmentation and clustering, 3D object detection and completion, and tracking of motions have been implemented and integrated, yet additional or auxiliary methods or approaches e.g., [85] [117] can also be integrated at this stage; For example, in [85] a mathematical algorithm was implemented for reliable 3D object recognition in clustered arrangements without 3D segmentation is developed, while in [117] alternative and oriented methods for object recognition under scale and perspective variations have been proposed. In Table 1.1 we also show some exemplary images related to some of the processes mentioned for each stage; For example, in the geometric layer: back- and foreground detection, 3D object segmentation and clustering, person-skeleton and hand detection, location areas and the object container which is a data structure used to collect and store attributes and functional information about the objects in the scene; In order to *emphasize* that a higher level of analysis or a deeper *perceptive* processing based on more abstract information can be performed at the top level of our framework, we call this level, the **abstraction layer**; At this highest layer external agents of perception or **plugins** utilize the re-organized and recollected data from the geometric layer, like traces, locations and physics, to analyze and understand a certain, specific event of the many that might be running in parallel inside a dynamic scene. In this work we implemented and present a plugin for the modelling and labelling of physical interactions; For example, in Table 1.1 we present some images corresponding to the action *'writing on white-board'* where motion patterns of the tracked hand are re-projected onto the interacting background (white-board), encapsulated and decomposed for identification and labeling of this particular action. For the sensor-level we show a 2D image, a stereo-reconstructed depth image and a 3D image respectively. Basically the output data of a layer is the input data for the next one. In the following sections we present first the concept of the proposed framework and give an overview of its functional blocks, next we give an overview of the concept for the external agent.

1.2 External-Agent Concept

In our *plugin* approach we relate the human capacity of recognizing actions to three significant factors: **what** object is in use, **how** it is handled or manipulated, and **where** the action takes place; As shown in Fig. 1.1 these factors are linked directly to the object; **what** stores two main object attributes: physics and functionality; the first attribute is related with the spatial and appearance properties as well as other abstract parameters like weight, center of mass, grasping points, etc; the second refers to the primary functionality or functionalities the object offers; **how** considers the dynamic aspects the object undergoes like motion parameters, path, trajectory, etc; in **where** we consider the *affordance* of the environment, this is, all those relevant areas inside the scene that are related with the object's functions or states.

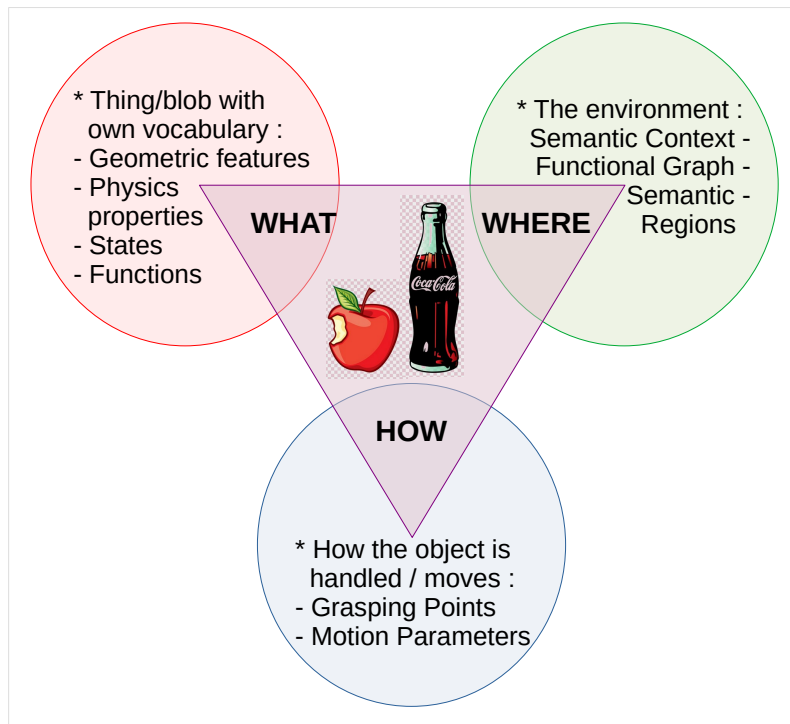


Figure 1.1: An object-centric perspective in robot perception analysis of a scene. The *what*-factor is related with the object attributes; the *where* is related to those specific places or location areas the object have contact with inside the scene; the *how* is related to the motions the object undergo during its employment.

We consider the relevance or weight of these factors decreases correspondingly; for example, we can deduce the action of '*cooking*' not necessarily inside a kitchen but also outdoors e.g., in a camp or on the street, we can also infer if a person is '*reading*' or '*typing*' in the public transport. Although the recognition of the object employed could lead directly to the labeling of the action since objects are designed to fulfill a particular task or function, many of them possess more than one functionality, or they could be not properly manipulated or be used with other purposes.

Our plugin utilizes the captured knowledge in the framework and boosts the traces and prediction horizons of motions from the geometric level representation to the level of actions or tasks. In Sec. 4.2 we present a new method to characterize and model human-object actions based on the observation of the manipulation motions

and the interaction background upon which the action takes place; Our implementation allow us the labeling of actions like '*hitting vs. pulling*', '*picking vs. tightening*', that are distinguished when using the same object, hammer or screwdriver, respectively; furthermore, we are also able to differentiate actions with similar motion patterns and that are performed with different objects like '*eating vs. drinking*' or '*chopping vs. writing*'.

Extended or modified versions of this plugin can be applied for example in human-robot collaboration areas to infer the task in progress and assist in its execution, or in surveillance systems to supervise the proper use of objects, or simply to learn new objects functionalities or *improvise* objects by transferring known functions to geometrically similar objects.

1.3 Challenges

The current, ongoing development in artificial scene perception and understanding requires not only a modeling of the static scene structure but also requires additional knowledge, i.e., some scene facts to be maintained in the world model, like motion parameters, physical properties and functionalities of potential objects or structures. It is then necessary to further enhance the information delivered by visual sensors beyond the appearance or geometry level in order to perceive and understand the world based on a more functional, operative knowledge, for example, by incorporating in hierarchical manner additional layers that abstract, organize this supplementary information: a layered representation of the environment that couples the pure geometric 3D representation of the world to the abstract knowledge about the tentative object structures, *blobs*, in the scene, this is, the sensory information is augmented and acts in the abstraction layer as a set of geometric tokens linked to their attached functions and attributes that, in turn, will support the completion and execution of more complex abstract tasks; This knowledge will represent known, a-priori, task-relevant information or facts of object candidates like mass, handling properties and grasping points being examples in a case of manipulation tasks. The coupling of abstract knowledge to the geometry in a layered scheme of the map helps to ensure consistency of the representation. From an implementation point of view it must be required to identify back- and foreground areas, isolate foreground shapes that represent possible objects, identify and complete their properties and track their motions, as well as to identify the affordances or locations areas in the background.

In Fig. 1.2 it is shown a general, functional structure of a devised layered framework with the fundamental data blocks in blue and with some supplemental and exemplary blocks in green. The **Geometry** block represents the captured sensory data that is split in back- and foreground areas for the segmentation and identification of 3D shapes, **Blobs**, representing places of affordances or location areas in the background and tentative object candidates in foreground; the attributes and functions of the detected blobs are estimated locally, and if required, completed by the *Atlas*; the gained knowledge is recollected and organized in the **Knowledge Containers**: physics, locations and motions. As shown in the scheme this knowledge can be supplied and accessed by additional processes like semantic mapping, trajectory analysis, navigation, etc; that might expand the functionality of the framework. Similarly, external agents of higher level of perception or understanding, *plugins*, can be also coupled

to build models or mechanism of inference for labeling, categorization or recognition of events, actions, gestures, etc. In Chap.3 and Chap.4 these layers are described in more detail.

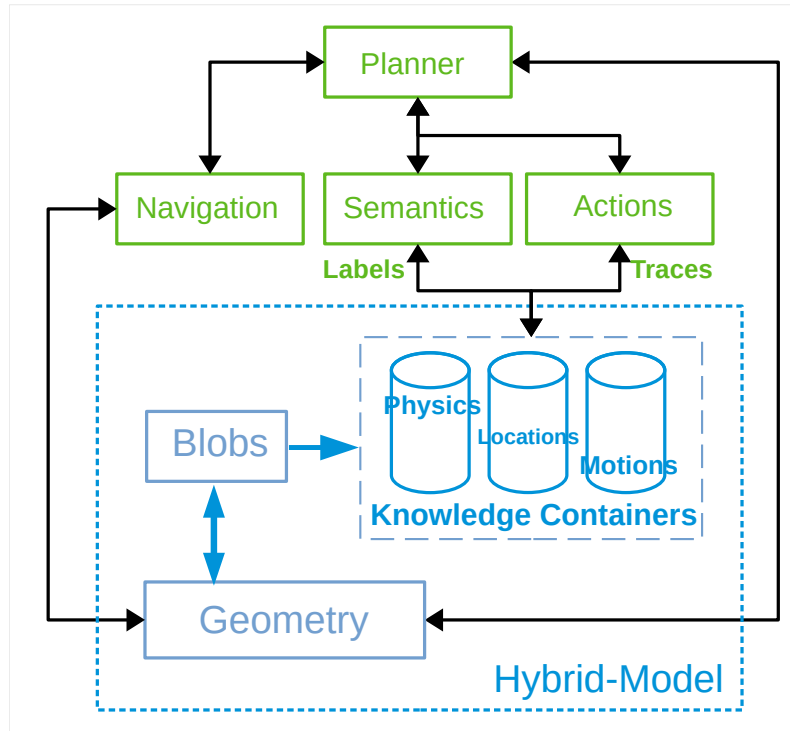


Figure 1.2: General hybrid-model architecture and supporting functionalities. The core elements of the hybrid model (in blue) couples the geometry data coming from the sensors with a higher level of information, the knowledge containers, which in turn can support alternative processing methods of perception (in green).

1.4 Related Work

Regarding the previously described features and functionalities of the framework and its layers, in this section we present some approaches and works from the robotics and computer vision fields that we consider are either pioneer, relevant, or closely related with the methods and implementations in this work; since these topics could widely range from camera calibration processes to prediction-horizons-related approaches we present this compilation based on two main subjects: *i) environment representation*, which basically corresponds to the work implemented in the geometric layer of the framework and would involve topics like 2/3D mapping, object detection/recognition, scene segmentation, semantic mapping, etc; and *ii) human action understanding*, namely, the analysis of Human-Object Interactions (HOI), which is the main topic of our external agent of perception. In Fig.1.3 it can be observed how we arrange and site these topics among the huge amount of proposed and related approaches in these fields.

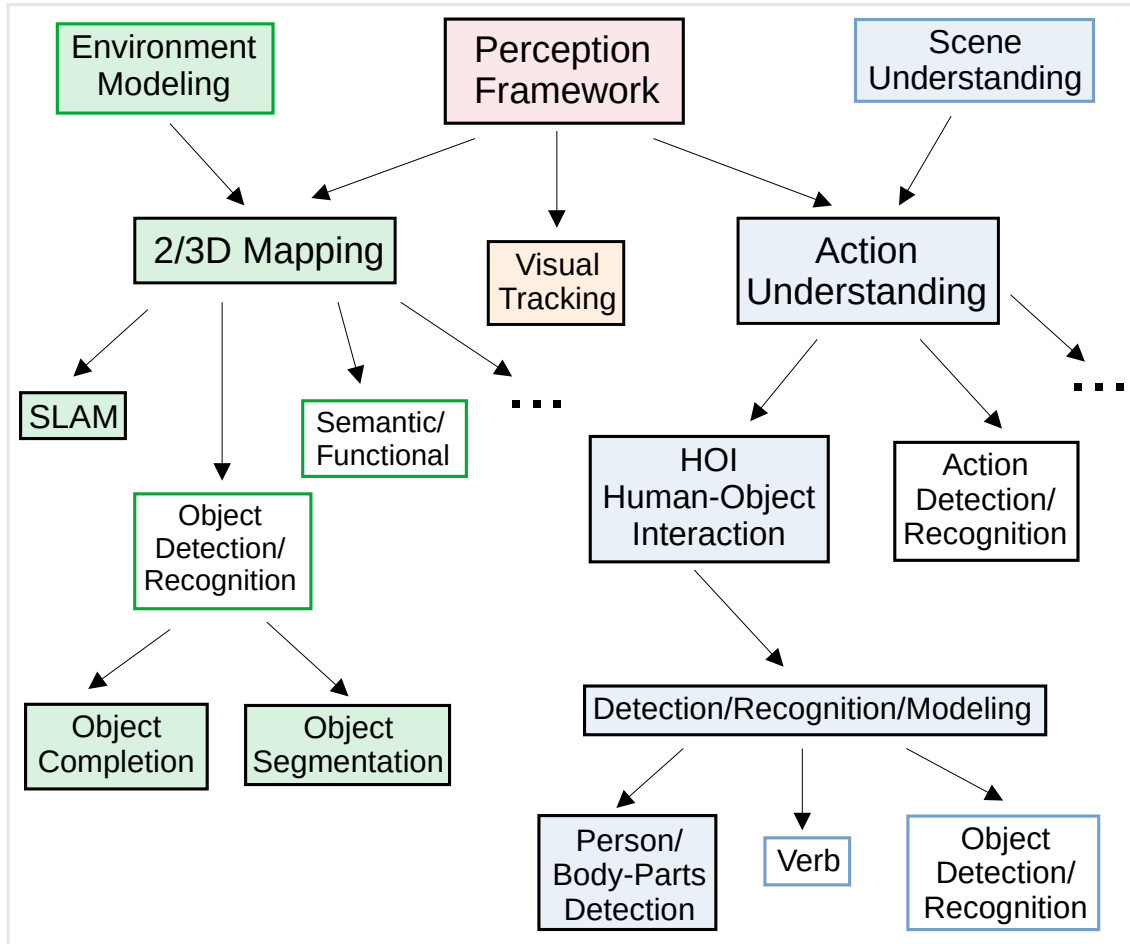


Figure 1.3: Placement of framework’s main topics inside the robotics and computer vision fields. We can derive our work on the geometric layer fundamentally from the subject of *Environment Modelling*, in which we perform a 3D scene mapping where auxiliaries topics like SLAM, functional mapping, object detection/completion, etc; are also treated; likewise, the implementations presented in the abstract layer, namely our plugin (Sec.1.2), can be related to *Scene Understanding*; as described further in this section this is a huge field of study that comprises several research branches that give origin to a huge amount of approaches as well. *Visual Tracking* is a entire research area on its own and it is almost an ubiquitous tool in computer vision nowadays; in our framework visual tracking is also an important auxiliary tool (Sec.3.6.1).

1.4.1 Environment Representation

Sparse Mapping, SLAM approaches. The surroundings representation of a robotic agent depends, in large part, on the main objective of the application or planned robot-scene interactions, and it is also influenced, to some extent, by the type of data the sensor devices supply. We can find, for example, in robot navigation works based on Visual Odometry (VO) approaches [79][98][35] the representation of the environment is based mainly on sparse data points, where the principal objective is to build a consistent path of the sensor’s ego-motion subject to the matching and alignment of salient visual features (like SIFT, BRISK, SURF [70][66][5]) between consecutive image frames. VO approaches have been widely utilized to support wheel

odometry readings on planetary exploration rovers [44][17][124][73] where the environments are predominantly static. Moreover, VO approaches have been integrated and have become part of more complex and global mapping schemes like in Simultaneous Localization And Mapping (SLAM) [26][3], in particular Visual-SLAM (V-SLAM) [56][62]; Unlike the local and incremental estimation of the sensor paths in VO methods SLAM approaches consist of a constant estimation and refinement of sensor poses and the building and maintenance of a consistent world map, this is, sensor pose and environment map are correlated, refined and updated in a global context. A huge amount of SLAM variations have been proposed covering as well a huge variety of exploration applications like in rovers, Unmanned Aerial Vehicles (UAV)'s, underwater vehicles, humanoids, medical applications, pipe inspections, etc; some other representative SLAM derivations are monocular SLAM (MonoSLAM) [19][21], stereo SLAM [20][112][68], fastSLAM [76][78], graph SLAM [39] and more recently RGB-D SLAM [51][28]. For a more dense spatial representation Occupancy Grids approaches [27] divide the environment into cell or voxel regions and define for each cell a probabilistic estimate of its occupancy state: occupied/empty. VO and SLAM methods constitute in our framework auxiliary processing components at the geometry layer that support our 3D object completion and fusion with the estimation of sensor ego- and independent-object motions.

Dense Mapping approaches. In our work we aim to observe the world as a set of 3D independent, in-foreground shapes embedded in rigid 3D background structures, therefore, we need to be able to augment and split a rigid environment representation to discern and incorporate tentative, independent and active elements. Mapping models based on octrees [75], k-dimensional trees (**kd-trees**) [6] or spatial kd-trees (**skd-trees**) [80][81], offer an efficient way to re-arrange and re-group volumetric data based on certain data characteristics like proximity of neighbor data points. Although kd-trees are a valuable data structure to evenly distribute multidimensional point data and get efficient access to it we employ in our framework an octree as a base data structure since it allows to *tessellate* the observed environment, and to distinguish and define potential 3D foreground structures by the clustering of connected components through its leaves (as explained in Chap. 3); in a further processing stage we could apply and benefit of (spatial) kd-tree features for a balance storage and fast access to the data elements of segmented structures. 3D environment modeling based on octrees is widely applied in the area of robotics and computer graphics; their main feature is that they recursively *discretize* the stored data into voxels, allowing a hierarchical segmentation and representation of the environment, with this, we can determine at what level of resolution our world representation will be; this also permits a compression of data and memory saving (at cost of lower resolution); With octrees we do not need to know beforehand the size of the area to be mapped since they can be gradually extended; they also allow a local, confined map regions to be updated by isolating the data only in the corresponding voxels. A representative example of octree-based 3D-modelling tool is OctoMap [50] which is a c++ open source library implementation of a 3D probabilistic occupancy grid mapping. In this work we coded our own octree implementation.

1.4.2 Human-Action Analysis

In the field of robotics and computer vision an interesting, important and ongoing topic of research is *scene understanding*; this area is very attractive for such research communities since an accurate evaluation and judgment of the environment and actions inside it by artificial agents will allow to perceive and interact with their surroundings with high levels of reliability and engagement. This is an immense area of study, where, an equally immense amount of approaches have been developed, and new ones are being continuously proposed. Scene understanding can exclusively comprise the labeling of indoor and outdoor scenes or environments based generally on the detection, recognition and at times also the spatial order of the visual elements they contain; output labels can be like: bedroom, library, forest, office, landscapes, etc; it can also comprise, as shown in Fig.1.3, the *understanding* of human actions individually or in group, where by understanding we imply the detection/recognition or modelling of such actions or activities; based on the current approaches we can find that detection/recognition can be generally carried out on still images since it mainly consists on identifying the object in use, a human pose or body part and the evaluation of the semantic or geometric relation between them, whereas the modelling is accomplished on footages; Recognition might tell us *what* an action is observed, while modelling might inform additionally *how* it is performed. In our plugin (Sec.4.2) we focus on the analysis of *Human-Object Interactions* (HOI); we consider this area, shown Fig.1.3, as a division inside Action Understanding and whose principal characteristic is that it must be mandatory an object involved in the action. In our work an action, or task, always implies the use of an object, otherwise we refer to an *activity*, this is explained in more detail later in Sec.4.2.1, along with other related terms; other approaches might go a bit further and differentiate between action and task, where examples of the former are like 'closing the door', 'take a glass', 'open the fridge', etc; on the other hand, tasks imply a more steady grasping and manipulation of the object rather than a simple, short contact with the object like just pushing, pulling or shifting it. Another term to highlight in this field is *event*; one of its definitions can be found as "something that happens or is regarded as happening; an occurrence, especially one of some importance"; according to this and in this context, we can understand it as an unexpected *behaviour* or a *motion* that suddenly comes up and breaks or differs from the motion patterns observed so far, [87]: they do not exhibit temporal or spatial repetition. The relevance of such event depends on the purpose of the approach, such as it can vary from detections of *irregularities* like small, short body gestures or incidents, like a '*person's fall/stumble*', '*stealing an object in a shop*' or as simple as '*start extending the arm to grasp an object*' [123][58][54][42]; its importance can be also given in a global context where the focus is not only on one single person nor the motion of his/her body parts but on all, or most of, the visual elements (persons, objects, animals, buildings, surroundings, etc;) and the relations between them; in this case the main event might be marked by a set of different actions or activities executed simultaneously by different actors that all together, as a whole, describe the same single situation: event [116][67][114]; e.g., watching a group of sitting persons in a room one could tell whether it is a '*classroom*' or an '*office meeting*' by closely observing furniture, arrangement, clothes, poses, etc.

Human-Object Interactions (HOI). As mentioned above, we consider HOI a research area part of *Action Understanding*. As shown in Fig.1.3, we distinguish three

fundamental components that can help us to describe in general an action in this area: *i*) the person or body parts that execute the action and are in contact with the object, *ii*) the object in use and *iii*) a verb, which is eventually the searched component that describes the action. Detection/recognition approaches, mainly in still images, require basically the first two elements to infer the third one; out of these components other elements can be derived or new features can be extracted, like in modelling approaches, in which an additional element, the *motion*, can substitute or complement the object, and vice versa. In our HOI modelling plugin three elements are relevant: *i*) the principal *doer* of the action: the hand, *ii*) the motion: hand's trajectory, and *iii*) the scene background, namely, the supporting plane upon which the action is executed. We can find that our approach shares and combines close similarities with two other research streams in this field: 1st-person/ego action recognition and gesture recognition; in the former the hand(s) also plays an important role, they focus on the manipulation of objects by evaluating hand-related aspects like grasping, motion parameters, hand-object pose, etc: for the latter approaches, gestures are basically short, ordered hand motions, therefore, hand trajectories are the most important feature they focus on. Similarly, in our approach we search for the fundamental and characteristic traits of hand motion trajectories that belong and describe the basic manipulation of the object; these traits have to be discriminated from other attributes also conveyed in the motion, like noise, speed, personal manipulation techniques, etc. In the rest of this section we describe briefly how some selected and related approaches in these areas represent and model actions and interactions with objects.

In [40] the authors present a Bayesian framework to integrate different perceptual aspects like analysis of human movements or detection of manipulable objects, that are involved in the HOI understanding in video sequences or static images, and apply spatial and functional constraints on them for coherent semantic interpretations; although there is no dynamic in static images they utilize contextual information instead. In their research they also found that the functional capabilities or properties of objects can be derived from shape [94][109] and physics [25]. Similar to our analysis, they also consider that two objects similar in appearance can also be recognized by their functionality i.e., how people interact with them, e.g., in the case the objects are not visible at all, the analysis of the persons' *pantomime* can help to identify them; in the same way, two identical poses or motions could be also differentiated by the context. In their approach three classes of human movements are identified: *i*) reach movements: enable object localization, *ii*) grasping: ignored, since they are too subtle to be perceived in their approach and *iii*) manipulations: provide contextual information about the type of objects being acted on. Additionally, they also utilize the object reactions, i.e., the result, effect or consequence of the manipulation, as contextual information, e.g., the pouring liquid into a cup, or the light bulb on/off as result of pressing a button. In [33] they are concerned to recognize *primitive* HOIs; they call primitive interactions to actions on objects that basically consist on a primitive body motion, like '*reaching and grasping a cup*'. They claim that motion information alone may not be sufficient to achieve higher-level reasoning about activities that involve interactions with objects; the authors introduce the concept of actor-object state, which combines the information about the interaction dynamics, actor-object static appearances and spatial configurations, and base their analysis on the fact that the motion and appearance of the actor as

well as the actor-object configuration are constrained by the object. The interaction scenarios they dealt with are the *'grasping'*, *'touching'* or *'pushing'* different objects, like forks, spoons, cups or small toys. For a given image, the authors in [69] try to identify triples of the form: $\langle \text{human}, \text{verb}, \text{object} \rangle$. Their framework consists of a human-object detection followed by an interaction prediction stage; a scoring scheme based on spatial, context and motion encoders, and a fusion module that combines the accumulated knowledge predicts the final score for each detected candidate human-object pair and outputs the plausible triple. In their work [30] the authors consider that conventional methods treat the HOI problem at a holistic body or a very coarse-part level (torso, legs, head, etc;). However, according to their research [9] [95], attention is non-uniform and humans tend to focus on different body parts according to different contexts, therefore, different body parts should be paid with different attention levels, and correlations between different body parts should be also considered. They proposed a pairwise body-part attention model to learn to focus on the crucial parts; their goal is to evaluate the probabilities of certain interactions on a predefined list of HOIs; their non-uniform model can discover the most informative body parts for recognition, they also focus on the correlations between multiple body parts through joint correlations between each pair of body parts. In other approach [127] the authors acknowledge that HOI consists of basically two different problems, detecting: *i)* an object that is commonly small or partially visible, and *ii)* self-occluded body parts; however, objects and body poses can serve as mutual context to each other, i.e., recognizing one facilitates the recognition of the other. In this regard, they propose a mutual context model to jointly represent objects and human poses; object detection provides a prior for better human pose estimation, while human-pose estimation improves accuracy detecting the objects. A 4D modelling of HOI is presented in [119]; such model is defined as the combination of 3D mutual space of human pose and object, plus the 1D temporal dimension of event transitions. The authors present a hierarchical graph model of events: a major event is decomposed in sub-sets of smaller, atomic events, which in turn are decomposed into human poses, object(s), and their geometric relations; for the temporal component, they evaluate the transition probability between consecutive atomic events; in their approach, the eight events under study are of the type of *'fetching water from dispenser'*, *'drink with a mug'* or *'press the button'*. The work presented in [37] is an approach to modelling and recognizing temporal structures of visual activities; these structures can be extracted by representing an activity as a trajectory of an observation vector, which besides positions and displacements can include additional features like object's salient points, its color distribution, 3D pose, etc; they present a probabilistic framework to recognize gestures as spatial-temporal structures in state space, where covariance in the measurements are also taken into consideration and state predictions are simply previous states plus arbitrary Gaussian noise. In [125] the authors tackle the problem of recognizing human motion in real time; they present a method for fast-recognizing 3D motion trajectories by parsing them into segments of sequential primitives: straight line, plane arc, left/right hand helix, etc; then each primitive is labelled and represented by an integral variant descriptor. In the motion recognition stage the primitives are matched by a proposed hierarchical dynamic time warping (H-DTW) algorithm. In the real-time recognition stage, as soon as a new primitive is complete, it is matched with the candidate trajectories learned for every motion class; actions under study

comprise whole body activities as well as upper- and lower-extremities motion, like *'jumping in place'*, *'jumping jacks'*, *'throwing'*, *'waving/clapping hands'*, *'sit down'*, *'stand up'*, etc. In [74] they extract visual motion trajectories of gestures and actions and model them in a probabilistic framework as a series of visual events based on the image moments (area, centroid, and elongation); they restrict the study to four gestures for visually mediated interaction to control a camera: *'pointing left/right'* and *'waving high-up/low-down'*. Instead of splitting a whole trajectory into primitives they divide them into three main events: initial transitional phase, middle phase, and a final transitional phase. Observed and modelled events are matched by estimating the likelihood of the observed trajectory given the model in a Gauss density function; a corresponding gesture is outputted by the matching of its events sequentially. The approach in [11] exploits the global spatio-temporal information about where and when 2D interest points are detected. They represent actions as clouds of interest points observed at different temporal scales. For this interest-point detection they propose a two-step image differencing detector based on the Gabor filter [22]; they build clouds of these interest points by accumulating previously detected points along the consecutive image frames and distinguish two set of features: *i)* 1st set is related with the shape and speed of the foreground object: height-width ratio and its speed, *ii)* 2nd set is a vector of features of the interest points: height-width ration of the cloud, speed of the cloud, density, amount of overlap between two clouds. Their representation is able to capture smooth motions, robust to view changes and occlusions. The authors of [115] propose to characterize human actions with an *actionlet* ensemble model, where an *actionlet* represents the interaction of a subset of human joints. They start from the observation that human motion is articulated by nature, therefore, extracting it from video is a difficult task and present novel features to represent human actions in depth data: *i)* Local Occupancy Pattern (LOP): describes the depth appearance in the neighborhood of a 3D joint and is also able to capture relations between human body parts and the objects the person interacts with; *ii)* Fourier Temporal Pyramid (FTP) that represents the temporal structure of an action, *iii)* *actionlet ensemble model*, where an actionlet ensemble is a linear combination of actionlets that represent an action, and the model represents a particular action that may only involve a small subset of joints. In this way, they take into consideration the relative position of a joint with respect to the other, and with the LOP they compute the local occupancy information based on the 3D point cloud around each 3D joint; through the FTP they recursively split the action and apply the short Fourier transform [82] to the previously estimated 3D joints and LOPs, and take its low frequencies coefficients as features. They analyzed actions like: *'high arm wave'*, *'horizontal arm wave'*, *'hand catch'*, *'press button'*, *'pour water from kettle'*, *'fetch water from dispenser'* *'use mouse'*, *'make a call'*, etc. In their paper [111] the authors try to demonstrate that *'specific'* human actions can be detected from *'single frame postures'* in a video sequence. They are able to recognize human actions by detecting the image of a person's posture corresponding to a particular key frame that corresponds to certain action; they apply a transformation of matching points between a key frame of the action and the frames of the action to evaluate; this transformation corresponds to an image (edge/silhouette) deformation like pure translation, similarity, affine, etc; that defines a relation between the images (edges); pure translation yielded the best results. In [83] the authors build a framework of pose- and feature-based action recognition methods to both classify

fine-grained human actions and locate manipulated objects. They represent pose trajectories as sequences of meaningful '*dynamic instants* and *intervals*'. A *dynamic instant* is a single point in a trajectory that represents an important change in motion characteristics, like direction or acceleration, and occurs at salient moments of motion and object interaction ('moving the arm', 'picking up', 'touching', etc;). An *interval* is a trajectory segment between two instants. They augmented the work in [92] by introducing the concept of *visual instant*, which is a dynamic instant plus a set of (visual) features extracted from the frames and pixels around such instant: these features contain spatio-temporal interest-points (STIP) [22] descriptors that lie inside a given spatial and temporal radius. In their approach [91][92] the authors present a representation of human action to capture '*dramatic changes*' using spatio-temporal curvature of 2D trajectories. These dramatic changes are changes in speed and direction of the trajectory; they also introduce the concept of action units called: dynamic instants, and intervals. Based on this, they proposed a mathematical model based on how humans perceive motion by start and stop instants emerging from any type of acceleration: continuous (impulse or step function) or discontinuous (ramp or smooth function). For the matching of actions, they compare only actions with the same/equal number of instants and since the analysis of trajectories is performed in 2D from observing 3D trajectories, they also developed a view-invariant model based on the 'instants' and their 'signs' (defined by direction of turns in the action: clockwise is positive). The authors in [122] utilize histograms of 3D joints locations (HOJ3D) as a compact representation of human postures. The histogram representation of postures is based on 12 body joints and employ a spherical coordinate system for the body-joint location binning, whose center is the hip-center joint; the sphere is partitioned in 84 bins and the 3D joints are cast into these spatial histogram bins through a vote scheme based on Gaussian weights. As a result of the accumulated votes the observed or given posture is represented by a n-bin histogram, a HOJ3D. They collect a large collection of these histograms and vectors of dominant features are extracted from them through the Linear Discriminant Analysis (LDA); these vectors are finally clustered to create a K -word vocabulary where each posture is individually identify. The temporal evolution of those visual words are modeled by the discrete Hidden Markov Model (HMM). The type of actions they analyze are like: '*walk*', '*stand up*', '*sit down*', '*pick up*', '*carry*', '*throw*', '*push/pull*', '*wave/clap hands*'.

The following works are related to 1st-person/ego HOI recognition, which also focus on the hand-object motion trajectories; as described below, they also present additional features that can be integrated in the to the framework (Fig.3.25c), e.g., a single action can be split into a set of smaller sub-actions by observing the evolution of the grasping over the object. In their approach for egocentric recognition of interactions [113] the authors define actions as *verbs*: '*pour*', and interactions as *verb + nouns*: '*pour juice*'; they try to overcome some of the limitations in HOI works like the lack of semantic meaning about the actions of the subject since they obtain the sole knowledge of the subject's pose, or the lack of environmental understanding since they focus on capturing hand motions without recovering the object pose; therefore, the authors propose an approach to predict simultaneously 3D hand and object poses, object classes and action categories from a single image, with the ultimate goal to construct comprehensive interpretations of egocentric scenes to understand human activities. For hand-object pose estimations they

jointly parametrized the control points of the articulated grasping-hand skeleton together with 20 more points corresponding to the object’s bounding box (corners, midpoints edges and centroid); given hand-object poses, actions and object classes a set of probabilities are trained and stored. The trained-mapping learns the explicit dependencies between hand and hand-poses and interactions. Their model takes as input a sequence of frames and outputs for each frame 3D hand object poses predictions along with the estimates of object and action categories for the entire sequence. The work in [38] presents techniques that extract an *understanding* of objects from the *understanding* of hands. The authors demonstrate that what the hand is doing reveals information about the state of the object, how to interact with it and where the interactions occur, and apply all these to the two aspects of interactive object understanding: *i)* learning state-sensitive features and *ii)* inferring object affordances. For the first point they exploit the appearance and motion of the hand as it interacts with the object to derive *supervision* for the object state; for the second point a context prediction task is designed: they mask out the hand and train a model to predict the locations and grasp-types for the surrounding context. They use the hand and object-of-interaction detector from [101]; with these detections, object, hand and hand motions are stacked together to learn the state sensitive feature space, and by adding hand-grasp labels they train to learn regions of interactions and the grasps afforded by those regions. The egocentric approach for RGB-D manipulations of handheld objects presented in [97] takes into consideration hands’ contact points and forces for parsing object manipulations/interactions from a functional perspective. The work is related with *fine-grained* grasp classification: similar hand kinematic poses can have different functions and the authors also differentiate and classify these situations by analyzing contact regions of the hand and the force vectors over the objects. A taxonomy of a 3D hand model for 18 grasps showing the contact and forces for each grasp is also presented. The type of analyzed interactions are like: *’opening a lid’*, *’writing’*, *’flat hand cupping’*, *’holding chopsticks’*, *’trigger press’*, *’can/jar open’*, *’ball hold’*, *’wiping’*, etc. In their approach [36] the authors also present the benefits of using hand poses as a cue for action recognition. They also study the use of 3D hand poses to recognize 1st-person dynamic hand actions interacting with objects. They show that one object can have multiple grasps associated depending on the action performed on it and one (type of) grasp can have multiple actions associated (e.g., *lateral grasp* at *’sprinkle’* and *’clean glasses’*). They obtain, however, hand pose annotations via 6 magnetic sensors (fingertips + wrist) attached to the back of the hand and a 3D full hand pose is inferred using inverse kinematics over a defined 21-joint hand model. In their hand-action taxonomy they involve different actions/verbs, like: *’unfold’*, *’open’*, *’use’*, *’wash’*, *’flip’*, etc; with different objects: *’milk’*, *’juice’*, *’mug’*, *’spoon’*, *’spray’*, *’pen’*, etc; to give actions like: *’put sugar’*, *’open wallet’*, *’open soda’*, etc. In order to capture the connections among different *heterogeneous features* the authors in [52] propose a joint heterogeneous feature learning model for RGB-D activity recognition. Heterogeneous features are those same attributes but that are extracted from different channels, e.g., HOG -Histogram Of Gradients- features extracted from RGB and depth channels. They learn a set of subspaces (one subspace for each heterogeneous feature type) such that features with different dimensionality can be compared and their shared and specific components can be encoded; for this they also introduce a linear projection matrix for each type of feature, to control the dimensionality

of each subspace. Three channels are utilized: dynamic skeleton features, dynamic color pattern and dynamic depth pattern; From each of them the temporal pyramid Fourier (TPF) [115] features from the RGB and depth sequences are extracted. The type of actions under study are like: *'play guitar'*, *'cheer up'*, *'use laptop'*, *'sit still'*, *'brushing teeth'*, *'relaxing on couch'*, *'cooking(stirring)'*, *'cooking(chopping)'*, etc. In their paper [126] the authors confirm that the type of grasp of human over objects give away their intentions of action. The grasp type provides crucial information about human action: *i)* grasp type contains fine-grain information about human action, *ii)* the grasp contains information about the action itself and it can be used for prediction or as a feature for recognition and *iii)* it also contains info about the beginning and end of action segments. Two applications utilizing this information are presented: *i)* inference of human action intention and *ii)* fine level manipulation action segmentation. Their system takes an image, patches around the hand and outputs the type of grasp is used; the types of grasp they consider are of: power (cylindrical, spherical and hook), and precision (pinch, tripod, lumbrical), and closely related with this classification three human action intentions are considered: force oriented, skill oriented and casual. The hypothesis in [12] is that is necessary to model the grasp types and the attributes of manipulated objects in order to accurately recognize manipulation actions, they both contain complementary information for characterizing different actions. *Grasp type* helps to describe the functionality of an action, they are a discrete set of canonical hand poses, nine grasp types in total, two main types: power and precision, which are divided in prismatic, round and flat. *Object attribute* characterizes physical properties of the objects such as rigidity or shape; it also indicates possible hand motion in hand-motion interaction e.g., the body part of a bottle indicates a motion of 'holding', while its cap with small and round shape indicates a motion of 'screwing'. The authors identify three different shape classes: prismatic, round, flat, and additionally deformable.

1.5 Contributions

In this work we present a hybrid framework for scene analysis that contributes to robot perception systems in dynamic scenarios. We conceptualize this proposed framework as hybrid since it divides and organizes an observed scene in two layers of different knowledge domains, i.e., a geometric layer composed of re-ordered and identified, spatial data coming from the sensor devices and an operative layer of functional and dynamic attributes of objects and actors. In this way, the geometric components of the scene are *augmented* at the operative layer by *indexing* each relevant geometric structure to its functional attributes. The framework provides with the required components and procedures to keep and maintain this connection, e.g., the *knowledge containers*, which store relevant information that is current and locally recollected in the scene, as well as the *Atlas*, which store a-priori or gained-by-experience information about general attributes and features of objects.

The field of scene robot perception and understanding is a huge subject of research involving multidisciplinary aspects and knowledge of human daily life that eventually might not be required all of them at once in a single application. In this context, the hybrid scheme also gives the framework the capacity of being extended or customized by external data-processing agents or clients that might require all or some of the collected knowledge the framework offers. Along this work we present different applications in which, depending on the application purposes, the framework is demanded to work on the geometric layer like in cases of 3D object segmentation and reconstruction, navigation or mapping, or rather to work on the augmented layer like in the analysis of functions and motions of foreground elements. The contributions and benefits of the presented work are listed in the following points:

- the sensory data, as a rigid geometric representation of the world composed by a set of 2D images and 3D point cloud data, is analyzed and broken down into a set of independent structures (foreground) embedded in a fixed geometric layout (background). With this arrangement each detected foreground body represents a tentative object candidate that can not only be analyzed and updated individually but also serves as a local, solid-index token linked to corresponding knowledge (functionalities and attributes) about the object itself.
- purely geometric information corresponding to dynamic objects is augmented with functionalities and attributes at the operative layer of the framework which in turn facilitates a consistent prediction of the world since geometric and dynamic state of objects can be analyzed, updated and refined individually: assumptions on 3D shape completion of partial observed objects (due to (self-)occlusions, sensor misreadings, etc.) can be validated as newly spatial, textural and dynamic information is analyzed and collated.
- knowledge at the operative layer is anchored and indexed to the geometry through the segmented bodies in the scene; the structure of the framework provides with the functional blocks to expand and update knowledge: the *Atlas* is the database block that contains general, a-priori or gained-by-experienced information of the world and can be used to hypothesize about object states of the scene, e.g., functions, shape and behavior of objects; the *knowledge*

containers, instead, recollect local, current and specific information about the observed scene, e.g., object completion, traces or paths of dynamic objects.

- prediction horizons i.e., the scope or extent of state predictions, related to dynamic actors in the world described as paths or trajectories with a few seconds of time projection at the geometric layer are levered and interpreted at the operative or functional level as task or action descriptions with more extended prediction windows of multiple seconds.
- a framework with the capacity to supply external and specialized agents of perception with the required spatial and operative information: a *plugin* agent for analysis of human-object interactions is implemented and a new model for description of object motion manipulation is proposed. Unlike other motion analysis approaches that take into consideration transportation movements in their analysis, our proposed motion model focuses exclusively on the *location/functional areas* (those certain places inside a scene where an object fulfills its main function) and decomposes the manipulation movements into a set of motion components that have a direct connection with the function of the object and its immediate background.

1.6 Thesis Outline

This work has been organized as follows. Chap.1 contains the introductory chapter of the thesis; in this chapter we present the reasons and purposes that encouraged the research and implementation of this work, we introduce the general scheme of our approach and also present the research of some representative approaches in the robotics and computer vision communities that we consider are related with the topics and implementations presented in our work, and in Chap.2 we briefly introduce the perception fundamentals of the sensor devices utilized in this work.

We dedicate Chap.3 and Chap.4 to describe the structure and core components of the main framework layers as well as the methods and analyses that are carried out at each. In Chap.3, The *Geometric Layer*, we explain its functional blocks and describe the methods and implemented tools with which the captured, spatial information is analyzed and re-organized; we also present our approach to fuse 3D data from different camera poses, which is utilized for our 3D object reconstruction and completion process; In Chap. 4 we describe our highest layer in the framework, the *Abstraction Layer*, where the re-ordered and recollected data at the geometric stage is processed and analyzed in a more abstract level in order to obtain patterns or models that help to understand or characterize certain scene events or phenomena; we also propose the implementation of a *plugin* i.e., an external agent of perception, and present a new motion model for the analysis and description of object manipulation motions; Results, evaluations, analyses and images corresponding to the tests, implementations or methods of each main layer are found in Chap.5 and Chap.6 respectively.

To conclude this work, in Chap.7 we give some final comments and remarks on the presented work and present some additional ideas for this work to be continued and extended with potential research directions and implementations.

Chapter 2

Perception Fundamentals

Sections

Overview	23
2.1 Stereo Camera Rig	25
2.1.1 Stereo Camera Principle	25
2.1.2 3D Stereo Reconstruction	26
2.2 RGB-Depth Sensors	27
2.2.1 Sensor Data Set	28

Overview

Commonly in data processing applications it is not longer relevant to know where the information come from or how it was obtained since it is in general understood that the data already passed through a *pre-processing* stage that makes it reliable within certain margins. These reliability or uncertainty margins are obtained by the identification of the processes (measurement devices in our case) the data came from; the characterization of sensor devices allow us to know how precise (statistical error) or how accurate (systematic error) are the readings they supply. Nowadays, imagery data of hybrid sensors (e.g., RGB-Depth) offers not only spatial information but compound two- and three-dimensional data that can be acquired, accessed and linked in several ways for different devices. Comparable to a *signal conditioning* phase in electronics, the sensor layer of the framework is the stage in which data pre-processing is performed to condition and understand the data we are working with; in computer vision intrinsic and extrinsic camera calibration belong to this stage. In this chapter we briefly review the measurement principles of the sensors we employed and describe the pre-processing procedures we performed, where we mainly focus on the adjustment or correction between the raw two- and three-dimensional sensory data.

2.1 Stereo Camera Rig

With a stereo camera setup the 3D location of points in space can be estimated. Typical stereo camera arrangements are attempts to emulate human -animal- binocular vision to get a 3D perception of the environment. A common stereo camera setup consists in mounting side-by-side two *twin* cameras, i.e., cameras with identical internal parameters (focal length, pixel dimensions, image size, etc.), separated (horizontally) by a fixed distance, **baseline**, and with image planes kept parallel, fixed in the same plane. Design variations or adaptations to specific purposes can also be found as shown in Fig.2.1. In general the left image is considered the reference image.

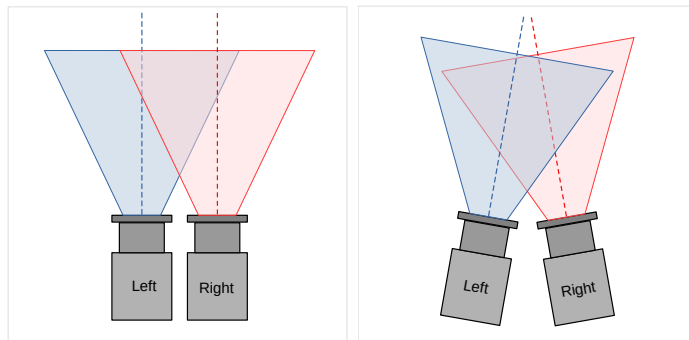


Figure 2.1: Examples of two stereo camera setups. The **parallel** setup (left) is the most common stereo camera arrangement that emulates the animal stereo vision when looking objects in a long distance, while the **convergent** setup (right) can be also considered as the emulation of the natural eye rotation to find or focus on a specific object inside the scope.

2.1.1 Stereo Camera Principle

The physical principle of a stereo camera setup is to obtain a **parallax** effect on the objects inside the overlapped visual field of the cameras, this is, to obtain a measurable shift or difference in the position of a point that is observed from two different points of view; This optical effect is known in computer vision as **disparity** and can be quantified through **depth-from-triangulation** techniques to determine the amount of shift or displacement (in pixels) between the left and right images of an object; the disparity is inversely related to the distance of the object normal to the the image plane since it becomes larger for closer objects and smaller for distant ones, (see Fig.2.2):

$$z = \frac{b \cdot f}{d}, d = dl - dr \quad (2.1)$$

where z is the distance of the object, b is the baseline, f the focal length of the cameras, and dl, dr are the object image on the left and right images. Sparse or dense stereo matching methods are applied to obtain disparity values of only matching salient features (sparse approach), or over entire images in which a higher template-similarity value of each left-image pixel is searched and estimated on each right-image pixel inside a given disparity range $[d_{min}, d_{max}]$.

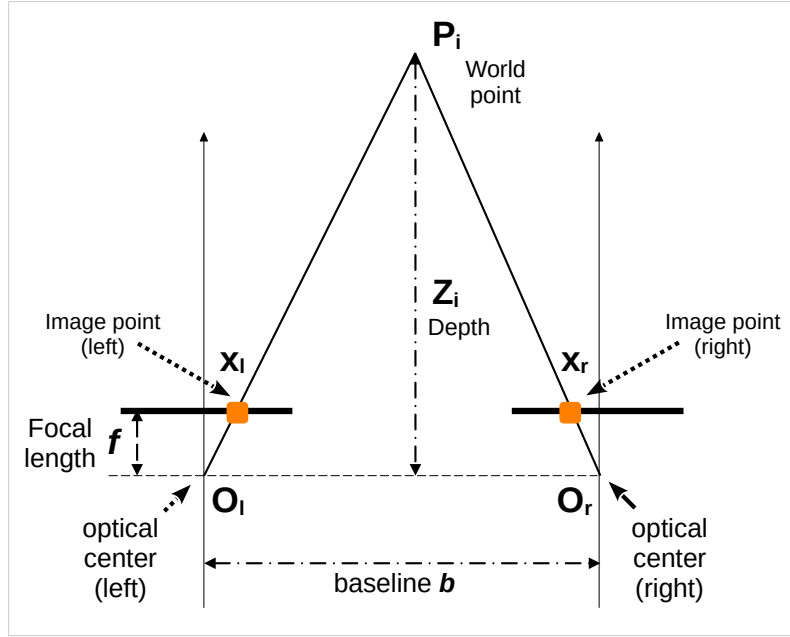


Figure 2.2: Typical stereo camera setup of non-verged cameras to estimate the depth of an observed point in the 3D space as a function of the image disparity.

2.1.2 3D Stereo Reconstruction

In most robotics and computer vision applications it is required to count with a 3D model of the world rather than a disparity image; this 3D representation can be obtained out of the estimated disparity values and the stereo camera setup. Ideally, the position of a point $P_i = (X_i, Y_i, Z_i)$ in 3D space is related to a pixel point $p_i = (u_i, v_i, d_i)$ in disparity space:

$$\begin{aligned} Z_i &= \frac{b \cdot f}{\rho \cdot ({}^L u_i - {}^R u_i)} = \frac{b \cdot f}{\rho \cdot d_i} \\ X_i &= \frac{b \cdot ({}^L u_i - u_0)}{d_i} \\ Y_i &= \frac{b \cdot ({}^L v_i - v_0)}{d_i} \end{aligned} \quad (2.2)$$

which gives, as before, the distance of the point in space now in function of the pixel width ρ , and (u_0, v_0) is the pixel coordinate of the center of the image. In practice, the estimation of stereo reconstructed points contains also uncertainty components in each dimension, where the Z -component dominates over the other uncertainty dimensions:

$$\sigma_z = \frac{b \cdot f}{d^2} \cdot \sigma_d = \frac{Z^2}{b \cdot f} \cdot \sigma_d \quad (2.3)$$

here the disparity d is considered a random variable with normal distribution and standard deviation σ_d that is propagated to the uncertainty of the object depth estimate σ_z ; this uncertainty increases with the distance squared Z^2 of the image plane to the object. Fig.2.3 shows an example of a 3D reconstructed image using our stereo camera setup.

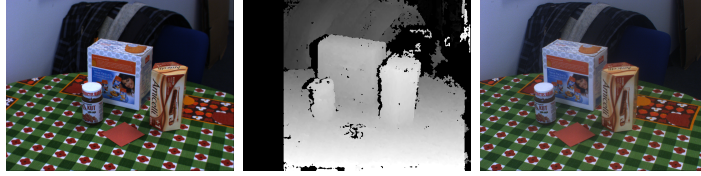


Figure 2.3: Example of a 3D stereo reconstructed image. At the sides are the left and right images of the stereo cameras, resp; and in the middle their reconstructed image; the brightness of each pixel is proportional to the proximity of such point to the sensors, the brighter, the closer; black areas indicate the lack of depth information.

2.2 RGB-Depth Sensors

Besides the stereo camera setup we also employ an RGB-D sensor, namely, the Kinect-1®. Essentially it consists of three components: an infrared (IR) laser, an IR and an RGB camera. The function of the IR components is to supply a *textureless* depth image that is obtained through stereo triangulation of an IR point pattern emitted by the laser and detected by the IR camera. As shown in Fig.2.4 Kinect-1 supplies basically three types of images, an RGB, a Depth and an IR image, from which the first two are the most commonly used in user-end applications either separated or compound, the last image is rather used for calibration purposes of the IR camera. Although the RGB and IR images are the same size (480 x 640 pixel) we can also observe that the sizes and positions of the captured objects differ inside each image; this is basically due to (AO) the difference of focal lengths, normally being in the RGB shorter ($\sim 2.9mm$) than in the IR camera ($\sim 6.1mm$).

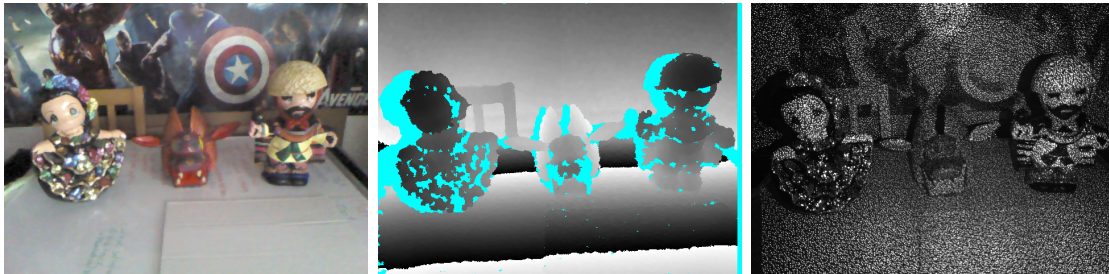


Figure 2.4: Kinect's three basic types of images: (left) RGB, (middle) Depth and (right) infrared (IR) images. The cyan-colored areas in the Depth image correspond to points with no valid depth values; the IR image was taken without blocking the IR projector so that the IR projected points can be observed as those brighter dots scattered throughout the image. Note how the sizes (hence also positions) of the objects differ from RGB to IR-Depth images; the objects appear to be more distant, and hence smaller in the RGB image; note the gap from the objects to the image edges.

This image difference between the RGB and IR-Depth can be better exposed if we overlap the RGB and Depth images as shown in Fig.2.5; however, to work with proper 3D RGB-textured images an RGB-IR camera correction is required. This adjustment can consist in an extrinsic calibration to obtain a full transformation matrix (rotation and translation) that relates the two cameras in space, or it can also consist in an image warping procedure. Since the Kinect cameras are close to each other ($\sim 2.5cm$ baseline) and assembled in a common stereo setup we opt to warp the

images in our framework. The main objective in this procedure is to create a *distortion* in one image in such a way that fits *as much as possible* the other one; Two outcomes from this procedure are possible: the warped (distorted) IR is projected onto the RGB image or vice versa; In either case, this distortion is an image transformation that basically shrinks the Depth or expands the RGB data to fit the image counterpart; we proceed as follows.

Using the RGB as base image we warp the Depth image:

$$\begin{aligned} u_{rgb} &= i_{rgb} - C_{rgb,x} \\ v_{rgb} &= j_{rgb} - C_{rgb,y} \end{aligned} \quad (2.4)$$

the resulting pixel coordinates $\langle u, v \rangle_{rgb}$ are now expressed w.r.t. the image center; the corresponding Depth pixel values are then found:

$$\begin{aligned} u_{depth} &= (u_{rgb} \cdot \frac{f_{ir,x}}{f_{rgb,x}} + C_{ir,x}) \\ v_{depth} &= (v_{rgb} \cdot \frac{f_{ir,y}}{f_{rgb,y}} + C_{ir,y}) \end{aligned} \quad (2.5)$$

where the parameters $[f_x, f_y]_{rgb}$, $[f_x, f_y]_{ir}$ of camera focal lengths and $[C_x, C_y]_{rgb}$, $[C_x, C_y]_{ir}$ of camera centers are obtained through the RGB and IR intrinsic camera calibration.

Respectively, for the RGB warping:

$$\begin{aligned} u_{Depth} &= i_{Depth} - C_{ir,x}, & u_{rgb} &= (u_{Depth} \cdot \frac{f_{rgb,x}}{f_{ir,x}} + C_{rgb,x}) \\ v_{Depth} &= j_{Depth} - C_{ir,y}, & v_{rgb} &= (v_{Depth} \cdot \frac{f_{rgb,y}}{f_{ir,y}} + C_{rgb,y}) \end{aligned} \quad (2.6)$$

The wrapped corrections can be seen in Fig.5.3 of Sec.5.1.2.

2.2.1 Sensor Data Set

The resulting working data in our approach consists of 2/3D-image sets $\{I_{2D}, I_{3D}\}_k$, where $I_{2D}(k) \equiv [u_i, v_i]_k$ is the set of image pixels, $I_{3D}(k) \equiv \{p_i, P_i\}_k$ is a set of 3D reconstructed points p_i along with their spatial uncertainty P_i , and k is the observation or capturing time-stamp.

A huge amount of information about stereo cameras and RGB-D sensors regarding identification, precision, accuracy, calibration, etc; is ubiquitous nowadays. Here we only addressed the particular topics and issues that we considered are directly related with our work and that are meaningful to be referred to at this stage. Some of our consulted references regarding stereo calibration are [89] [60] [96], regarding kinect [106] [128] [45] [59] [77]. In addition to the own captured image frames and footages the framework was also adapted to work with external footage databases.



Figure 2.5: Kinect RGB-Depth overlapped image. Basically due to the difference in focal lengths ($f_{IR} > f_{RGB}$) the RGB and Depth image data do not fit: the cyan-colored areas of non-valid IR-depth values do not correspond to the object shapes of the RGB image.

Chapter 3

The Geometric Layer

Sections

Overview	31
3.1 Structure and Overview	33
3.2 3D Scene Representation	34
3.2.1 Octree Data Structure	34
3.2.2 Storage and Mapping	34
3.2.3 Background Detection	35
3.2.4 Foreground Detection	37
3.3 Location Areas	40
3.4 3D Object Completion	41
3.4.1 Object Fusion	41
3.4.2 Z-buffered Re-projection and Data Association	43
3.4.3 Object Maintenance	45
3.5 Foreground Motion: Non-Static Environments	48
3.5.1 Motion Detection	49
3.5.2 Ego-Motion Estimation	49
3.5.3 Independent Object Estimation	50
3.6 Foreground Actor Tracking: Dynamic Environments	51
3.6.1 Visual Object Tracking	51
3.6.2 Hand Tracking: by Detection and Kalman Filter	52
3.6.3 Hand Tracking: by Skeleton Detection	53
3.6.4 Optical-Flow Estimation	54

Overview

Our actual world consists primarily of environments imposing physical constraints through geometric structures, where each of them has its own meaning either as a context-related function specific to the current scene or as a general object attribute. While the awareness and consistent registration of these geometrical constraints can help supporting simple perception processes like localization, navigation, mapping, grasping, etc; the acknowledgment of functions and attributes of these surrounding objects or structures can help to understand the environment in a higher level of perception. The geometric layer is the first stage to extract and convert simple pixels and 3D points to a more concise and concrete information towards a more human semantic or familiar daily vocabulary; In this layer we decide what and how this new information, embedded in the imagery data, is extracted; The goal of this layer is to implement and apply the appropriated mechanisms and elements to observe, register and organize the captured data from a geometric perspective. In this chapter we focus on the geometric modelling of fore- and background entities inside scenes. First we present the 3D segmentation of captured scene data to obtain a set of independent structures or **blobs**; This segmented data is further processed and analyzed (clustered, associated and fused) for modelling and completion of 3D objects, where we introduce a Z-buffered re-projection method as a way to get more consistent structures and filter out noisy information. On the background side we also detect and confine spatially the functional areas i.e., those regions within background structures with possible semantic meanings, (specially on supporting planes); this contributes to the understanding of actions and activities as well as current states or functions of objects. Since our registration approach does not store data as one rigid model but as a set of independent point clusters (foreground) embedded in a 3D point cloud of supporting structures (background) we are able to update independently not only the *chassis* of single foreground objects but also their poses, which in turn allows us to cope with dynamic changes in the world. From this geometric level, we also describe the mechanisms we applied to observe and register these dynamic changes and motions; this is a valuable information that will also help us to infer or support possible, general attributes or current functionalities of objects in a higher level of understanding.

3.1 Structure and Overview

The presented framework is an effort to model the actual world by introducing a layered representation of the environment that couples, as solid tokens, the pure geometric 3D representation of the world to the knowledge of their functional or operative attributes. An overview of this layer can be seen in Fig. 3.1. The scheme shows the data flow and the relations between the principal components and procedures that are explained in more detail later in this chapter.

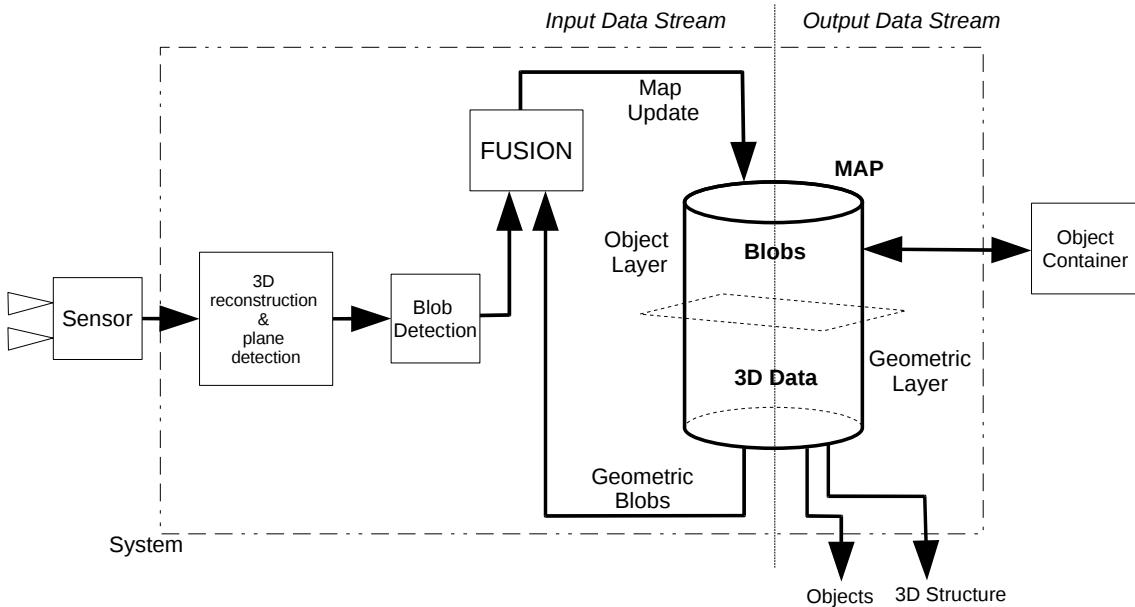


Figure 3.1: Structure of the geometric layer. The input data stream coming from the sensor layer is firstly spatially split to obtain a set of tentative object candidates (blobs) that are independently updated as more data is available.

From a stereo camera rig we obtain a range image of the observed scene (Figs. 3.2a, 3.2b), after determining the supporting background plane, we build a first rough geometric representation of the scene by means of an octree (Fig. 3.2c). A 3D segmentation procedure on the octree leaves (Sec.3.2) is performed in order to cluster the groups of 3D points that are spatially connected through the leaves. The output of this procedure yields a set of 3D blobs, (tentative objects) see Fig. 3.2d, that represents geometrical tokens linked to the higher level of the framework. The data association and fusion is executed **blob-wise** between two consecutive 3D segmented scenes by applying a **Z-buffered** re-projection method as explained in more detail in Sec. 3.4.1. The resulting map model at this stage is a spatial representation of the world that comprises a 3D background structure holding a set of potential object candidates and whose 3D structures are to be completed and refined over time as more captured data is collated. The **object container** is our data structure representing and linking these segmented tokens to the higher-perceptive layer, where additional object functions and attributes can be assigned and stored.

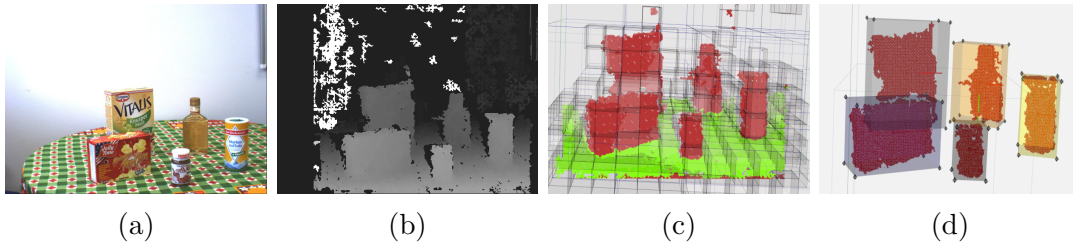


Figure 3.2: Exemplary images of the main data flow processes at the geometry layer. a) 2D left-camera image, b) range image of the scene, c) octree scene representation with potential foreground (red) and background (green) structures and d) set of 3D clustered and encapsulated blobs.

3.2 3D Scene Representation

The rigid 3D information supplied by the imagery sensor still constitutes a simple group of captured points without any particular assumption about their spatial connections or relations among them. The main goal of the scene segmentation is to obtain a first spatial notion of the captured 3D data; The result of this process is the spatial identification of the back- and foreground structures along with a set of independent and encapsulated 3D blobs that represent tentative objects in the scene. The role of an octree, our storage and mapping component, as well as the required procedures employed at this stage are following described.

3.2.1 Octree Data Structure

An octree is a computational tree-based data structure (DS) primarily used in 3D graphics and game engines; at the present it is commonly utilized in computer vision and robotics applications. As illustrated in Fig. 3.3 octrees allow to spatially organize and access three-dimensional data in a hierarchical manner by partitioning recursively the space it encases in eight smaller and equally-sized cubes (or rectangles) known as **octants**, **voxels** or **oct-nodes**. This spatial subdivision continues until the leaves, fixed at a maximum tree depth d_{max} , are reached. The leaf, *a.k.a.* oct-leaf, is the smallest and *un-splittable* voxel where data is stored, and its size determines the resolution of the octree. We can consider, for example in mapping applications, data-occupied leaves' connections form a **downsampled** 3D tessellated model of the world as the data resolution is reduced from a 3D point to a voxel representation. This downsampled representation is the finest and closest 3D spatial model of the world that an octree can achieve.

3.2.2 Storage and Mapping

An octree has a *linearithmic* spatial complexity $\mathcal{O}(k \log N)$, where k is the number of points and N is the number of nodes (Fig.3.4a), and as a tree-based data structure the octree has a running time of $\mathcal{O}(\log N)$, which is faster than a linear running time but gets a bit slower as the number of nodes increases (Fig.3.4b). Accessing oct-leaves for data storing or retrieving is achieved by traversing the octree according to the **Depth-First Search** (DFS) algorithm [18], see Fig.3.5a. As its name indicates this algorithm focuses on traversing a DS deeper at each search step as opposite to the

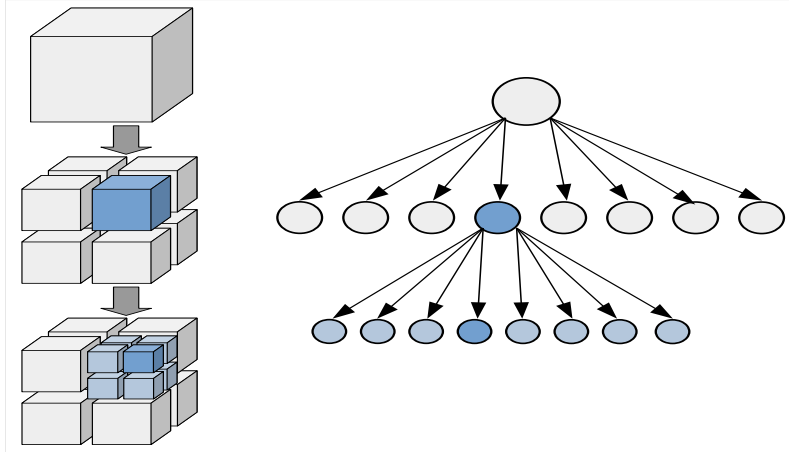


Figure 3.3: Geometric and graph representation of an octree DS. From the octree root, largest node (top cube), each *oct-node* divides recursively into eight cubes of equal size (octants or voxels), until the leaves, the smallest voxels where data is stored, are reached. In this example $d_{max} = 2$.

Breadth-First Search (Fig.3.5b) that explores first thoroughly along the spread of the structure before going one step deeper; DFS is the common option to reach the tree leaves that lie at the deepest level in the tree structure. BFS is also later employed in 2D segmentation procedures. Mapping approaches based on octree DS allow a simplified and flexible occupancy model of the world. The Octomap library [50] is a representative mapping work implementation based on this DS that offers several storage features and algorithms to obtain and maintain an updatable and flexible 3D model of arbitrary environments. A 3D scene mapped from several camera poses can be observed in Fig.3.6a along with some snapshots of our octree-based storage implementation for this scene.

3.2.3 Background Detection

Opposite to object structures, and from a geometric point of view, background regions are in general large, flat and predominantly static areas; but like foreground objects, they can also have different purposes or functionalities in the world, like walls, shells, doors, desks, tables, etc. Due to the limited sensory scope and the partial and scattered occlusions of foreground structures background areas are in general captured as broken or incomplete surfaces. To detect such background regions we apply either of two procedures: the first is based on a pure **RAN**dom **SAM**ple **Consensus** (RANSAC) procedure [34] over the whole scene, the second method is by picking three reliable image pixel-points (x, y, z) that belong to the background plane model, according to Eq. 3.1:

$$ax + by + cz + d = 0 \quad (3.1)$$

for the normal-point form equation of a plane a, b and c are the constants that define the normal vector $\hat{\mathbf{n}} = [a, b, c]$, and d is the distance from the origin of the coordinate system to a reference point on the plane $d = -\hat{\mathbf{n}} \cdot \mathbf{p}$.

For the first method we can assume the background regions cover larger areas than the foreground areas in the observed scene, therefore we can expect large parts of

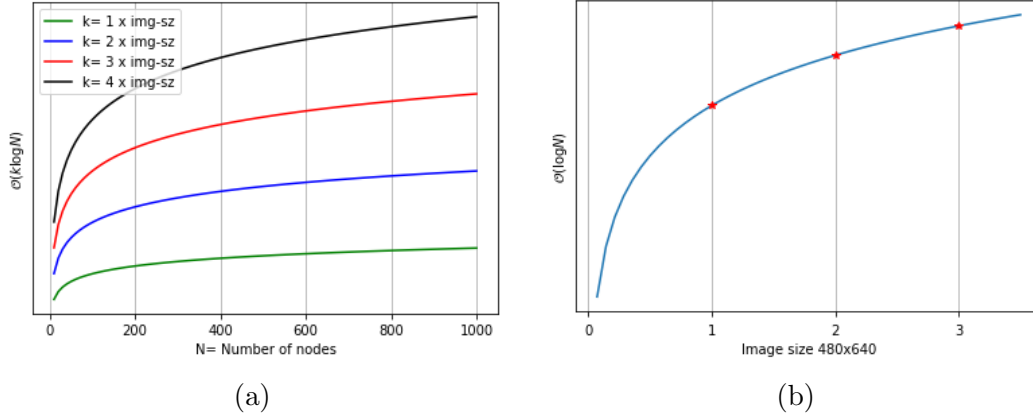


Figure 3.4: Octree computational complexity. a) Spatial, building time based on the number of nodes N for different image sizes k , b) running, accessing time based on number of images (data size), with an image size of 640x480 pixel-pts

scene 3D points belong to these structure planes; following [43] we choose a tentative percentage of outliers ϵ , i.e., points that do not belong to the current searching plane, to define the first number of iterations N (Eq. 3.2) over the next steps: a set of three 3D-points is randomly selected to define a normal-vector candidate of a temporal plane model; through a voting scheme we determine the number of inliers supporting this plane model, i.e., how many scene points lie or fit to this plane based on a threshold distance from the plane to each tested point. This process is repeated until the number of iterations N is reached or some other ransac conditions are satisfied, e.g., N can be adjusted inside each loop according to a better plane estimate. The plane model with the largest number of supportive points, inliers, is chosen. This whole procedure can be run repeatedly until all background planes are detected (see Fig. 3.7).

$$N = \log(1 - p) / \log(1 - (1 - \epsilon)^s) \quad (3.2)$$

Where p is the probability that at least one set of selected plane points is free of outliers, ϵ is the proportion of outliers and s the size of model points.

For the second procedure, the model set of three points defining the best plane candidate is directly selected by picking the corresponding pixels on the 2D image; After confirming the selected pixels have valid depth values the 3D plane points are selected as those lying inside a fixed perpendicular threshold distance from the plane. This second procedure can be employed when the first method is not giving optimal results, or the visible 3D background surface is too small or shattered, and we want to *force* a 3D plane model under these conditions. Moreover, among all kind of possible background regions that can be found inside a scene, in this work we are primarily interested in the **supporting** or **interacting background planes** since these planes do not only support objects but also actions or activities are commonly executed upon them. In this work we apply mainly the second procedure to model directly these planes; Fig.3.7 shows some examples of background detection and in Sec.3.3 we continue the analysis of these regions, specifically, the supporting surfaces.

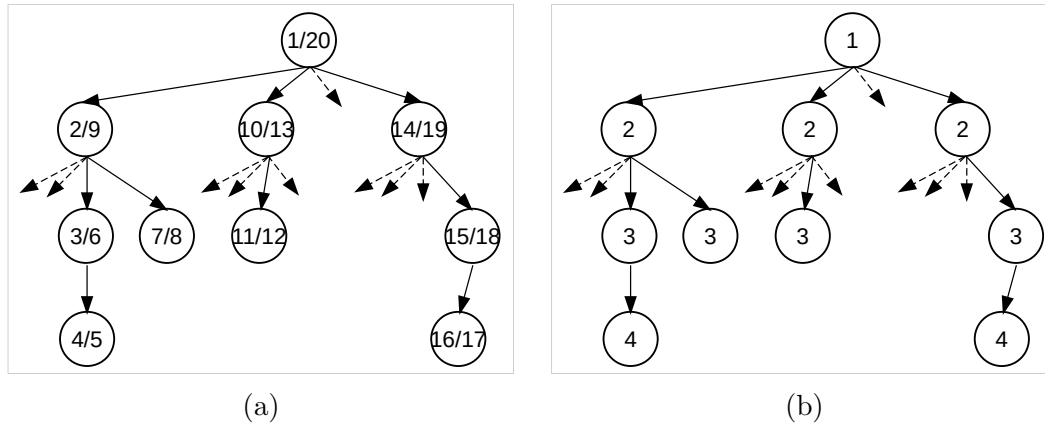


Figure 3.5: Graph traversing algorithms for a tree-based DS: a) Depth-first search (DFS) and b) Breadth-first search (BFS). DFS traverses the nodes deeper whenever is possible as indicated by the searching-timestamps inside the nodes that mark the **discovery/finishing** times, respectively; BFS firstly traverses the nodes on the same tree level at the same search step, which is also indicated by the traversing step numbers.

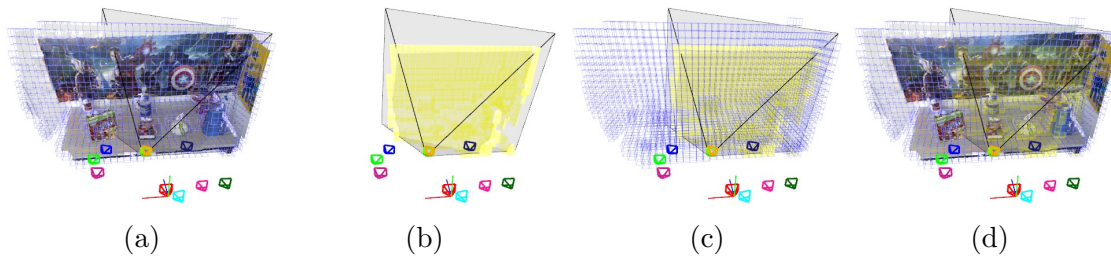


Figure 3.6: Implementation of a 3D Octree-based storage and mapping. a) shows a 3D mapped scene along with all previous camera poses. In some applications it is practical to detect b) the oct-leaves (yellow) inside a camera Field of View (FoV) or camera frustum, or to count with c) an octree occupancy model: only occupied oct-leaves (blue wire cubes) are shown; d) highlights the detected oct-leaves inside the current camera scope.

3.2.4 Foreground Detection

After background detection the 3D structural association among foreground points is achieved by identifying the occupied oct-leaves throughout the octree following the *DFS* (Fig.3.5a) to associate neighbor leaves. The outcome of this *DFS* leaf-connecting process is a group of 3D oct-leaves **segmented** out of the scene. We **cluster** the 3D points inside each single bunch of segmented leaves to obtain a set of 3D **blobs** that represent potential object candidates in the scene foreground. Under some circumstances the octree resolution (leaf size) might be increased (smaller leaf size) in order to adapt the 3D segmentation to the **object density** of the scene(s), and with this to avoid overlapped or merged object shapes during the clustering, see Fig. 3.8. At the geometric level, each clustered 3D blob contains firstly a partial 3D shell of the actual object structure; this portion of object hull will be completed and refined as more observations of the object are collated. Since we are not certain about the resulting entire shape during the object completion and, in passing, to simplify the



Figure 3.7: Geometrical detection of background regions. (left) Wall and table planes are found by repeatedly running ransac over the entire scene points (1st procedure); in the framework, supporting or interacting surfaces like a cooking table (middle) or a whiteboard (right) are determined by picking three plausible plane points on the image (2nd procedure).

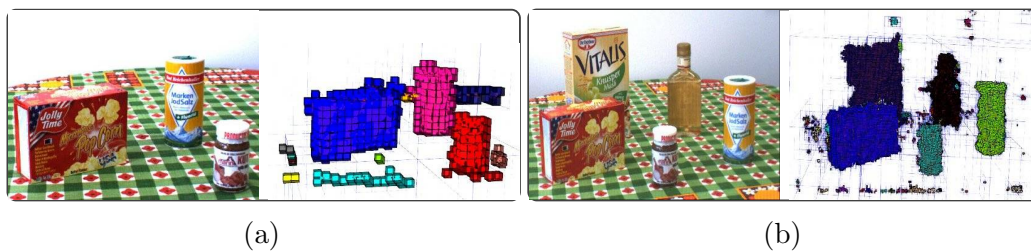


Figure 3.8: Clustering of connected leaves at two different octree resolutions in two different scenes (a, b). The size of the octree leaf can be adjusted to get a finer search and association of connected leaves and to avoid merged segmentation among closer objects.

scene representation we model or **encapsulate** each object shell into a rectangular box whose dimensions and axis orientations are estimated by the Principal Component Analysis (PCA) procedure [102] [107] as illustrated in Fig. 3.9.

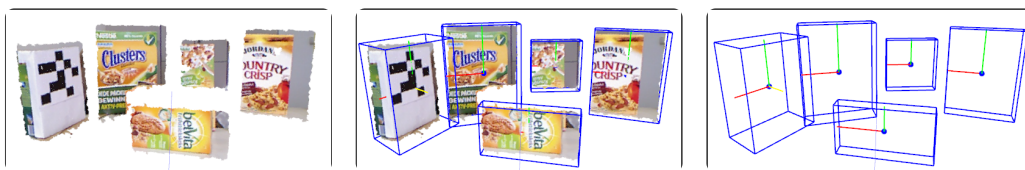


Figure 3.9: Examples of PCA encapsulation of 3D blobs. (left) clustered front faces of some boxes, (middle) the 3D PCA-encapsulated blobs, and (right) their corresponding PCA box frames and axes as an object model simplification.

As an object candidate, a 3D blob has the potential to acquire a more active and functional role in the framework. The **object container** (OC) is the model that represents this blob abstraction and enhancement, in which more object-related features are collected and stored; therefore, a clustered 3D blob also embodies a solid **token** that links the geometric and spatial data to the attributes and functional information of an object in a higher level of perception. In the Chap.4 we describe in more detail these abstract properties. In Sec.3.4 we continue with the geometrical aspects of the blobs and describe the procedures for 3D object completion. At this

point we outline the processes described so far with the data flow shown in Fig. 3.10.

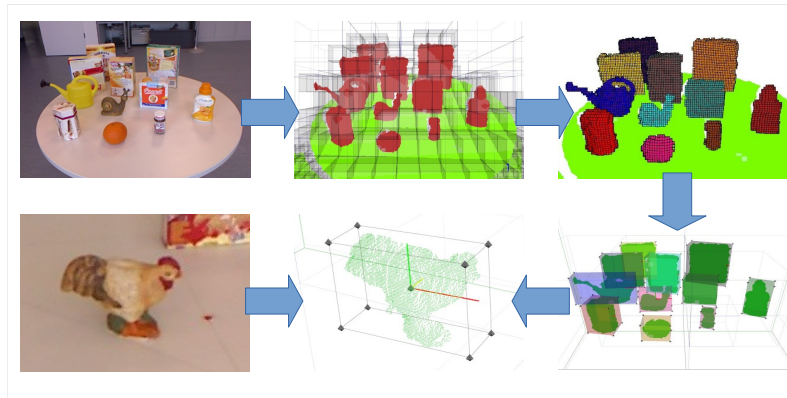


Figure 3.10: Outline of 3D scene segmentation processes. 3D data points of a scene (up-left) are stored into an octree (up-middle), background supporting plane (green) is detected and foreground structures (red) are **oct-leaf** segmented; 3D point clustering (up-right) of segmented leaves in a higher octree resolution (smaller leaf size) is performed to avoid improper merging of individual 3D blobs; finally, clustered 3D blobs are PCA encapsulated (down-right). An encapsulation example (down-middle) of a figurine (down-left) is shown.

3.3 Location Areas

We call location or functional areas to those confined spaces within the scene background that have certain semantic meanings related with either object states, like trash bin, cupboard, etc; or with object functionalities (sink, cook plate, etc;) through the execution of actions, or activities as described in [16]; in other words, they are those background spaces upon which an object is handled to fulfill a function (e.g., a plate on the table, for eating) or to confirm a state or condition (e.g., a plate in the sink, it is dirty). Although current semantic mapping approaches e.g., [103] [46] [129] [63] [13] could help us to identify relevant areas or rather furniture, inside the scenes it is common that more than one location area can lie or exist inside a piece of furniture or in a single surface area as shown in Fig. 3.11a. Opposite to an action or object manipulation a transportation action is a translational movement that generally describes a simple path that connects two functional areas, see Fig. 3.11b; Identification of ending points of transportation actions can lead in turn to the uncovering of possible functional areas. The Fig. 3.11 shows a cooking-activity scenario in which the table surface has been divided into different functional areas corresponding to some cooking tasks like chopping, frying, stirring, etc.

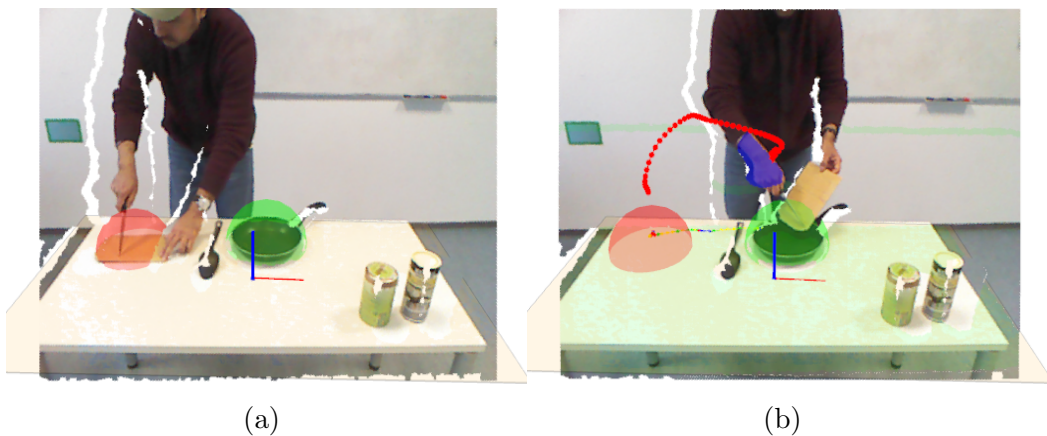


Figure 3.11: Example of location areas in a cooking scene. a) The background plane surface was detected and two locations areas corresponding to some cooking tasks were defined: the red half-sphere indicates the *chopping* area and the green half-sphere defines the *cooking plate* area. b) The transportation action path (red points) described by the segmented hand (in purple) connects both areas for a *chopping-frying* sequence. Location areas are those regions in the background scene upon which an object is handled to accomplish its function, this is, where an action occurs.

3.4 3D Object Completion

The result after the oct-leaf segmentation and point clustering (Sec.3.2.4) is a set of independent 3D structures or blobs that represent object candidates in the scene foreground. This set of blobs composes a **Blob Map**, which is a mapping tool concept that helps to detect spatial overlappings between the newly encapsulated with the stored blobs; it also supports the object-completion management by updating the map after the blob fusion processes as described below. Basically, our map $\mathbf{M}(k) = \{\mathbf{B}_j(k)\}$ at time k (see Fig. 3.12) is defined as the set of blobs $\{\mathbf{B}_j(k)\}$, where each blob $\mathbf{B}_j = \{p_i, P_i, CV_i\}$, is in turn a set of 3D clustered points p_i , along with their spatial uncertainty P_i and confidence value CV_i , (Sec.3.4.3).

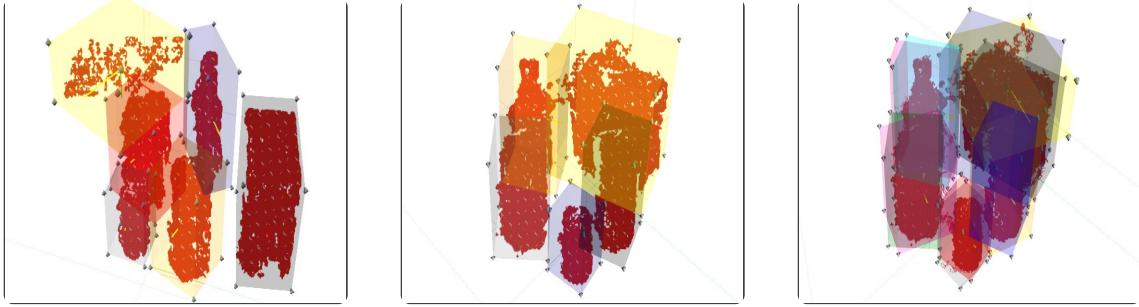


Figure 3.12: Blob Map: Two sets of 3D blobs of the same scene captured from different camera poses at times (k) and ($k+1$). (left) set of blobs $\{\mathbf{B}_j(k)\}$, (middle) set of blobs $\{\mathbf{B}_i(k+1)\}$, (right) intersection of both sets observed at pose ($k+1$).

Object completion basically consists to collate the 3D structural data of the mapped blobs with the corresponding data of the newly encapsulated blobs to compose a consistent and closer 3D model of the actual object shape; A core concept for this is **Data Fusion**, in which it is defined how the previous- and newly captured data are to be combined and/or organized; Object Fusion is addressed in next Sec.3.4.1. Since object completion is carried out blob-wise, a first step is to determine the corresponding blobs to be collated. The pseudo-code for the blob 3D-intersection search is outlined in Algorithm 1. In Algo-line 1.5 the condition refers to the intersection of 3D geometric bodies, in our particular case, to the collision of rectangular solids; for this kind of plane-face geometries it is sufficient to detect a contact of an edge on one object with a face of the other as suggested in [10]; we obtain as output a list of geometric intersections of mapped with new blobs $\Psi(k) = \{\psi_b\}$.

3.4.1 Object Fusion

The main objective of a Data Fusion System (DFSys) is to combine and integrate data from different (types of) sensors, or multiple observations from the same object in order to understand the phenomenon under observation, and with this to increase the confidence of such observation when several sensor readings confirm that phenomenon state or measurement. Under a DFSys architecture [110] two essential procedures are: the **data alignment** and the **data association**. The former consists in transforming the data received from the different observations into a common spatial and temporal reference frame, it comprises basically coordinate and time transformations, see Fig. 3.13. The latter is related with the correlation of the

Algorithm 1 Blob Intersection Search

Require: Map $\mathbf{M}(k) = \{\{\mathbf{B}_j(k)\} \cup \{\mathbf{B}_i(k+1)\}\}$, with $j = 1, 2, \dots, J$ and $i = 1, 2, \dots, I$

Ensure: List of intersection containers $\Psi(k) = \{\psi_b\}$ where $\psi_b = \{\mathbf{B}_b(k) \cap \mathbf{B}_m(k+1)\}$ with $b \leq J$ and $m \leq I$

```

1:  $\Psi(k) \leftarrow \emptyset$ 
2: for all  $\mathbf{B}_j(k) \in \mathbf{M}(k)$  do
3:    $\psi_j \leftarrow \emptyset$  ▷  $\mathbf{B}_j(k)$ 's intersection container
4:   for all  $\mathbf{B}_i(k+1) \in \mathbf{M}(k)$  do
5:     if  $\mathbf{B}_j(k) \cap \mathbf{B}_i(k+1)$  then
6:        $\psi_j \leftarrow \psi_j \cup (\mathbf{B}_j(k) \cap \mathbf{B}_i(k+1))$ 
7:     end if
8:   end for
9:   if  $\psi_j \neq \emptyset$  then
10:     $\Psi(k) \leftarrow \Psi(k) \cup \psi_j$ 
11:   end if
12: end for
13: return  $\Psi(k)$ 

```

multiple observations to determine whether a group of those belong to the same or a new event or target. Roughly speaking, given two noisy data sets, the data association (DA) problem is defined as that of finding for each data point in one of the sets the appropriate corresponding matching data point in the other set, as illustrated in Fig. 3.14. DA problems arise mainly in registration systems in which new observations have to be integrated to previous ones.

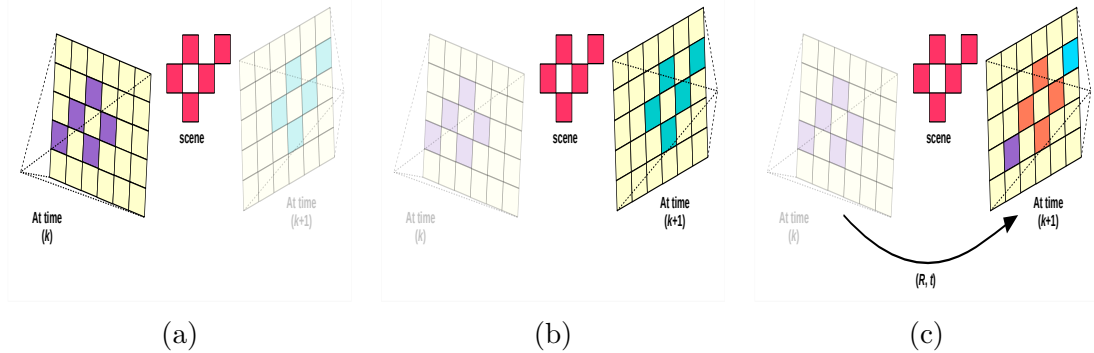


Figure 3.13: Data Alignment: Data re-projection for object fusion. Scene data (red squares) is observed by camera planes (yellow screens) at different times and poses: a) At time k data is captured (purple pixels) by the camera at the initial pose, b) camera pose at time $k+1$, new data is captured (cyan pixels) from the same scene, c) the captured data at time k is re-projected to the camera frame at time $k+1$, the orange pixels indicate the re-projection of previous data at occupied pixels by the current data.

Typically in registration methods, the output after minimizing the distance of entity correspondences between two sets is a rigid, global spatial correction that is applied and affects to all entities in such sets uniformly. In stereo vision, the 3D

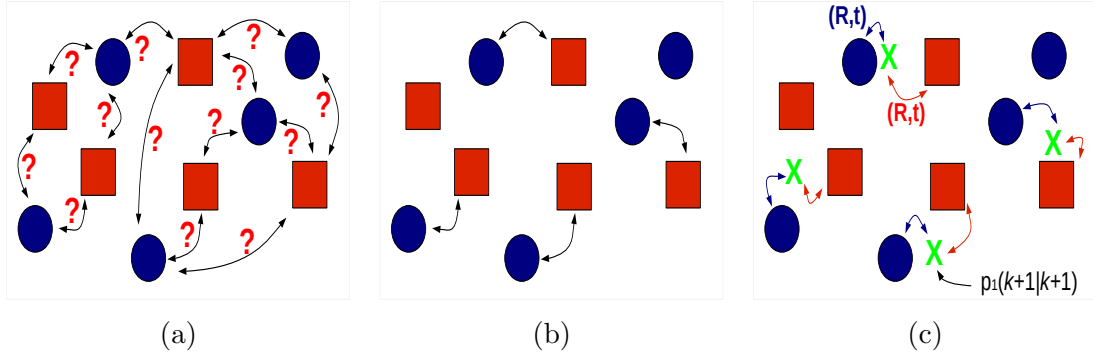


Figure 3.14: Data Association (DA): Example of Data Matching between two point sets (discs and squares). a) DA searches for potential matchings at local, point level, b) the most plausible correspondences are defined based on a fixed metric or criterion (e.g., distance, color, etc); c) a global correction, here a transformation $[R, t]$, is applied and evaluated; data can be fused (green crosses).

reconstruction of points does not present a fixed uniform error, and applying such a global correction might hinder a proper registration in some regions while improving it in others. As we know, in this framework the newly captured 3D data is associated to our map not as rigid entity but as a group of individual clusters, the blobs; the fusion between new sensor readings and the map is performed in a blob-paired manner considering the local measurement errors presented in that region and the association metric of the tentative matching entities. A benefit in this scheme is that of correcting and updating the state of individual blobs by propagating to all its points any spatial modification even in the presence of occlusion or partial visibility. Algorithm 2 outlines the steps to iterate through the list $\Psi(k)$ (Algo.1) to fuse the blobs and obtain an updated map $\mathbf{M}(k + 1)$; in this case the condition in line 2.5, makes reference to the fusion of blobs through the association of their points. The call `DOASSOC()` (Algo-line 2.6) follows the DFSys for the data alignment and association procedures that are treated in Sec. 3.4.2; the call `UPDATELIST()` is required to update the references of the blobs as they are fused.

3.4.2 Z-buffered Re-projection and Data Association

With z-buffered re-projection we refer to the spatial transformation of data points from its original capturing spot i.e., at camera pose $\{C_1\}$, to a second pose $\{C_2\}$. We emulate the camera pinhole model at the second pose and re-project the first set of captured data onto this second camera screen to associate the two sets of data as shown in Fig. 3.13; To choose which data point is visible when more than one is re-projected to the same emulated screen pixel we utilize the z-buffer of each point to determine its depth in the scene, the point closest to the camera is selected as the visible one. In Fig. 3.15 we show different re-projection screens of two 3D blobs each captured at different poses. In this re-projection phase we imply that the transformation estimates between camera poses are based on positional information that has been determined previously with some degree of certainty by an unbiased sensing process.

At time $k + 1$, a measurement $z_i(k + 1)$ of a point $p_i(k + 1)$ with covariance matrix

Algorithm 2 Blob Fusion**Require:** Map $\mathbf{M}(k)$ and List $\Psi(k)$ **Ensure:** Map $\mathbf{M}(k+1)$

```

1:  $\Gamma \leftarrow \emptyset$  ▷ updated list of blobs
2: for all  $\psi_b \in \Psi$  do
3:    $\mathbf{B}_b(k+1) \leftarrow \text{EXTRACTFROMMAP}(\mathbf{B}_b(k))$ 
4:   for all  $\mathbf{B}_m(k+1) \in \psi_b$  do
5:     if  $\mathbf{B}_b(k+1) \cap \mathbf{B}_m(k+1)$  then
6:        $\mathbf{B}_b(k+1) \leftarrow \text{DOASSOC}(\mathbf{B}_b(k+1), \mathbf{B}_m(k+1))$ 
7:     end if
8:   end for
9:    $\Gamma \leftarrow \Gamma \cup \mathbf{B}_b(k+1)$ 
10:   $\text{UPDATELIST}(\Psi(k))$ 
11:   $\text{UPDATELIST}(\Gamma(k))$ 
12: end for
13:  $\mathbf{M}(k+1) \equiv \mathbf{M}(k) \leftarrow \mathbf{M}(k) \cup \Gamma$ 
14: return  $\mathbf{M}(k+1)$  ▷ Map updated

```

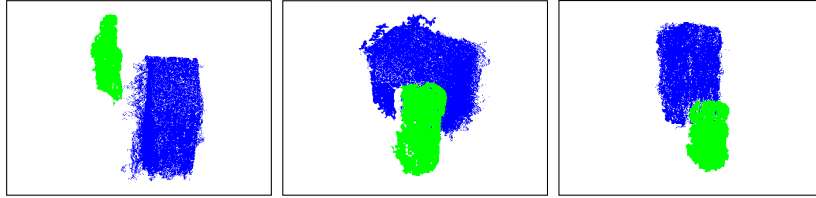


Figure 3.15: 2D z-buffered re-projection images of two 3D blobs in three different scenes. The blobs in blue were captured at previous camera poses (k) and re-projected to camera screens at time $k+1$ where the objects in green were captured. Since they were not captured at the presented camera screen ($k+1$) and due to the re-projection note how the shapes of the blue blobs are less well-defined i.e., diffuse points at the edges and some blanks inside.

$R_i(k+1)$, is normally distributed around its estimated value or state $\hat{z}_j(k+1|k)$ of $p_j(k+1|k) = T_k^{k+1} p_j(k)$ with covariance $P_j(k+1|k) = H_j P_j(k) H_j^T$, where T_k^{k+1} represents a rigid transformation to the current pose and H represents the measurement model matrix; the innovation or observation residual is defined as:

$$v_{ij}(k+1) = z_i(k+1) - \hat{z}_j(k+1|k) \quad (3.3)$$

the quantification of similarity between the observations is given by the Mahalanobis distance χ^2 :

$$\chi_{ij}^2 = v_{ij} S_{ij}^{-1} v_{ij}^T < \chi_\alpha^2 \quad (3.4)$$

where

$$S_{ij} = P_j(k+1|k) + R_i(k+1) \quad (3.5)$$

is the innovation covariance. In order to find the correspondences between an observation $z_i(k+1)$ from a data set $\mathbf{S}_1 = \{p_{1,i}\}$ with $i = 1, 2, 3, \dots, n$ and prediction state $\hat{z}_j(k+1|k)$ from a data set $\mathbf{S}_2 = \{p_{2,j}\}$ with $j = 1, 2, 3, \dots, m$, that fulfill

Eq.3.4, we re-project both sets to an emulated camera screen with camera model $C(k+1)$ in the current pose $T(k+1)$

$$(u_i, v_i) = C(k+1)z_i(k+1) \quad (3.6)$$

$$(u_j, v_j) = C(k+1)\hat{z}_j(k+1|k) \quad (3.7)$$

By the re-projection we have bucketed [130] the data points in screen pixels (u, v) and confined the matching search to a small pixel neighborhood around the predicted state represented by (u_j, v_j) and corresponding to the point $p_j(k+1|k)$. The matching search cost is reduced from $\mathcal{O}(nm)$ to $\mathcal{O}(km)$ where k is the number of neighbor pixels. Considering bucketing as a re-quantization loss of information can occur when more than one point is re-projected to the same pixel. For the case in which we re-project the current-pose observed points, (Eq. 3.6), the loss is negligible. For the case of Eq. 3.7, we examine the z-buffer of each point in order to determine which point is visible and which ones are occluded. We consider only the visible points of each set for association as we are interested in the restoration of what can constitute the shell or chassis of the actors in the scene.

The validation of correspondences by Eq.(3.4) constitutes the local-heuristic part of the process. After this, a new set of corrected, fused points are obtained:

$$p_l(k+1|k+1) = p_j(k+1|k) + K(k+1)v_{ij}(k+1) \quad (3.8)$$

with the gain K defined by:

$$K(k+1) = P_j(k+1|k)H_jS_{ij}^{-1} \quad (3.9)$$

and the covariance propagation

$$P_l(k+1|k+1) = P_j(k+1|k) - K(k+1)H_jP_j(k+1|k) \quad (3.10)$$

We obtain a set $\mathbf{L} = \{(\mathbf{S}'_1, \mathbf{S}'_2)_l\}$ of L matching-ordered subsets $(\mathbf{S}'_1, \mathbf{S}'_2)$, where $\mathbf{S}'_1 \subset \mathbf{S}_1$ and $\mathbf{S}'_2 \subset \mathbf{S}_2$ and a set of fused points $\mathbf{F} = \{f_l \equiv p_l(k+1|k+1)\}$ with $l = 1, 2, 3, \dots, L$. As a global matching validation we apply RANSAC to the matched subsets in order to determine the two rigid transformations [2] that relate with the smallest error each of them to the corrected point set, Eq. 3.11

$$\text{RANSAC}(\Sigma_i^2 = \sum_{s \in \mathbf{S}'_i, f \in \mathbf{F}} \|f - (\mathbf{R}_i s + \mathbf{T}_i)\|^2) \quad (3.11)$$

for $i = 1, 2$.

The application of these rigid transformations to the entire blob sets $\mathbf{S}_1, \mathbf{S}_2$, can also be considered as a blob state update propagation in a static environment since they modify spatially the blob data points to the current state.

3.4.3 Object Maintenance

The object maintenance at the geometric layer is based on the degree of credibility or confidence value (CV) assigned to each point during the fusion process, i.e., a value that corresponds to the existence of each point. Within the re-projection process we can state whether a certain re-projected data point is inside the current camera scope or screen, and with its depth we can also infer whether it is visible or not. This

leads to the next analysis based on the **current observed** point $p_i(k+1)$, or p_{curr} for simplicity, at time $k+1$. Similar analysis would result based on the **predicted** point $p_j(k+1|k)$, here p_{pred} , i.e., a point captured at time k and re-projected at $k+1$. Fig. 3.16 illustrates this analysis.

p_{curr} is visible and ...

- it has to be visible, i.e. there exists a corresponding, previous point p_{pred} predicting the existence of p_{curr} to be matched. This constitutes a **matching**. The confidence value of the fused point is increased.
- it has not to be visible, i.e. there is not a corresponding predicting point: p_{pred} is occluded or out of scope. p_{curr} might be either a new or a noisy point. This is an **unpredicted observation**. The confidence value of p_{curr} is initialized.

p_{curr} is not visible and ...

- it has to be visible, i.e. there is a corresponding prediction p_{pred} anticipating the point p_{curr} at this camera pose: $p_j(k)$ was probably a noisy point or outlier. This is an **unobserved prediction**. The confidence value of the predicting, previous point p_{pred} is decreased.
- it has not to be visible. This affects to the rest of the points in- or outside the scope. Their confidence values remain the same.

In general, CV update of data or features is based on the number of times an observed feature is supported $so(k)$ or unsupported $uo(k)$ up through time k ; this same procedure applied at blob/object level will be the base for an object-removal/addition detection (surprise detection); with learning α and forgetting β rates the rule to CV updating can be implemented as: $CV_i(k) = 1 - e^{-(so_i(k)/\alpha - uo_i(k)/\beta)}$, as proposed in [64].

Experimental evaluations and results corresponding to the implementations and approaches presented in this section are shown in Chap. 5.

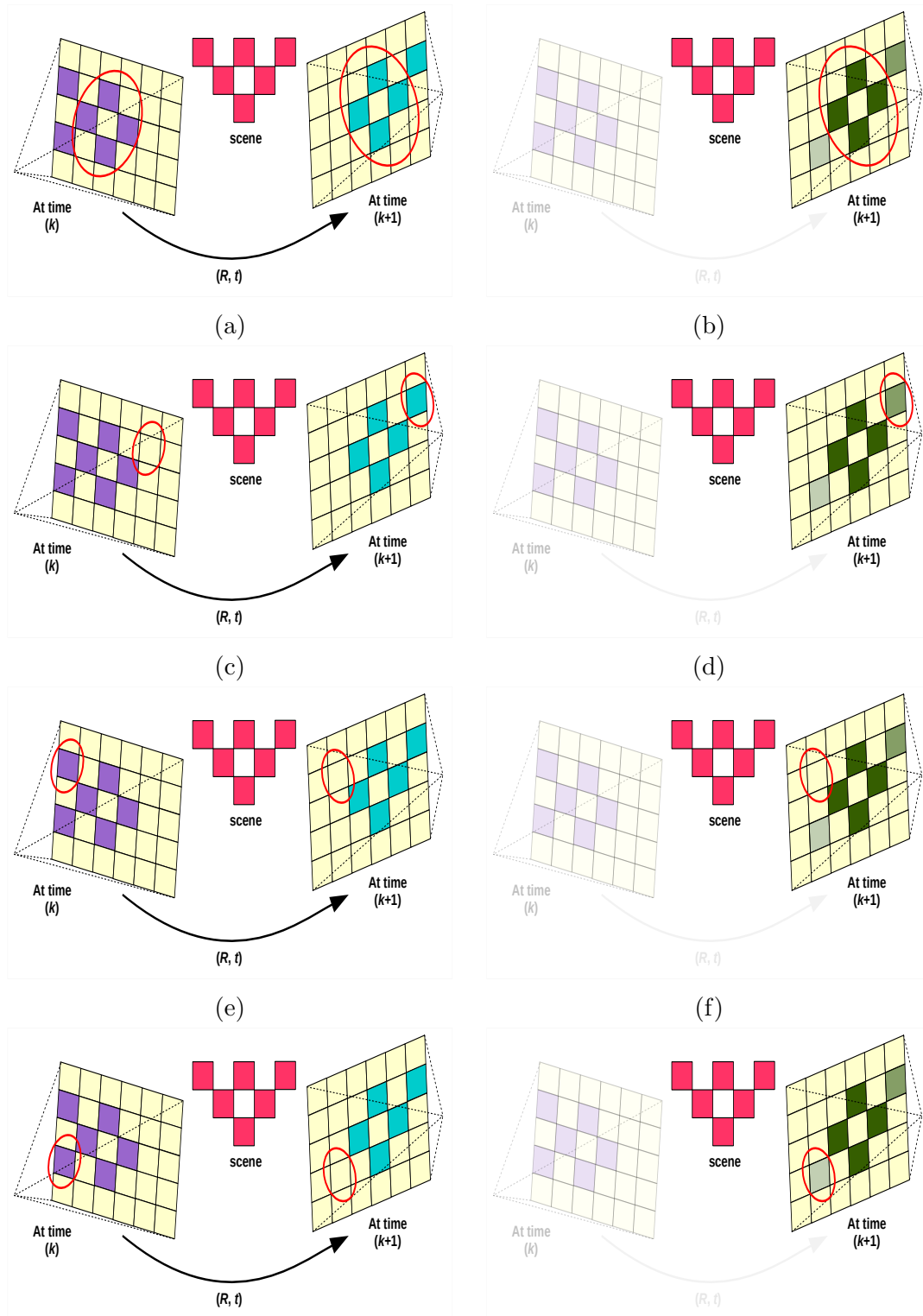


Figure 3.16: The four cases of CV assignment. First row illustrates a **matching** of pixels/points: a) the matching points are encircled in both camera screens, b) their corresponding CV values are increased. Second row presents an **unpredicted observation**: c) a point is captured at time $k + 1$ but not in at k , d) the CV of the new (valid/noisy) point is initialized. Third row shows an **unobserved prediction**: e) the captured point at time k predicts a point inside the camera screen at time $k + 1$, f) the CV of the point is decreased. The last row shows an example of a noisy point captured at time k but not at $k + 1$ and whose CV will be hence decreased.

3.5 Foreground Motion: Non-Static Environments

Besides the structural aspects like the fusion and completion of 3D bodies at the geometric layer additional mechanisms of perception like navigation and mapping can be added at this level to better support potential and higher agents of perception. The analysis of dynamic changes provides us with more information about the spatio-temporal relations and behaviors of the structures inside the scene. This analysis can be also in turn performed at different levels of abstraction. Structures can start moving or being moved at different times, for different periods of time and at different speed rates; At the geometric level the analysis of motion can range from the perception of the initial and final poses of objects that might have been moved outside the sensor observation periods, to the uninterrupted observation of the object motions inside the sensing scope; for the former case we can estimate the motion parameters or pose modification between those only two observed states, for the latter we can follow the motion evolution over time with help of visual tracking mechanisms, with which we could obtain in general a short-lived prediction horizon of object states in the length of seconds.

In the previous section we assume the pose transformation between consecutive camera spots (e.g., for 3D object completion) was given by an external tool while the objects were kept in their places, static. In this section we work on non-static environments i.e., changes in the scene occur outside the sensor scope, see Fig. 3.17. Under these circumstances the system is then demanded to cope not only with the inherent noise of the sensory data but also with potential feature mismatches generated by dynamic factors. As a supporting component to the framework we describe in this section the visual estimation of the camera motion (ego-motion) as well as the estimation of independent motion parameters corresponding to individual objects moved in non-static scenes. This mapping update can be a supportive feature of awareness for assistant or surveillance perception systems that supervise confined and relatively well-known indoor environments. Next, we first organize and present the data we are working with for this task, then we illustrate the ego- and object-motion estimation.

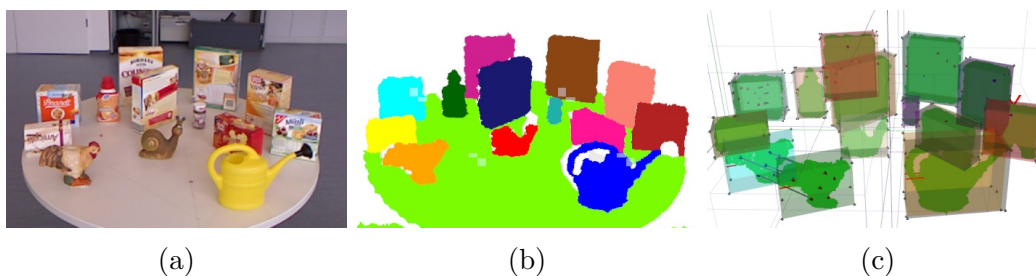


Figure 3.17: Exemplary non-static scene for visual estimation of independent foreground motions. a) 2D RGB image of the scene, b) supporting plane is detected and tentative objects are clustered and mapped, c) an object was moved and the change of its blob pose is detected, its mapped and current poses are linked through the matching features.

3.5.1 Motion Detection

At time k a set of N 3D points $\mathbf{S}(k) = \{p_n, P_n\}$ is taken from the sensor devices, being $p_n \in \mathcal{R}^3$ the measured, mean point value and P_n its z-spatial uncertainty. After segmentation of $\mathbf{S}(k)$, we define our map $\mathbf{M}(k) = \{\mathbf{B}_i(k)\}$ as a set of blobs $\mathbf{B}_i(k) = \{p_j, P_j, \gamma_j\}$, where each blob point is assigned a confidence value γ_j , as described in Sec.3.4.3.

We also employ a set of 2D image features $\mathbf{I}(k) = \{u_f, v_f\}$. These features are detected by the Speed-Up Robust Features (SURF) detector [5] with each of these having a corresponding 3D feature point in the set $\mathbf{f}(k) = (p_f, P_f)$ related by $H : (u_f, v_f) \mapsto (p_f, P_f)$, where H is a mapping function from pixel to 3D coordinates. At pose $(k + 1)$ new blob and feature sets $[\{\mathbf{B}_j(k + 1)\}, \mathbf{f}(k + 1)]$ from sensory data $[\mathbf{S}(k + 1), \mathbf{I}(k + 1)]$ are determined and a set of L 2D-feature correspondences $\mathbf{C}_{2D} = (\mathbf{I}_1, \mathbf{I}_2)$ is established, where $\mathbf{I}_1 \subseteq \mathbf{I}(k)$ and $\mathbf{I}_2 \subseteq \mathbf{I}(k + 1)$; Their corresponding set of 3D matching points $\mathbf{C}_{3D} = (\mathbf{F}_1, \mathbf{F}_2)$ is also determined from \mathbf{C}_{2D} by means of the mapping function H . Fig. 3.18 presents an example of independent motion detection by 2D feature detection and matching when camera and objects are moved. Although the matching lines seem to be generated by a same single motion the longer lines of the mid object suggest the effect of an extra motion component.

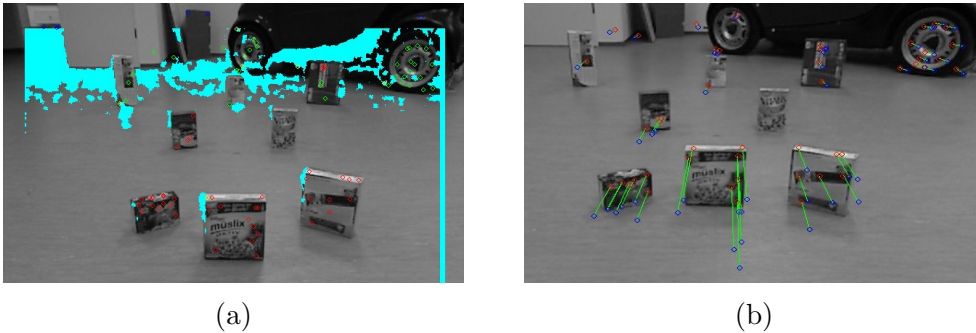


Figure 3.18: Example of 2D detection of independent motions in non-static scenes. a) First set of detected features $\mathbf{I}(k)$: features with valid 3D points are presented by red circles, with invalid 3D values in green and areas with no-depth values in cyan. b) Valid 2D feature matches $\mathbf{C}_{2D} = (\mathbf{I}_1, \mathbf{I}_2)$: the closest object to the camera in (a) was moved backward while the sensor was moved forward; Although sensor and object moved in the same direction we can observe that the lengths of the matching lines differentiate each motion: while the objects on the sides (and some objects behind) support the camera motion, the longer matching lines of the mid object characterizes its only independent motion.

3.5.2 Ego-Motion Estimation

Assuming a static scene only with camera motion, all lines of matching features would ideally correspond to this single motion and they would converge to one specific point, the **epipole**, which is the projection of the camera center of the previous pose onto the current camera screen. Unlike a homography computation between two images of a planar surface or two images involving a pure rotation due to the spatial structure given by the corresponding 3D points of the 2D matching features we can estimate a rigid transformation; like previously applied, we find a

rotation matrix and a translation vector $[\mathbf{R}^{ego}, \mathbf{t}^{ego}]$ in this case, involving only the 3D feature points of the whole scene (back- and foreground) $\{\mathbf{C}_{3D}\}$ that minimizes a cost function in Eq. 3.12.

$$\Sigma^2 = \sum_{l=1}^L \|p_{2,l} - (\mathbf{R}^{ego} \cdot p_{1,l} + \mathbf{t}^{ego})\| \quad (3.12)$$

with $p_{1,l} \in f_{1,l}$ and $p_{2,l} \in f_{2,l}$. Essentially due to noisy sensor readings or feature mismatches not all 3D lines of matched features in $\{\mathbf{C}_{3D}\}$ support the minimization in Eq. 3.12. We discriminate under a ransac scoring scheme the matching features that support the camera ego-motion (i.e., the inliers), and the matches caused by independent flow of motions (i.e., the outliers). We find a proper subset of matched features $(\mathbf{F}'_1, \mathbf{F}'_2)$ that is geometrically consistent with the motion of the cameras; we define the transformation hypothesis $(\mathbf{R}^{hyp}, \mathbf{t}^{hyp})$ with the largest amount of scores as the one which gives this set of inliers. The scoring is based on the similarity of the matching points:

$$p'_{1,j} = \mathbf{R}^{hyp} \cdot p_{1,j} + \mathbf{t}^{hyp} \quad (3.13a)$$

with

$$v_{jj} = p_{2,j} - p'_{1,j} \quad (3.13b)$$

$$\chi_j^2 = v_{jj} S_j^{-1} v_{jj}^T < \chi_\alpha^2 \quad (3.13c)$$

and

$$S_j = P_{1,j} + P_{2,j} \quad (3.13d)$$

where $(p_{1,l}, P_{1,l}) \in \mathbf{F}_1$ and $(p_{2,l}, P_{2,l}) \in \mathbf{F}_2$. We use the set of matched points that fulfill the Mahalanobis metric χ^2 of Eq. 3.13c to minimize the sum of squared residuals Σ^2 of Eq. 3.12 and to obtain the transformation from pose k to pose $(k+1)$ corresponding to the ego-motion of the cameras. The matched pairs that do not fulfill Eq. 3.13c constitute the group of outliers. In Fig. 3.19a the set of flow inliers supporting the ego-motion is shown.

3.5.3 Independent Object Estimation

We profit from the fact that not all information classified as outlier is derived from noisy or mismatched data, and that this information gives in turn new patterns indicating independent events inside the scene. At this stage we can distinguish three types of outlier flows: generated by noisy 3D points, generated by mismatched features and the ones generated by independent object motions. In order to detect these last good matches for each independent motion we inspect the spatial connections that each matching flow provides between the mapped and newly captured blobs. Detecting that some outlier features in \mathbf{F}_1 and their correspondences in \mathbf{F}_2 belong to some blobs at time k and $(k+1)$ respectively, i.e., $\{(f_{1,l})_i\} \in \mathbf{B}_m(k)$ and $\{(f_{2,l})_i\} \in \mathbf{B}_n(k+1)$, we could infer that blob $\mathbf{B}_m(k)$ was moved to blob $\mathbf{B}_n(k+1)$ and compute its motion parameters $[\mathbf{R}^n, \mathbf{t}^n]$ by following the same procedure for the ego-motion estimation but now with a reduced set of I outliers $\{(f_{1,l}), (f_{2,l})\}_i$. Fig. 3.19b shows the subset of outliers from the set of matches shown in Fig. 3.18 that indicates the motion of the object; this first motion estimate can be refined with Eq.3.12 now taking into consideration all the blob points; the Iterative Closest Point (ICP) procedure [7] can be also applied for the refinement.

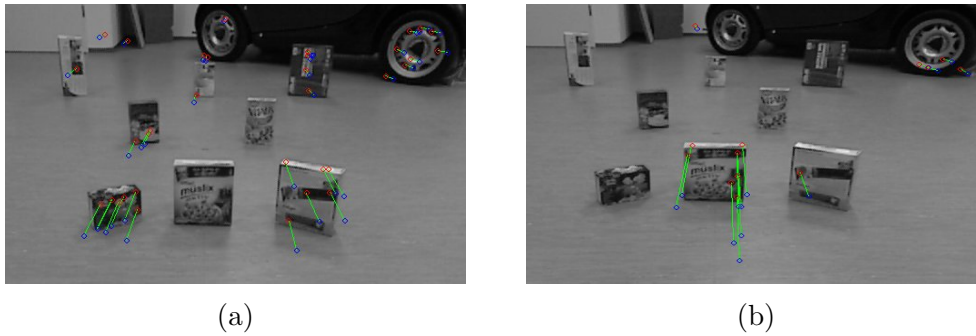


Figure 3.19: Exemplary scene for feature-matched flows of inliers and outliers for camera ego- and object motion estimates. a) Flow of matching inliers supporting the ego-motion of the sensor, b) flow of matching outliers indicating the motion of the object.

3.6 Foreground Actor Tracking: Dynamic Environments

We understand as foreground actors those active or dynamic 3D structures in the scene that are able to move or to generate motion in other scene structures; segmented 3D blobs as tentative object candidates are the palpable choice to represent potential foreground actors. In order to identify these potential objects in the scene, object recognition and detection can be supported and achieved in our framework by the addition of external applications, like OpenCV feature matching modules (Fig. 3.20a), You-Only-Look-Once approach (YOLO) [93] (Fig. 3.20b) or other available, public libraries. In general, the output of these approaches are 2D-image information like feature-matching locations or 2D bounding boxes, as shown in Fig. 3.20; this information can be complemented with the 3D-scene information, e.g., locations or dimensions, of the 3D segmented blobs for a more precise modeling world.

Dynamic behavior of foreground actors can be analyzed and characterized in a higher level of perception by their motions observed and registered by visual tracking mechanisms at the geometric layer; these visual-tracking tools are extensively studied, researched and applied in the robotics and computer vision fields. Visual trackers allow to predict and correct over time the state of target or targets in despite of relative motion between the tracking device (camera) and the target(s), occlusions, image blur, jitter, changes of illumination, etc; currently a huge variety of tracking methods have been developed to adapt and better cover an equal huge variety of applications [84] [65] [105] [29] [57].

3.6.1 Visual Object Tracking

Object tracking is performed by utilizing visual salient, distinctive features or attributes of interest of the target objects; these features can be shape, contour, colors, texture, pattern of edges, etc. Fig. 3.21 and Fig. 3.22 show two object tracking implementations based on 3D object models and object appearance, respectively. Fig. 3.21a shows the outputs of a contour tracker in which the 3D model of a plane square is fit to the output of an image edge detector [23] [24] [1] in three different

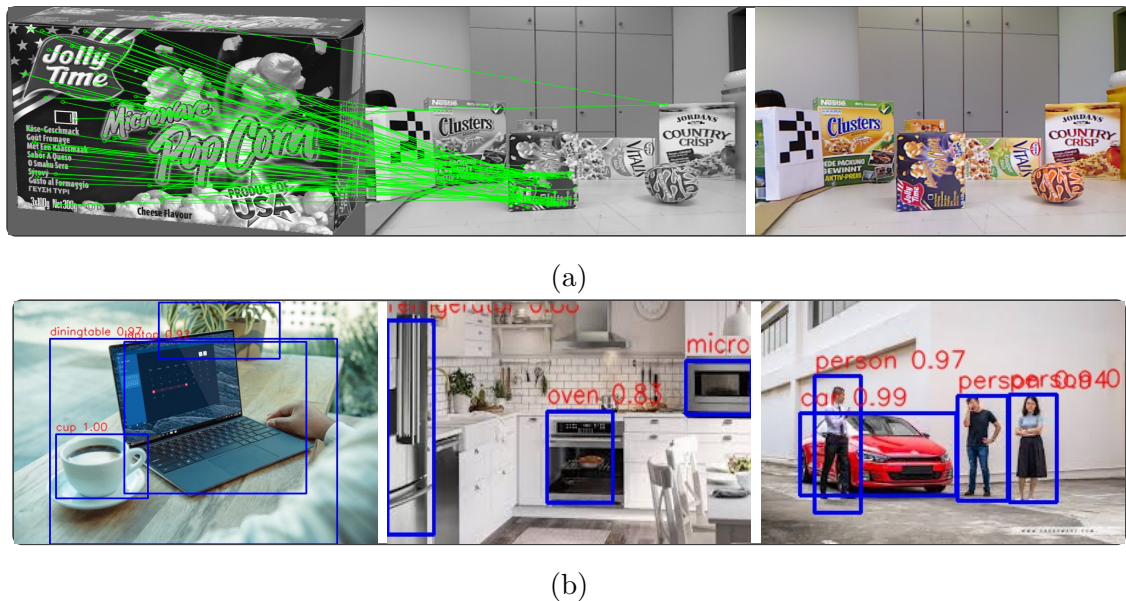


Figure 3.20: Images of OpenCV and YoLo object detection. a) OpenCV FLANN-based matcher for object recognition: most of the 2D SIFT-features detected on the object model (left) are rightly matched to the object in the scene (middle) and whose pose is different to the model’s (see right image). b) YOLO outputs (bounding boxes and probabilities) in three different scenes for object recognition.

situations; Fig. 3.21b shows a more complex contour model based on BSplines [4] to fit a person’s head-and-shoulders silhouette [53]. Fig. 3.22a shows a Particle Filter (PF) tracker [61] [88] based on the color histogram of a coffee box, and in Fig. 3.22c we can see the same PF tracker acting on a person’s face [86] [32]. Fig. 3.22b shows the model templates for these examples.

In order for some objects to interact with the environment, or rather, to fulfill their functionalities they are required to (be) move(d); in many situations, however, a person’s hand covers total- or partially the object it handles, and makes in turn, the visual tracking of the object difficult to succeed; One benefit of this, however, is that once the hand grasps an object its shape does not change abruptly, which might allow to keep a simpler 2/3D model representation of it rather than keeping a more complex model for each grasped object. Since in this work we are not interested in hand gesture recognition nor finger position detection, either, we refer to the foreground actor as the person’s hand grasping and manipulating objects, whose model can be represented as a round-ish blob surface. In this way, by knowing the object (hence, its functionalities) along with the the manipulation motions such object undergoes we can analyze the fore- and background interactions in the scene. Next, we present the two main approaches we follow to detect and track a person’s hand in the framework: by detection/Kalman Filter and by Openpose Skeleton.

3.6.2 Hand Tracking: by Detection and Kalman Filter

In this procedure the 3D segmentation of the hand is manually initialized by enclosing the area where it lies on the 2D image and its corresponding 3D hand blob is segmented by the process at the 3D image; after this first step, the 3D hand blob is detected by the visual tracker and segmented at each 2/3D image frame. Starting

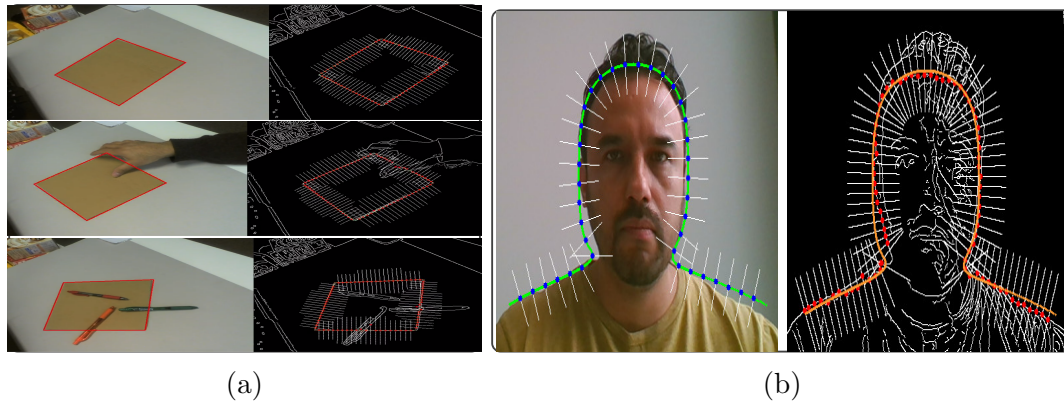


Figure 3.21: Visual 3D model-based object tracking: 3D plane square and BSpline-based contour. a) a 3D model of a plane square (red) is fit to the outputs of an image edge detector (right, dark images) in different scene situations (in motion, occlusions); b) a BSpline-based contour of a person's head and shoulders is fit to a person's head-and-shoulders silhouette. The short white lines around the contour models are locally perpendicular to the models at their respective crossing points and represent the feature searching path of that contour point over the image.

from a pixel-point ($px-pt$) seed we employ this time the Breadth-First Search (BFS) algorithm [18] (Sec. 3.2.2) to connect 2D neighbor pixels under 3D conditions, i.e., each pixel must have a valid depth value and the 3D distance between neighbor $px-pts$ must be less than a fixed threshold. Tracking by detection is a common and a simple tracking method that consists in searching and detecting either globally, i.e., over the whole current image frame, the salient features of interest and matching them with the ones detected in the previous image frame; or searching for these visual entities locally, this is, only in a confined image region close to the previously detected features. Fig. 3.23 shows a sequence of hand tracking by detection; the hand blob in the current frame is searched locally inside a 3D encapsulation sphere around the previously detected hand blob centroid. A 3D Kalman Filter (KF) [55][121] with added white-noise acceleration model was also implemented and employed as a visual tracker. KF, with its variants, are mathematical procedures extensively used in data fusion processes; their prediction-correction scheme allows them to be employed as general trackers; For this, it is required to know the previous state of the target, the target state uncertainty and the current target state measurement along with its associated measurement uncertainty. As example, we applied the KF tracker to a eating-with-a-fork sequence and show in Fig. 3.24 the different hand trajectories that can be generated by this tracker.

3.6.3 Hand Tracking: by Skeleton Detection

An alternative, supporting approach for the hand-blob detection and tracking in the framework is the integration of the OpenPose¹ body skeleton detection [14][104][15][120], as shown in Fig.3.25a. The OpenPose implementation allows the detection of human body skeletons in 2D as well as in 3D images; Additionally, the available c++ OpenPose library also offers the detection of faces and hand poses (see Fig 3.25b).

¹<http://www.openpose.x>

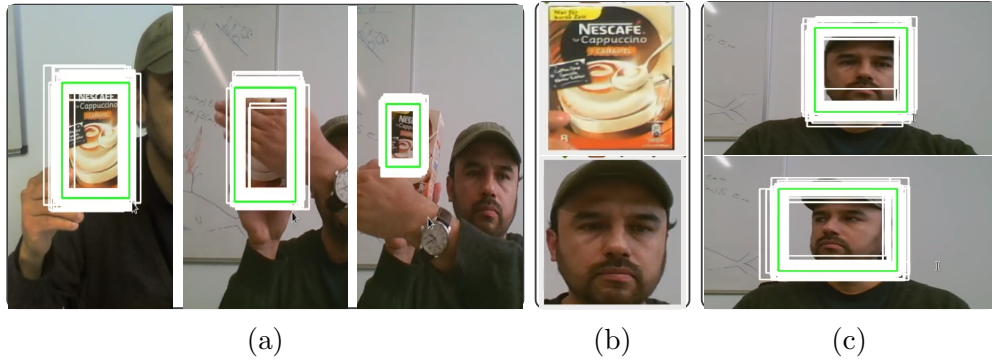


Figure 3.22: Visual appearance-based object tracking: Particle Filter (PF). a) shows the PF tracking a box front cover under temporal partial and total occlusion conditions, c) shows the PF tracking a person’s face, b) shows the model templates for these examples. The white rectangles represent the tracker particles, i.e., tentative target states; the green rectangle is the output of the PF, the average of all probable states.

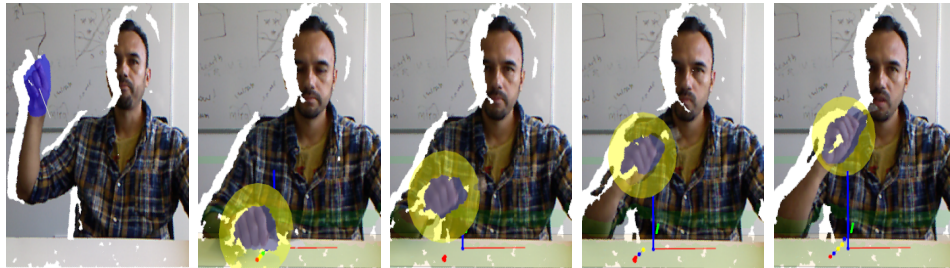


Figure 3.23: Hand tracking by detection. The left-most image shows the 3D segmented hand (blue) at the beginning of the eating-with-a-fork sequence, the rest of the images show the tracking-by-detection encapsulation sphere and the resulted segmented hand at different sequence frames.

Fig.3.25c shows the OpenPose integration in the framework for face, fingers and body parts in some task scenes. The implementation of all these additional features in our framework not only slows down the detection from frame to frame but they all are also not required at this stage for our manipulation modeling (Sec.4.2); In order to spot only the hand blob positions we augment the 25-key points skeleton frame (Fig.3.25a) by extending both arms at the wrist sides by a fixed percentage of the corresponding detected forearm length. In Fig.3.26a we can observe these additional 2D key points and in Fig.3.26b their corresponding 3D locations. Although a reliable hand-blob detection is obtained in this way, a latent drawback is that a stable skeleton detection is also required; In many cases, to get an entire and individual body detection OpenPose requires most of the person body to be visible, otherwise body fragments are detected and might be assigned to different person bodies; Fig.3.27 shows some of these cases.

3.6.4 Optical-Flow Estimation

In addition to the translational hand motion registered by the tracking tools we also aim to perceive cues of rotational movements of the hand. This is obtained by

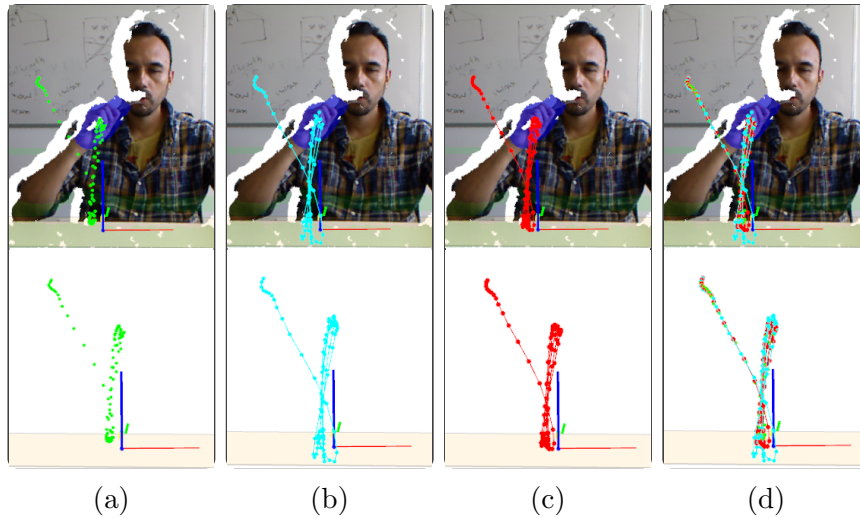


Figure 3.24: Hand tracking by Kalman Filter (KF): 3D KF tracker applied to the sequence eating-with-a-fork. a) path of measured hand blob centroids, b) path of KF predicted spots, c) path of KF corrected centroid spots and d) all paths of the sequence.

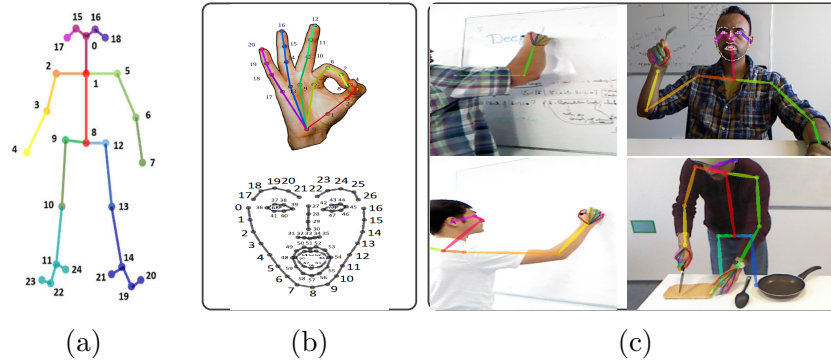


Figure 3.25: OpenPose (OP) key points and detection of body-skeleton, fingers and face. a) OP 25-key points skeleton, b) OP key points of hands and faces, c) OP skeleton, finger and face detection into the framework in some task scenes.

estimating the optical flow (OF) inside segmented hand areas between consecutive image frames. On one side the approach presented in [71] is extensively referenced and applied for the estimation of **sparse** optical flow; this method requires first the detection of salient visual features to be matched between consecutive frames; on the other hand approaches for a **dense** optical flow estimation have been also proposed [49] [31]. The main difference between these methods is that the former looks for a coordinate frame transformation between two images with help of the matching of salient features, whereas the latter refers to a vector field over two consecutive images in which every image pixel contributes to the estimation. For the scene in Fig. 3.28a, we present the sparse OF in Fig. 3.28b and dense OF in Fig. 3.28c. Considering that the hand covers in general a small area inside the visual environment and consequently it does not offer a large body surface for a sparse optical flow estimate we employ the Farneback method for a dense optical flow. In Fig. 3.29 we present some scenes of tracking and optical flow detection on a segmented hand, a manipulated object and a freely-hanging toy.



Figure 3.26: Augmented OP skeleton of hand-blob key points. a) Augmented hand-blob key points by extending the arms at the wrist sides in 2D images, the hand key points are marked with blue points inside yellow circles, b) the corresponding 3D spots of the 2D hand-blob key points are marked in the 3D images.



Figure 3.27: Examples of OP fragmented and failed detections under partial body visibility in some 2D scenes.

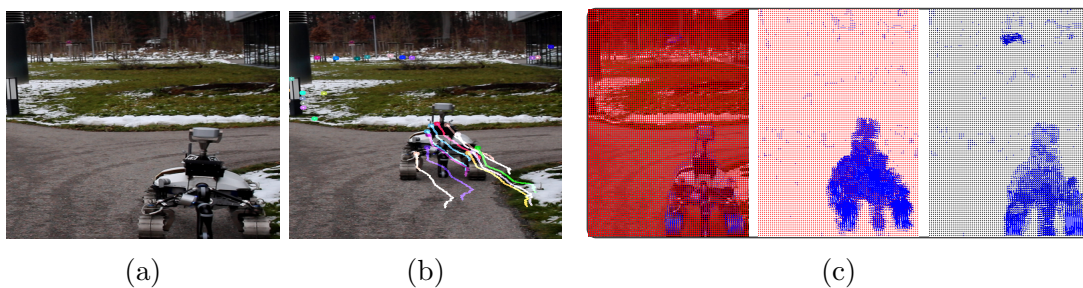


Figure 3.28: Sparse and dense optical flow (OF) examples: a) Image scene of a rover as it started moving, b) sparse OF of detected salient features, c) shows (left) the point grid over the image for dense OF estimation, (middle) OF vectors (blue lines) indicate the motion of the rover, (right) the motion of a car covering a few pixels in the image background is also perceived.

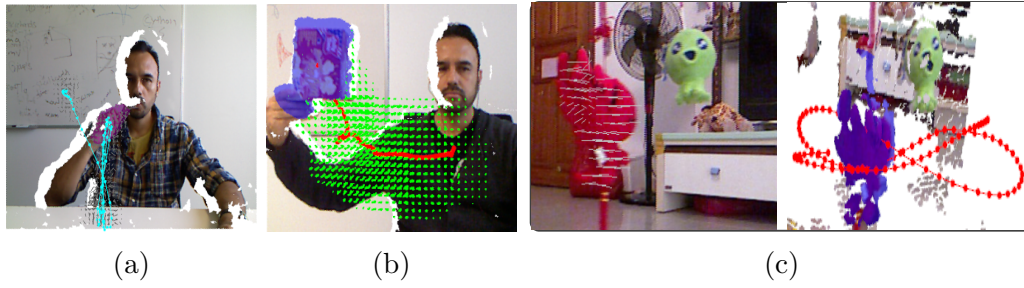


Figure 3.29: Exemplary scenes of object tracking paths and optical flow detections on: a) a hand (purple) manipulating a fork, b) an object, c) a hanging toy moving freely (left) 2D OF, (right) 3D segmentation and path.

Chapter 4

The Abstraction Layer

Sections

Overview	61
4.1 Semantic Structure	63
4.1.1 Knowledge Container, Atlas and Object Representation	64
4.1.2 Prediction Horizons	65
4.2 Dynamic Model for Manipulations	66
4.2.1 Action and Activity	67
4.2.2 Plugin Scheme	68
4.2.3 Fore- and Background Interaction Actors	68
4.2.4 Motion Representation	69
4.2.5 Manipulation Modelling	71
4.3 Dynamic Model for Autonomous Driving	74
4.3.1 Perception System	74
4.3.2 Segmentation and Mapping	74
4.3.3 Autonomous Parking Maneuver	76
4.3.4 Pedestrian Segmentation and Path Prediction	76

Overview

How high the level of abstraction or understanding about a scene is demanded depends partially on the planned and required strategies of interaction; In our framework these comprise to identify **what** information we want to extract from the scene and **how** we want to utilize it. In this chapter we present the procedures and methods with which the geometric data recollected and organized at the geometrical layer is further analyzed and utilized. First we introduce the structure of the abstraction layer, how it is organized, and describe related concepts and components, then we present three different scenarios in which the subtracted 3D blobs acquire different meanings and hence properties and functions; In the first scenario we focus on the **doer** of manipulation tasks, i.e., a person's hand, and for which we expand our framework by developing a **plugin**: *Dynamic Model of Object Manipulation* that allows us to characterize and model the manipulation motions applied to objects with different functionalities as well as to help to distinguish similar actions executed with different objects. The second scenario is an outdoor scene where we employ our framework to support an autonomous parking procedure of the highly maneuverable vehicle **RoMo** (Robo Mobil); the detected blobs acquire another different meaning, namely, as vehicles; The strategy in this case is to obtain the relevant, spatial scene information like the size of neighbor vehicles, localization and size of the parking lot in order to estimate RoMo's rotation spot and the amount of rotation as well to avoid collisions or scratches during its *crab* (sideways) parking maneuver as it enters to the parking box. In the third application, like in the first one, we consider a situation where the behavior (motions) of objects in the scene are more relevant than obtaining accurate 3D models of them; we analyze again the blobs as active elements, as pedestrians, and estimate their 2D path prediction around the RoMo.

4.1 Semantic Structure

Complex environments expose multiple parallel actions happening in different parts of the scene. Perception agents acting in the scene try to model the dynamic changes, which allows them to predict future state transitions and reduces the required frequency in which the actions need to be verified. Depending on the level of the abstraction, the **prediction horizons** may vary from a few seconds for primitive motion trajectories, over multiple seconds for basic actions, like screw tightening, all the way to multiple minutes in case that the system can recognize the current process being executed in some part of the environment. The dynamic environment model presented in [90] (also Chap. 3) provides different abstraction modalities and a-priori descriptors that can be used by dynamically configurable plugins, like navigation, object recognition, action labeling modules, etc. In Fig. 4.1 we present our abstraction layer constituted of two fundamental components: **Knowledge Container** and **Atlas**, along with an external agent of perception or **plugin**. The Knowledge Container and the Atlas are the interface between the geometric layer, represented by the dynamic 3D model of the scene, and the external, specialized plugins of different abstraction modalities like for object recognition, object transportation modelling, etc. For example, the action labeling presented in [16] characterized pure transportation actions that were segmented by changes in the contact relation between the manipulator and the object in the scene. The plugin extension presented below works on the **Location Areas** (Sec. 3.3) which are regions in the scene where main actions or activities occur other than pure transportation motions like fore- and back-ground actor interactions, manipulation of objects, etc. The plugin described in Sec. 4.2 utilizes the observed motion path associated with the dynamics of foreground objects to model and represent interactions related with certain background regions in the scene.

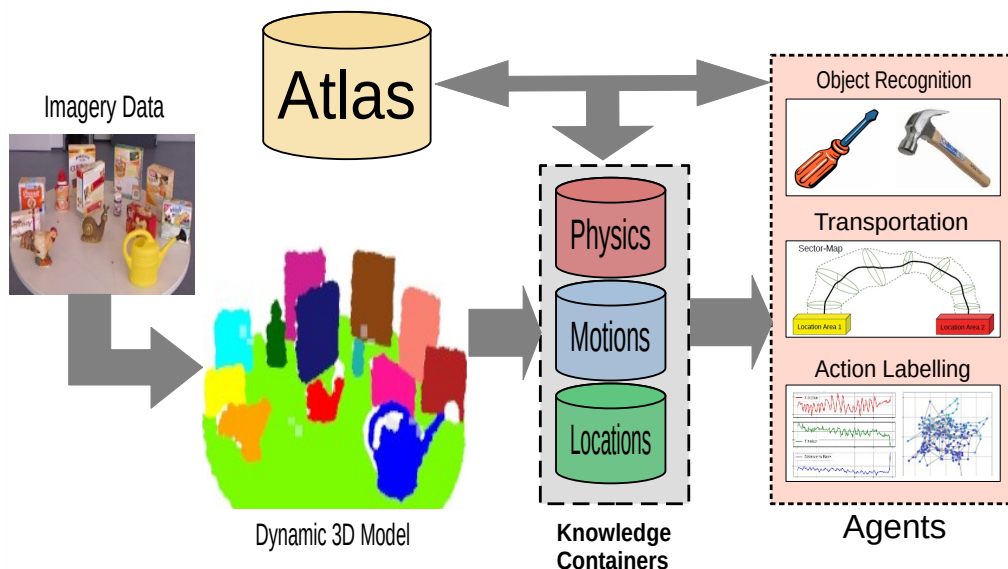


Figure 4.1: Fundamental components in the semantic structure: Knowledge Container and Atlas, plus *pluggable* extensions agents of perceptions or **plugins** comprise the layer of perception in our framework. Plugins with different abstraction modalities act on different aspects of the scene.

4.1.1 Knowledge Container, Atlas and Object Representation

These elements are the interface that store and organize the information extracted from the geometric layer and keep it available to agents of perception for further processing. The **Knowledge Container** (KC) collects the local information observed in the current scene and organizes it in three areas: **physics**, **motions** and **locations**. **Physics** is related with the physical properties of the foreground structures; **motions** stores the observed trajectories described by dynamic elements of the scene and **locations**, which is essentially the location areas defined in the scene background and that are related with objects' states and functions. The **Atlas** is our global database that contains common, generic data of the world; this represents a source of a-priori knowledge previously learned by experience (observation or interaction) in other scenarios. **Global information** stored in the Atlas can be attached to the current scene actors to enrich and complement the local information recollected in the Knowledge Container, and vice versa.

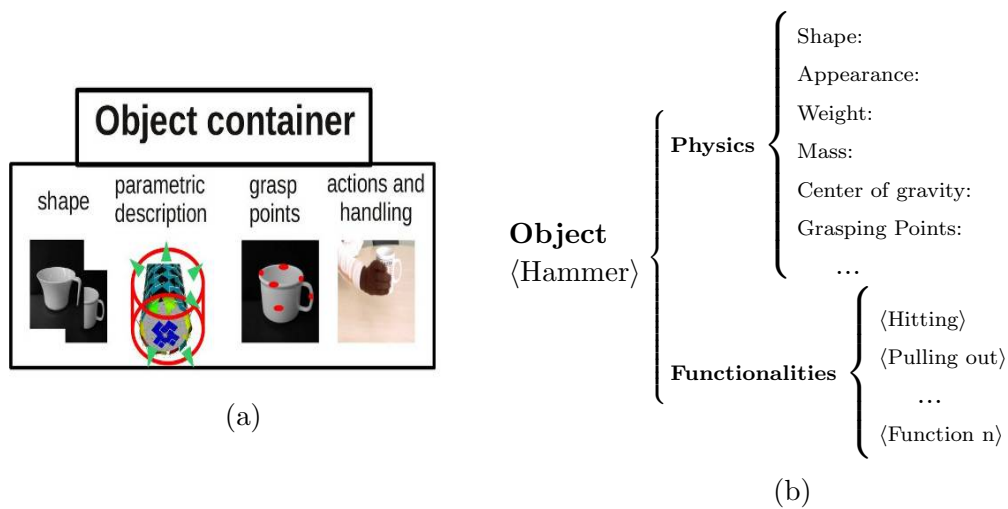


Figure 4.2: a) Object Container as abstraction of 3D-blob structure (Sec.3.2.4) is organized under the b) Object Representation; it contains two attributes for a foreground object: physics and functionalities.

As mentioned in Sec 3.2.4 the **Object Container** (see Fig. 4.2) is the enhancement of the 3D blob structure, whose geometric data is augmented with semantic, contextual information; Consequently the **Object Representation** we employed as a storage unit in the framework is depicted in Fig. 4.2. The first object attribute **physics**, comprises all the physically measurable characteristics like dimensions, color, texture, weight, etc; as well as the set of physics properties that are probably not directly perceptible or considered at first sight like center of mass, grasping points, parametric descriptions, etc; The second attribute **functionalities**, refers to the operational part of the object, whose principal objective is to satisfy a human demand: the main reason the object was made for.

4.1.2 Prediction Horizons

The visual tracking scheme of state prediction and correction presented on the geometric layer (Chap.3) is based on a dynamic model that is required to give plausible indications about a future state of a target; depending also on the nature of the observed phenomenon the predicted state normally vary from fractions to few seconds ahead of the current state. To extend this **prediction horizon** from a geometric level to multiple seconds in a higher level of awareness the agent has to be able to discern the feature(s) or pattern(s) that characterizes the main goal or objective of such event (action/activity in our case) from the short or sudden states that might belong to the natural development of the action but that also correspond to immediate, spontaneous alterations, or permanent adjustments that are irrelevant to the completion of the task. We illustrate this with the two examples shown in Fig. 4.3. The top-row images show a scene where a child starts walking from the right with the goal to reach the offered toy at the left side of the image (Fig. 4.3a); on his way the child suddenly turns back to his left to receive an object (Fig. 4.3b) and then continues to reach his original goal describing the path shown in Fig. 4.3c. As shown in Fig. 4.3d the application of a 1st-order estimator predicts and fits (red line) the actual path (cyan points) from the start to the goal, thereby neglecting the turn and other disturbances in this approach. A more complex action like writing on a whiteboard (WB) is shown on the lower-row images. Some words are started to be written in Fig. 4.3e until a change of row is done (Fig. 4.3f). The trace of the hand motions projected onto the interacting background surface (WB) in Fig. 4.3g, could be also modeled by a 1st-order or higher estimator, which in either case it would not describe or exhibit the main purpose of the motions, the action of writing. In order to recognize a certain action a higher-perceptive agent might collect more information like the objects in use (WB, pen, knife, etc.), the locations where the action takes place along with the analysis of the observed motions to find the distinctive features that characterize such action. Our proposed agent decomposes the projected xy-pattern on the WB into three elements: the evolution of the projected x- and y- motion components and the hand's distance-to-WB vs No.frame respectively. In Fig. 4.3h we can observe how the projected x-motion component (red plot) indicates with the ascent of the slope the progress and speed of the writing along the row, which in the end drops corresponding to the returning of the hand to start a new row; in the middle graph we can associate the size variations of the projected y-motions (green plot) to the size of the written letters, i.e., to the long of the hand strokes; the blue plot corresponds to the hand distance to the plane, we can observe that during the writing the hand stays continually close to the WB except during the gap or space between the two written words, which is marked by the high peak of the plot; Additionally we could also infer from this separation the long of the words, being the first shorter than the second one in this case.

In the next section we propose an extension for our presented dynamic 3D model (Chap. 3) that allows a hierarchical labeling of continuous interactions in scenes. While most systems focus on labels for pure transportation tasks, we show how **Atlas** information attached to objects identified in the scene can be used to label not only transportation tasks but also physical interactions, like writing, erasing a board, tightening a screw etc. We analyse the dynamic motion observed by a camera system at different abstraction levels ranging from simple motion primitives, over single physical actions to complete processes. The associated observation time

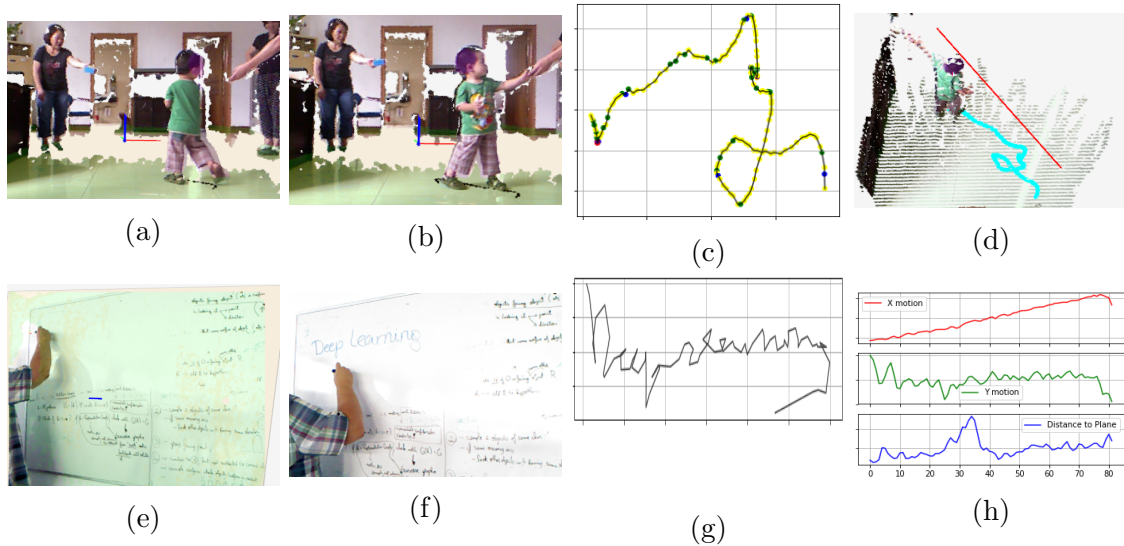


Figure 4.3: Prediction horizons in two different scenes. Upper-row scene: a) a child walks from the right to get a toy at the left side of the image, b) the child turns back to the right to receive another object and then continues his original walk, c) shows the actual path of the child projected on the floor, d) a 1st-order estimator (red line) tracks and fits the actual path (cyan points) and indicates the main goal of the action ignoring the child’s sudden turn and other disturbances. Lower-row scene: e) some words are starting to be written on a WB, f) two words were written and a change to a new row is done, g) shows the projected pattern of the hand motions on the WB, h) our plugin decomposes the projected pattern in x- and y- motion components and the hand’s distance to WB (red, green and blue plots respectively).

horizons can range from single turning motion on the screws tightened during a task over the process of inserting screws to the entire process of building a device. The complexity and the time horizon for possible predictions about actions in the scene increase with the abstraction level. Next we present the extension at the example of typical tasks observed by a camera, like writing and erasing a whiteboard.

4.2 Dynamic Model for Manipulations

Under the principle that the function of an object is meant to be its form, and hence its function determines its shape, one can state that those two attributes are complementary since by knowing complete- or partially one of them we can infer or confirm the other. There are many scenarios, however, in which we know in advance the object but we do not possess the specific information about how such object is intended to be used or handled. It becomes then necessary to disambiguate the possible object functionalities by observing and reasoning about the manipulation motions the object undergoes. In this context we can detect three factors that might help to the identification of actions: **what** object in use is, **how** the handling or manipulation is performed and **where** the action takes place, in descending order of influence. Just by observing a certain object a person recalls a catalogue of a-prior functions or states the object can perform and undergo; all these functions and states, however, might not be active concurrently but detached in time or space. It

is until we observe how the object is employed that we start setting apart the set of functionalities, see Fig. 4.4. Endowing an artificial perception system with such interpretation skill will allow him to take a more active role in areas like human-robot collaboration, elderly-persons monitoring or in security surveillance systems, where the system will be required to identify the task in progress and properly assist it, to supervise the proper use of detected objects, to learn new object functionalities, or to transfer functionalities to unknown but geometrically similar objects.

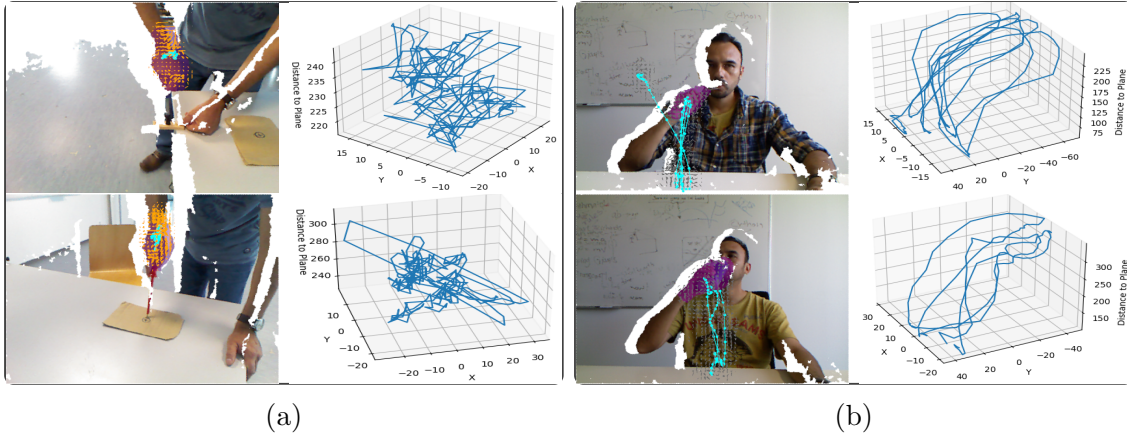


Figure 4.4: Examples of an object with two different functionalities (manipulations) and two objects with similar manipulations (but different actions). a) the same object, a screwdriver, is used with different purposes, for tightening (up) and for poking (down); although the 3D paths of the hand manipulations (at their sides) look different from each other no dominant feature or pattern can be identified at this point to characterize each action out of the chaotic traces. b) Eating (with a fork - up) and drinking (with a glass - down) are similar actions since they describe almost identical 3D traces from table to person’s face back and forth. We can also observe in the images that small or thin objects are almost not perceptible by the sensor; in some cases they are covered totally by the hand.

4.2.1 Action and Activity

In colloquial language we can find the use of both words without any critical distinction, like cooking, driving, eating, etc; In this area, however, it is common to encounter and advisable to distinguish terms like activity, task, action or atomic action. **Atomic Action:** also called *stroke or gesture*, they generally describe fast, short, instantaneous and continuous motion displacements, e.g., a hand twist or lifting an arm, etc. **Action or Task:** it is composed by an ordered sequence of atomic actions, e.g., drinking, writing, etc; normally a single action is associated with a single object. **Activity:** is an (un-) ordered series of actions that are executed to achieve a main goal, e.g., cooking, driving, etc; an activity is commonly associated with the use of multiple objects since it consists of more than one action. In this manner we can now differentiate that *cooking* as activity implies the whole process of preparing a meal: chopping, stirring, salting, etc; *driving*: turning the steering wheel, shifting the gear stick, etc; *eating*: drinking, sipping, chewing, etc. Additionally, activities can be expressed without any object, giving a generic meaning like *waiting*, *writing*,

cleaning, etc; whereas actions can be expressed with objects, giving a more interacting context e.g., *cleaning with a rag the board*, *writing with a pen over the table*, etc; In this work we focus on the latter where, for example, eating as an action will only imply the use of a either spoon or fork to take food from the table to the mouth.

4.2.2 Plugin Scheme

We propose a new method to characterize and model human-object actions based on the observation of the manipulation motions the object undergoes during the interaction. Our method extracts the temporal and cumulative translational manipulation motions and relate them via a geometrical projection with the immediate background, i.e., an interaction plane, which results in a **fingerprint** of the action (e.g., see Fig. 4.3g). With this linked fore-and-background pattern the key motion primitives that are related with the object functionality are correlated upon the background and decomposed in a new set of distinctive manipulation motion profiles as shown in Fig. 4.3h; simultaneously, angular motions of the person’s hand, the active element generating motion, are also kept tracked by means of a dense optical flow estimation (Sec. 3.6.4). Besides the geometry completion (Sec. 3.4), the additional information about functionalities attached to foreground actors can be gained by the utilization of object detectors (Sec. 3.6) and **Atlas** (Sec. 4.1.1). The obtained results in our model show that our approach is able to disambiguate the functionalities of the same object as well as to differentiate similar actions executed with different objects (Fig. 4.4); This allows to boost the set of observed motions to a higher level of understanding, namely as a task or action. A scheme of the agent coupled to the framework is shown in Fig. 4.5.

The information pipeline starts with the reception of 2/3D imagery data from the sensor Kinect-1 [®], the foreground actor and interacting background structure are defined, and the motion evolution of the hand (the principal active element in the scene) is observed and registered; All this information that belongs to the current, local environment is collected by the Knowledge Containers and linked with the Atlas. The current extension takes the required information from these elements; it projects the captured motion onto the background plane to obtain the pattern of the manipulation i.e., the fingerprint of the action, and encapsulates it to finally decompose it into a new set of motion components.

4.2.3 Fore- and Background Interaction Actors

We understand as **foreground** those active, dynamic 3D structures in the scene that are able to move or generate motion in other structures; in this plugin we refer in particular to the person’s hand grasping and manipulating objects. Many objects are designed to be handled by one hand, therefore they are partial or totally covered once they are grasped; however, the shape of a hand holding an object does not change abruptly and its shape is simpler to model than rather to keep a complex 3D model of an object for segmentation and tracking.

The 3D segmentation of the hand is initialized by OpenPose detection of the hand or by manually defining the area where the hand lies in the 2D image (Sec. 3.6.3), after that the 3D hand blob is detected by OpenPose or a visual tracker and segmented at each image frame. Starting from a pixel-point (**px-pt**) seed we employ the Breadth-

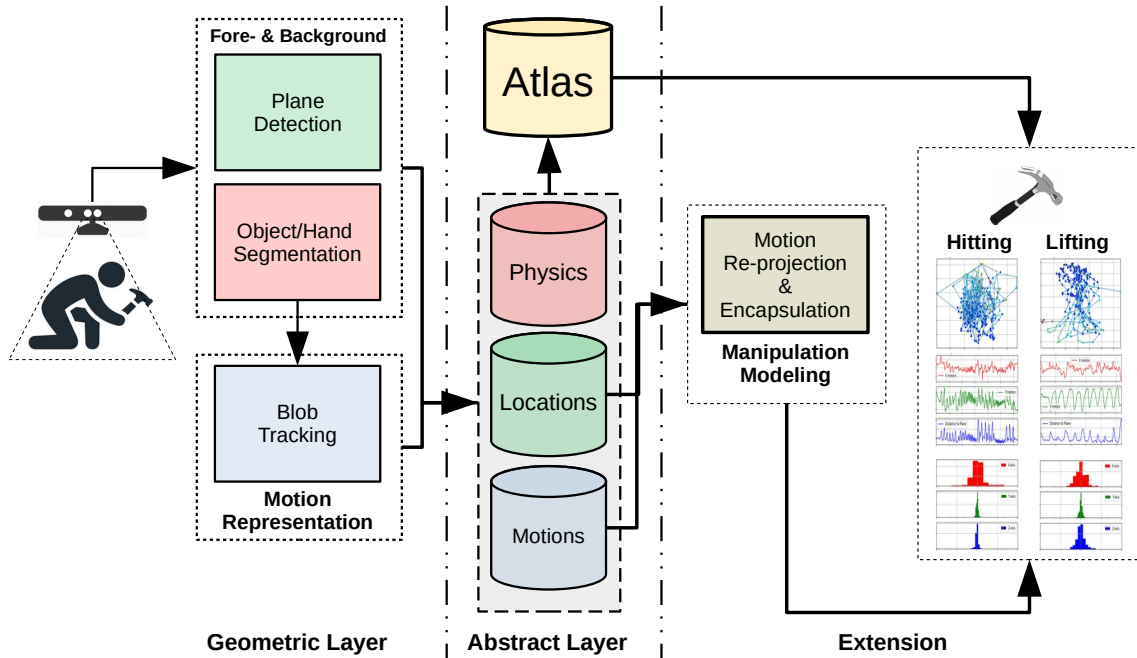


Figure 4.5: Plugin Scheme for dynamic model of object manipulations. The extension coupled to the framework’s geometric layer takes from the Knowledge Container the required data to obtain a new set of motion profiles characterizing the object manipulation.

First Search (BFS) algorithm [18] (Sec. 3.2.2) to connect 2D neighbor pixels under 3D conditions, i.e., each pixel must have a valid depth value and the distance between neighbor pixels must be less than a fixed threshold. In Fig. 4.6 fore- and background detection and segmentation of some experimental scenes presented in Sec. 6.1 are shown. The scene background in these scenarios is modeled as the 3D plane closest to the manipulation action since this element serves either as a supporting structure and the action is directly executed over it (Sec. 3.2.3).

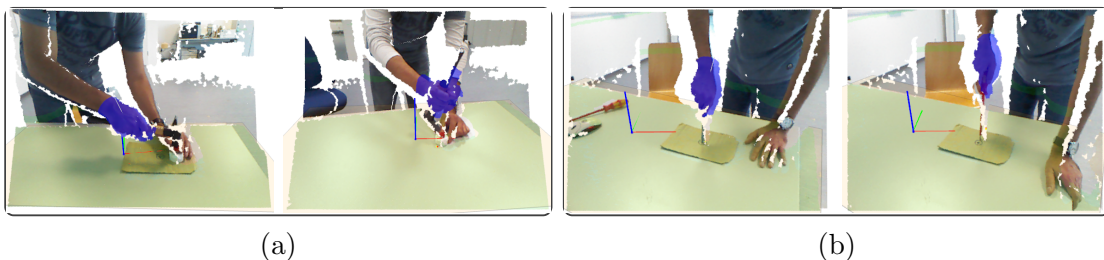


Figure 4.6: Detection and segmentation of fore- and background actors in some experimental scenarios. a) Hitting and pulling with a hammer, b) tightening and poking with a screwdriver. The segmented hand is highlighted in purple, the interaction background in green.

4.2.4 Motion Representation

To build up a path-based motion representation of the hand motions we utilize either our augmented Openpose skeleton-hand detection or our 3D Kalman filter [55] tuned with added white-noise acceleration model as a visual tracker to estimate the hand

position at each frame. In addition to this translational motion we also aim to perceive cues of rotational motions of the hand. This is obtained by estimating the changes on the 3D optical flow direction inside segmented hand areas between consecutive image frames. Considering that the hand covers in general a small area inside the visual environment and consequently it does not offer a large body surface for a sparse optical flow estimate [71] we employ the Farnebäck method [31] for a dense optical flow. In Fig. 3.28 we can observe the difference between the sparse and dense 2D optical flow in some examples. Fig. 4.7 shows some 3D KF path estimations and 2D optical flow detections in some exemplary scenarios.

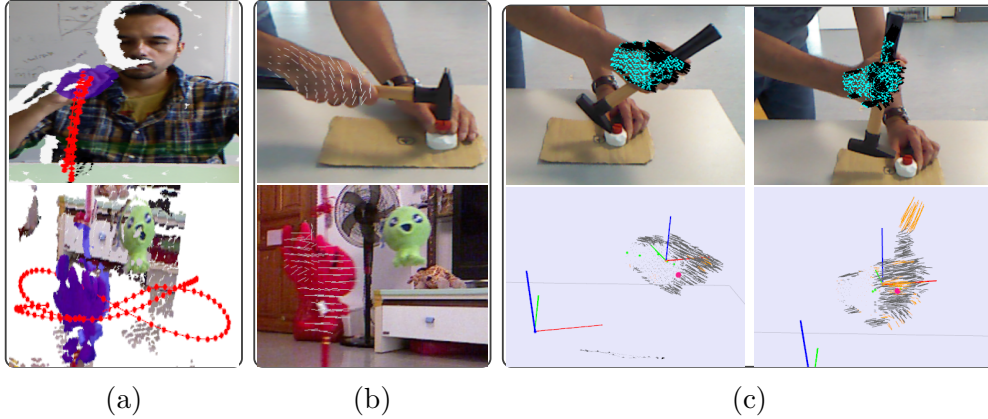


Figure 4.7: Motion representation: 3D path and rotation angles. a) shows the obtained 3D paths (red points) of a segmented hand (eating with a fork) and a hanging toy moving freely (both in purple); b) shows the detected optical-flow vectors (white lines) of a hand (hitting with a hammer) and the hanging toy; c) Scenes of pulling/hitting with a hammer: (up) 2D optical-flow detection over segmented hands and (down) mapped to their corresponding 3D flow vector inliers (black) and outliers (orange)

In order to determine the amount of angular motion exerted by the hand the detected 2D optical flow between two consecutive frames at time (k) and ($k + 1$) is mapped to their corresponding 3D vectors and a coordinate frame $\{H\}$ is initialized at the segmented 3D hand centroid at frame time (k), and whose initial ${}^H z$ -axis is aligned with the normal vector of the background surface (see Fig.4.7c). The 3D flow vectors are then projected onto the planes defined by each hand frame axis, i.e., onto the yz -plane for the ${}^H x$ -axis, onto the xz -plane for the ${}^H y$ -axis and onto the xy -plane for the ${}^H z$ -axis (see Fig.4.8); the angular displacement for each axis is estimated for each projected set:

$$\Sigma^2 = \sum_{l=1}^L \|\mathbf{f}(k+1)_l - ({}^{flow}\mathbf{R} \cdot \mathbf{f}(k)_l + {}^{flow}\vec{\mathbf{t}})\| \quad (4.1)$$

where L is the number of flow vectors, $f(k)_l$ and $f(k+1)_l$ are respectively the start and ending points of the l flow vector between consecutive frames (k) and ($k+1$). We change the rotation matrix \mathbf{R} to the axis-angle representation following [72]:

$$\theta = \|\mathbf{v}\| = \arccos\left(\frac{\text{trace}(\mathbf{R}) - 1}{2}\right) \quad (4.2)$$

with which we can obtain the vector representation of the motion parameters:

$$\hat{\mathbf{v}} = \frac{\vec{\mathbf{v}}}{\|\mathbf{v}\|} = \frac{1}{2 \sin \theta} (\mathbf{R}_{32} - \mathbf{R}_{23}, \mathbf{R}_{13} - \mathbf{R}_{31}, \mathbf{R}_{21} - \mathbf{R}_{12}) \quad (4.3)$$

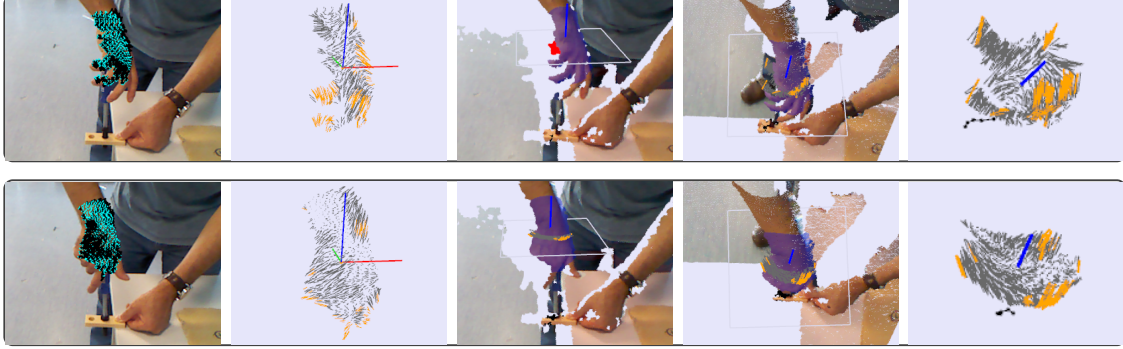


Figure 4.8: Projection of 3D OF vectors onto the plane defined by ${}^H z$ -axis at hand centroid for estimation of hand rotations between two frames during 'tightening with a screwdriver'. (From left, for each row) detected 2D OF over the hand are mapped to find their corresponding 3D vectors; two different views of the augmented plane at hand centroid are shown; finally a close-up view of the projected 3D OF vector inliers (black) and outliers (orange) onto the plane is shown.

4.2.5 Manipulation Modelling

The motion representation we obtained until now can be described w.r.t. some arbitrary frame e.g., camera frame $\{C\}$ or world frame $\{W\}$, which in any case do not directly expose any relation between the fore- and background interaction.

In order to extract the distinctive traits of the manipulation motions and to link them with the background we project the translation movements of the hand centroid onto the interaction plane and at the same time record its distance to this plane; following [100] we apply Eq. 4.4 and Eq. 4.5:

$$\mathbf{c}' = \mathbf{c} - (\mathbf{c} \cdot \hat{\mathbf{n}} + d) \hat{\mathbf{n}} \quad (4.4)$$

where $\|\hat{\mathbf{n}}\| = 1$, \mathbf{c} is the hand centroid and \mathbf{c}' is its projection onto the plane,

$$r = \frac{\mathbf{c} \cdot \vec{\mathbf{n}} + d}{\|\vec{\mathbf{n}}\|} \quad (4.5)$$

being r the perpendicular distance from the plane to the hand centroid, see Fig. 4.9 and Fig.4.10.

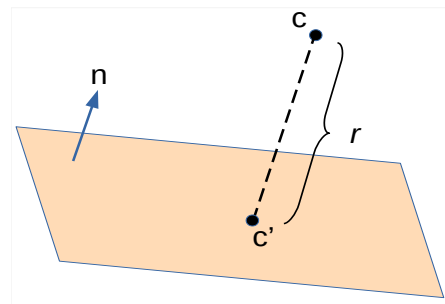


Figure 4.9: 3D geometric projection c' of a point c onto a plane defined by the normal vector $\vec{\mathbf{n}}$.

The projected motion path is next framed or encapsulated in 2D; the purpose of this encapsulation is to set some correlation of the projected motion directly upon the

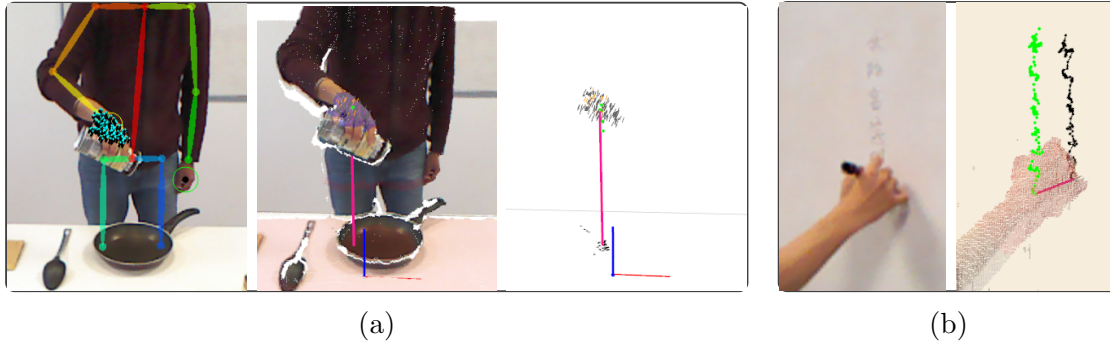


Figure 4.10: Hand centroid projections onto interaction planes. The red line in each scene (a, b) represents the projection vector parallel to the normal plane and whose magnitude gives the distance ' r ' between the current hand centroid position and the plane. a) 2/3D images and hand OF with projection vector of '*pouring on a fry pan*', b) '*writing on the whiteboard*', 3D hand centroids (green) are projected (black points) onto the board plane.

interaction background, where actually the action takes place. Such encapsulation can be performed under the PCA procedure [102, 107], in which the resulting major and minor $^{PCA}(x, y)$ axes are basically determined by the distribution of the data i.e., the scattering of the projected points; Additionally, the encapsulation axes can be alternatively defined in advance according to a fixed scene context or feature that might be more in *direct* relation with the manipulation, like the chest or face of the person, the active arm's axis up, etc; e.g., aligning the encapsulation x-axis to the person's chest axis or setting up the encapsulation y-axis parallel to the arm's axis; giving in this way a more semantic meaning to the encapsulation. Besides the PCA, in Fig.4.11 we show the PLN encapsulation, in which the x- and y-axis of the interaction plane are used as reference for this procedure. With the projection we obtain a *fingerprint* as a characteristic pattern of the interaction, which in turn is decomposed in its $^{Encap}(x, y)$ motion components to constitute along with the hand distances (to the plane) our new set of motion profiles that characterizes the action. In Fig. 4.12 we show some examples of the encapsulation, projected xy-pattern and the hand motion profiles obtained during '*writing-on-a-whiteboard*' scene, and also the distributions of measured and aggregated hand rotations obtained during the '*tightening-with-a-screwdriver*' scene.

In Chap.6 we present the results of our proposed motion modeling applied to different manipulation scenes in where each employed object is used with two different purposes i.e., two different functions; we also present scenes where the manipulation motions are similar in spite of handling different objects. The model is able in each case to characterize and distinguish each task or action.

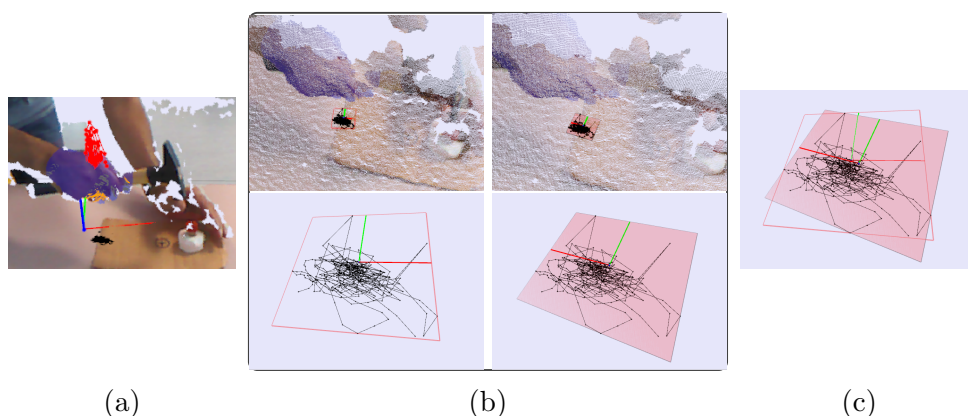


Figure 4.11: Examples of PCA and PLN encapsulations. a) shows as reference the 3D-RGB image of the segmented hand (purple), the motion path (red points) and the projected path onto the table (black points), b) shows the 3D-RGB upper-view (up) and their corresponding zoomed-in (down) images of the PLN (left) and PCA (right) encapsulations. While the resulted PCA encapsulation axes are determined by the 2D scattering of the projected points, in the PLN encapsulation the axes are defined in advance to be aligned with the background plane axes; with this, the projected motions could be geometric and semantically related with the edges of the supporting plane, c) shows the 3D overlapped PLN-PCA encapsulations.

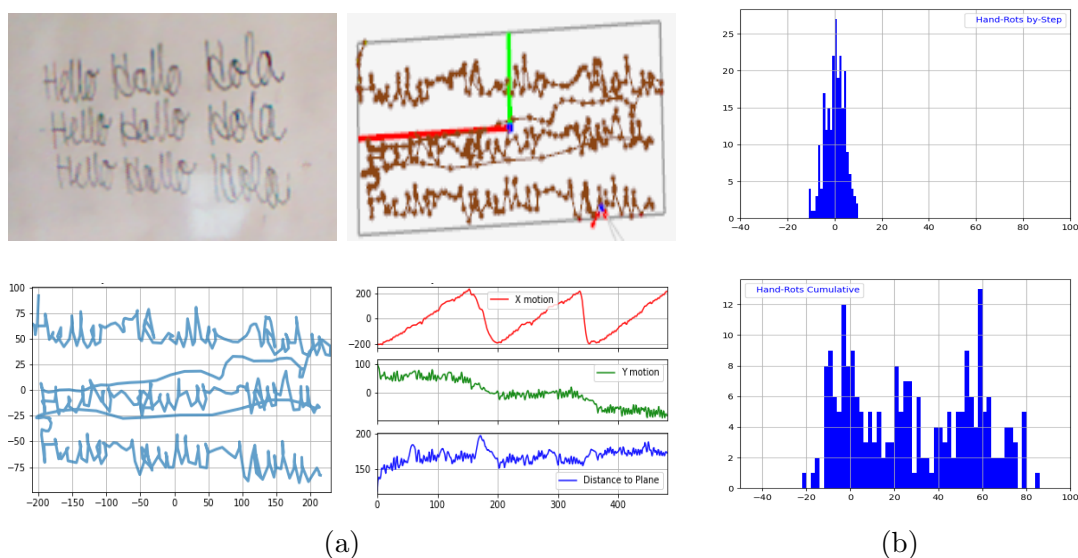


Figure 4.12: Manipulation Modelling: PCA encapsulation, motion and histograms profiles of projected manipulation patterns. a) Writing on a whiteboard "Hello Hallo Hola" (up-left), the PCA encapsulation was obtained (up-right); the xy-pattern or fingerprint of the action along with the decomposed motion profiles are shown in the lower row; b) Histogram profiles of hand angle rotations by-step (up) and cumulative (down) about a plane-normal vector corresponding to the scene 'tightening with a screwdriver'; more detailed analyses of these two actions are presented in Chap.6.

4.3 Dynamic Model for Autonomous Driving

The **ROboMObil** (RoMo)¹ (Fig.4.13a), is a driven-by-wire vehicle prototype designed and built at the **Center for Robotics and Mechatronics (RMC)** of the **Deutsches Zentrum for Luft- und Raumfahrt (DLR)** for research, development and tests of new technological approaches in several subjects of engineering and science like Human-Machine interaction, vehicle simulation and dynamics control, wheel slip control and torque blending, vertical dynamics, path following control, autonomous driving, car2x communication, ergonomics. Its high maneuverability is based on the individual high angle range and independent steering control of its four identical wheel robots, and whose operation is regulated and coordinated by an intelligent central unit. These features allow extended maneuvers like sideways driving, 360° rotations around its vertical axis or augmented zig-zag maneuvers where all four wheels are turned and parallel aligned at once. Additionally, RoMo can be fully or partially operated in both manual or autonomous modes.

4.3.1 Perception System

The RoMo perception system to support autonomous driving was conceptualized from the beginning to consist exclusively in cameras adapted to the chassis structure to cover a 360° view and 3D reconstruction of the surroundings. The RoMo vision system is composed of 18 cameras in total; As shown in Fig.4.13b the relevant camera arrangement is placed on the vehicle deck, where six stereo camera rigs are setup: two camera pairs with overlapping visual areas are set to each side, one more pair to cover the rear part and one more at the front; a set of three cameras aligned horizontally were also installed on the front (below the windshield), see Fig.4.13c, and also on the rear side (at the lower part of the back glass), see Fig. 4.13d. Additionally, ultrasound sensors were also installed at the rear and front bumper areas to perceive immediate close obstacles unseeable to the cameras, see Fig. 4.13d. A top view of the areas covered by the stereo camera layout can be seen in Fig. 4.13e.

The employed cameras are of type Prosilica GC780(C) with Ethernet interface and image resolution of 782(H) x 582(V) pixels; only the front cameras supply color images with color digitization of 3 x 8 bit and 12 bit for the rest of monochrome cameras. For the 3D stereo reconstruction of the point cloud the **Semi Global Matching (SGM)** algorithm [47, 48] is applied and computed with the support of parallel processing hardware like GPU's (Graphics Processing Unit) or FPGA's (Field-Programmable Gate Array). The software application used in the management and communication of the cameras as well as in the camera data distribution (data streams) is SensorNet [8], which is a middleware software developed and maintained at DLR for the distribution of sensor data in parallel applications.

4.3.2 Segmentation and Mapping

The segmentation and mapping previously described in this work were adapted and applied to the data supplied by the RoMo vision system to contribute to its autonomy, in particular, in an autonomous parking scenario. Unlike the indoor settings

¹<https://www.dlr.de/sr/desktopdefault.aspx/tabid-11633/>

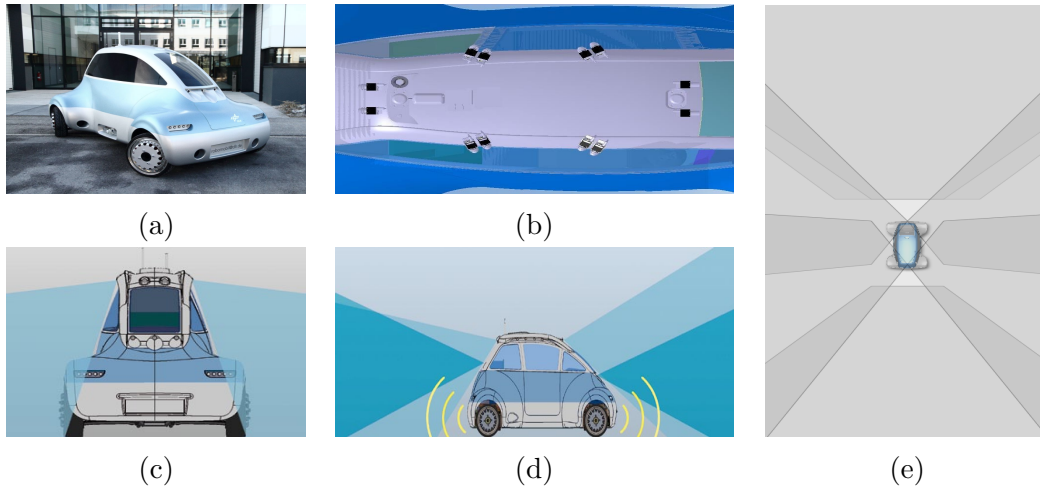


Figure 4.13: The ROboMObil (RoMo). a) RoMo is a test platform vehicle designed and built at DLR, b) location of the six stereo camera (dark squares) rigs arranged on the vehicle deck: two stereo rigs at each side (up and down in the image), one rig on the deck front (right in the image) and one more on the deck back (left in the image), c) the front face shows the two front cameras on the deck and the three additional cameras below the windshield; the light-blue areas show the visual scope of the side cameras, d) the light-blue areas indicate the scope of the front and back cameras on the deck, while the dark-blue areas mark the scope of the additional cameras below the windshield and at rear side; the yellow curves indicate the ultrasound sensors installed at the bumper areas. e) shows the general top view of the areas covered by the stereo cameras, i.e., areas of 3D stereo reconstruction.

presented in this work so far we are now dealing with an outdoor parking scenario where the average size of the objects are now potentially much larger; therefore, we are not interested in obtaining refined, precise or complete 3D reconstructions of segmented 3D blobs but rather in guesstimating their coarse occupancy geometries and behaviors (dynamics) since these factors affect and interfere more direct and drastically the autonomy strategies of the vehicle; e.g., occupancy is more relevant in a parking scene and blob behaviors are more relevant in obstacle avoidance maneuvers. As shown in Fig. 4.14 we firstly collect and align the individual 3D reconstruction of each stereo rig to obtain a single 360° rigid point cloud and break it down into a set of segmented entities that might represent individual objects in the world; the benefits of the segmentation in this application are, first to obtain a gross impression on how the immediate surroundings is structured; second, each segmented entity might have its own semantic meaning i.e., own functions and behavior. Additionally, for each 360° reconstructed 3D cloud we assume a flat terrain that is perpendicular to the RoMo's vertical axis and whose 3D points lying in this plane or below are removed from the segmentation; we found the proximity connections of the remaining 3D points by means of the DFS algorithm (Sec. 3.2.2) to cluster each point with its closest neighbors; each segmented 3D blob is then encapsulated into a box whose dimensions are determined by the PCA method (Sec. 3.2.4). The final outcome is a set of encapsulated 3D blobs representing potential active actors as shown in Fig. 4.14b, and whose dimensions and poses have been already determined and registered. The potential functions and behaviors of each segmented blob can

be inferred by means of additional routines or applications for object recognition and detection (furniture, cars, traffic signs, etc.), as presented in Sec. 3.6.

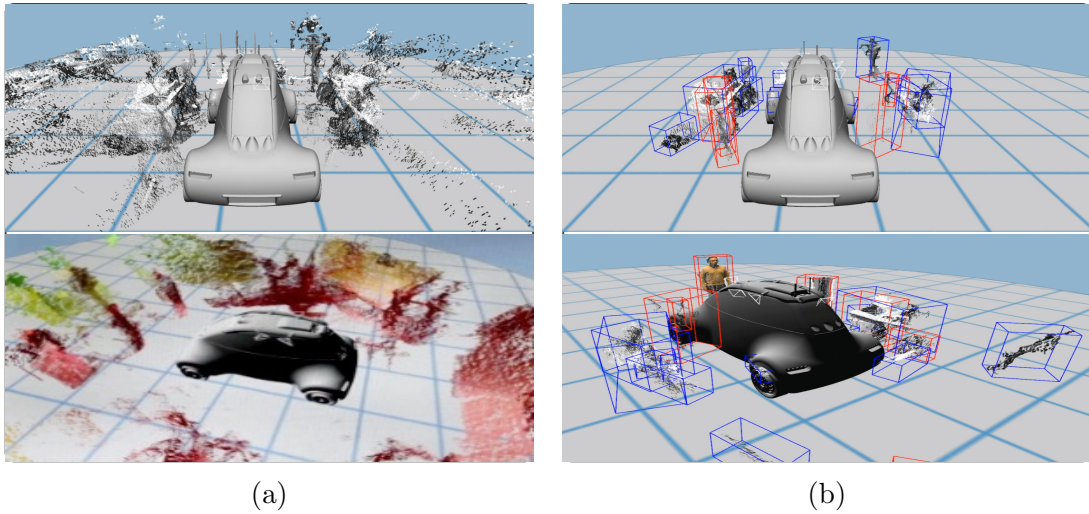


Figure 4.14: 360° stereo reconstruction and 3D segmentation around RoMo. a) shows the 360° collated stereo reconstruction around RoMo, the lower image shows a color-coded distance rendering of the point cloud (red points are closer), b) sets of encapsulated blobs.

4.3.3 Autonomous Parking Maneuver

In order to show the RoMo maneuverability the parking scenario shown in Fig. 4.15 was set. The images in Fig. 4.15a from top to bottom show the 2D image, the 3D reconstruction and 3D blob segmentation rendering of the parking scene captured from the RoMo front camera; The images in Fig. 4.15b illustrate the three autonomous parking stages, from top: the approaching step, where RoMo moves forward to a estimated spot outside the parking box (yellow arrow and circle in Fig. 4.15a-bottom) and centered between the two other parked cars; in the second stage RoMo turns around the reached point in order to be parallel aligned with the neighbor cars (blue turning arrow in Fig. 4.15a-bottom); in the last step, Fig. 4.15b-bottom, RoMo drives sideways into the parking box. More information and details about the presented parking approach can be found in [99].

4.3.4 Pedestrian Segmentation and Path Prediction

RoMo as a unique test-bed platform for diverse engineering approaches and in its road to achieve fully autonomy we also explore and test the segmentation and mapping of dynamic elements in the scene, namely pedestrians. We follow the segmentation steps as explained before in Sec. 4.3.2 and each clustered blob is encapsulated into a vertical, rectangular cuboid parallel to the terrain normal. In Fig. 4.16a and Fig. 4.16b we can observe that the height of each encapsulation cuboid remains almost constant while its length and width present abruptly and constant size changes mainly due to pedestrian's extended arms or separated legs during walking. Therefore, obtaining a precise 3D representation of each encapsulated pedestrian is not

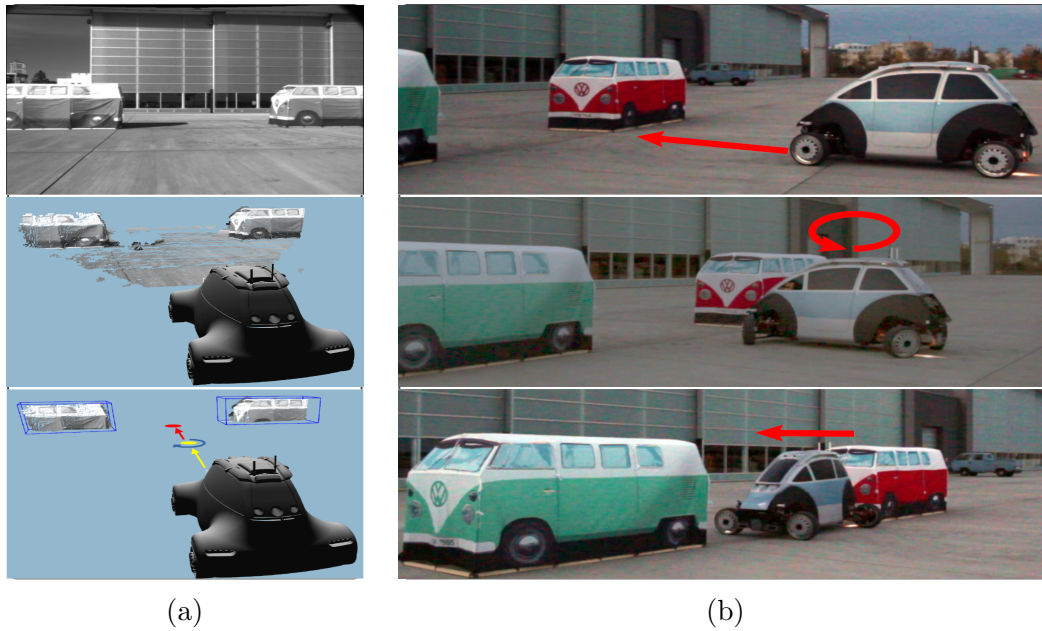


Figure 4.15: RoMo's Autonomous Parking Maneuver. a) shows front top: 2D image, 3D reconstruction and 3D segmentation renderings of the parking scene from the RoMo front camera; the bottom image also indicates with the yellow, blue and red signs the parking steps to follow; b) shows the three stages of the autonomous parking maneuver, from top: approaching, turning and alignment, and the sideways driving to the parking box.

the main concern in this outdoor scenario, it is rather important to describe and predict the walking path they exhibit. In the images of Fig. 4.16 we can also see the path trails the pedestrian blobs have left; the registered trails help us to predict and correct at each frame the probable path a pedestrian can follow. Unlike a full 6-DOF (Degrees Of Freedom) pose detection and tracking of an object like a hand that can move (or be moved) arbitrary in 3D with sudden changes of directions plus three more independent rotation axes: roll, pitch and yaw, a pedestrian motion exhibits in general 3-DOF since he/she is confined to move on the ground (xy-plane, 2-DOF) plus an additional rotation about his/her vertical axis (1-DOF) and whose estimate could contribute to detect unexpected path changes. Pedestrian behavior is a wide topic extensively researched in the community under a lot of different scenarios, in crowded or normal out- and indoor places, like stations, airports, corridors, sidewalks, etc; Regarding the pedestrian motion and behavior, we based our path estimates on two assumptions: i) a person walks to reach a fixed point, this point is in general not changed abruptly, ii) the person tries to reach the goal as direct and fast as he/she can, commonly describing a straight line and avoiding large deviations along it. The estimates of the predicted paths are performed utilizing only the first 2-DOF, i.e., xy-motions based on the collected trail points and updated with every newly observed position under a recursive least square procedure [108] [118]. In Fig. 4.16c we show some images of the RoMo's interface for 2/3D visualizations, where the predicted path for each encapsulated pedestrian in 3D (left, blue background) and for their corresponding tokens in the 2D mapping (right) is estimated and delineated. The top image of Fig. 4.16c shows the ideal case where

each pedestrian is individually identified and tracked; the middle image shows a total occlusion in which a pedestrian blob (green) blocks completely the detection of another one (red) becoming the latter not detectable during some frames inside the sensor scope; in the bottom image we can see a partial occlusion: the partially blocked blob (red) and the blocking one (green) are merged into a single larger blob (black box) for some frames.

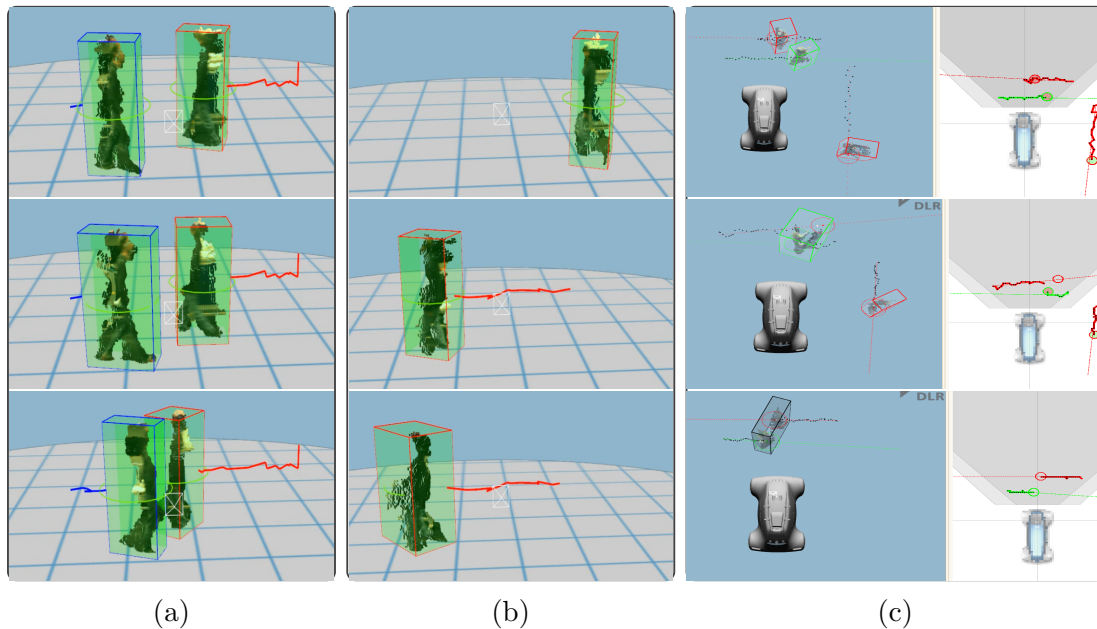


Figure 4.16: RoMo's interface for visualizations of 3D Pedestrian segmentation, path prediction and 2D mapping. a, b) show some 3D segmented and encapsulated pedestrians along with the trailing paths. We can observe the changes on the length and width of the rectangular cuboids according to the pedestrian shape; c) shows the 3D (left, blue background) and 2D mapping (right) visualizations of the RoMo interface, where the estimated path for each pedestrian is delineated (thin line; black points in 3D or thick lines in 2D rendering indicate the blob trails): (top) ideal case where the pedestrians are detected and segmented individually, (middle) shows a total occlusion: a pedestrian blob (red) is fully blocked by another one (green), (bottom) presents a partial occlusion, where the blocking and partially blocked blobs are merged into a single one (black box); as shown, in all these cases the predicted paths are kept, and updated with new observations.

Chapter 5

Results - Geometry Layer

Sections

Overview	81
5.1 Sensor Setups	83
5.1.1 Stereo Cameras	83
5.1.2 RGB-D Sensor	83
5.2 Octree-DS Performance	84
5.2.1 Building and Storing Times	85
5.2.2 3D Scene Mapping Test	85
5.3 3D Object Completion	87
5.3.1 Z-Buffered Fusion & Confidence Value	87
5.3.2 Blob State Propagation under Partial Visibility	88
5.4 Visual Estimation of Independent Foreground Motions	90
5.4.1 Ego- & Independent-Motion in Non-Static Scenes	90
5.4.2 3D Object Completion in Non-Static Scenes	91
Summary	93

Overview

In Chap.3 we presented the methods, procedures and implementations that we apply to organize the geometrical information inside a scene; we started by breaking down the rigid 3D point cloud of an observed scene into two main components: fore- and background, where the former constitutes a set of tentative object candidates, each with potential functionalities and attributes, and the latter corresponds to larger bodies representing (pieces of) furniture, walls, doors, etc. Both of these regions are further processed at geometric level e.g., segmentation, clustering, location areas, etc; to obtain a consistent world map. Besides the structural aspects we also went over the kinematics of the scene in non-static and dynamic environments; we presented the procedures for ego- and independent-motion estimates as well as the implemented tools for hand detection and motion tracking, which is a required, relevant information at the higher layer of the framework.

In this chapter we introduce some practical aspects and specifics regarding the implementations, devices and libraries that we consider are directly related with the core operation of the framework at the lower layer; at the sensor level we describe the sensor setups we employed; on the Geometric-Layer side we test and evaluate the behaviour of our octree-based mapping implementation. By means of some exemplary scenes we also run experiments for the evaluation of the geometric-layer methods and approaches presented in Chap.3; two implementation results are presented in this chapter corresponding to: *i*) 3D object completion, in which the data association and fusion of encapsulated 3D blobs is conducted by our proposed z-buffered re-projection approach and a confidence value is assigned to each fused-blob point, *ii*) independent-motion estimations of foreground structures in non-static scenes: ego- and independent-motions.

5.1 Sensor Setups

5.1.1 Stereo Cameras

Our stereo camera rig is constituted by two paired Guppy F-080C IRF (Fig. 5.1) cameras with a baseline of 8cm delivering images of size 640 by 480 pixels.

In order to obtain disparity map images (e.g., Fig. 5.2b) and 3D point cloud reconstructions (Fig. 5.2c) we made use of the XVISION-2 library [41]; this library allowed us to adjust and customize the 3D point reconstruction by accessing and setting some camera internal parameters like gain, exposure, shutter, white balance, etc; The intrinsic camera parameters as well as the extrinsic stereo calibration parameters were obtained with the `Matlab`¹ calibration toolbox. The first set of experiments presented in this chapter was executed using our stereo camera setup; in Fig.5.2 we show some exemplary images that can be obtained at each step during the 3D map building process and object segmentation. Fig.5.2d shows a snapshot of our first online detector of 3D object candidates. At the same time we can also observe how sensitive, and hence noisy, the 3D point reconstruction can be to spurious reflections of light coming from bright backgrounds or directly from shining object surfaces; these issues can be resolved at sensor level with the utilization of other sensor technologies like RGB-Depth sensors.



Figure 5.1: Guppy F-080C camera.

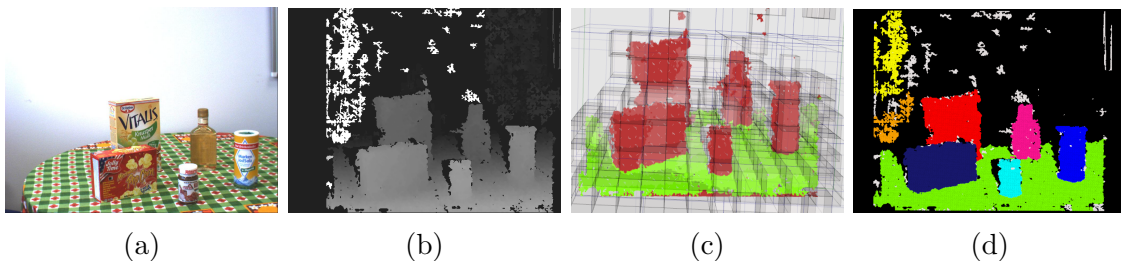


Figure 5.2: Stereo camera images of 3D object segmentation process. a) 2D RGB scene, b) disparity map obtained with XVISION-2 library, c) fore- and background structures in octmap, and d) snapshot of our first online detector for labeling of 3D segmented object candidates; note how a group of noisy reflected points can be also clustered.

5.1.2 RGB-D Sensor

As already mentioned in Sec.2.2 the RGB-D sensor we occupied is the Kinect-1; we employ it principally for the recording and analysis of object-manipulation scenes at the higher layer of the framework described in Sec.4.2, and whose experiments are presented in the next Chap.6. The middleware for accessing this device is the

¹www.mathworks.com

open source library `libfreenect`², built and running on linux; Although we can obtain RGB, Depth and IR images all of size 640x480 [px] as described in Sec.2.2 (and shown in Fig.2.4) Kinect-1 does not supply a compound RGB-Depth image, i.e., a textured/colored 3D point cloud; moreover, since RGB and IR cameras are assembled with a baseline of approx. 2cm in a typical stereo camera layout, plus the difference on focal-length between them, the raw data buffers coming from them are displaced i.e., RGB and Depth buffers do not correspond one to one. As described in Sec.2.2 the warping adjustments (Eqs.2.4,2.5,2.6) between these images are carried out and shown in Fig.5.3.



Figure 5.3: Kinect RGB-Depth overlapped and warped images. Basically due to the difference in focal lengths ($f_{IR} > f_{RGB}$) the RGB and Depth image data do not fit as can be observed on the RGB-Depth overlapped image (right), where the cyan-colored areas of non-valid IR-Depth values do not correspond to the object shapes of the RGB image. In the 1st adjustment, warped-Depth projected onto RGB (middle), we take the RGB as base and *distort* the Depth to fit the RGB image, observe that the resulting image is cropped to cover only the area overlapped by both images; In the 2nd correction, warped-RGB projected onto Depth (right), the base is the Depth image in which the *warped* texture of the RGB image is projected.

In Fig.5.4 besides the wrapped (Fig.5.4a) and non-wrapped (Fig.5.4b) RGB-D image corresponding to some cooking scene tasks, we also show at this geometric level some other exemplary images that can be obtained with this sensor and that help the processing and analysis of scenes in the framework;

5.2 Octree-DS Performance

In computer science a Data Structure (DS) is an abstract entity that helps to organize information inside a computer memory; depending on the type of DS sometimes it is required for its implementation to reserve or allocate a certain amount of memory, and also a set of procedures have to be devised and provided to perform the required DS functionalities, like data storage, data access, DS maintenance, etc; on the other hand, independently on its type or organizational structure a key parameter to quantify a DS performance is its computational complexity which is in general measured based on specific DS operations, like DS building/traversing, data inserting/deleting, etc. In this context an octree performance is typically described by the plots shown in Fig. 3.4

²<https://openkinect.org>

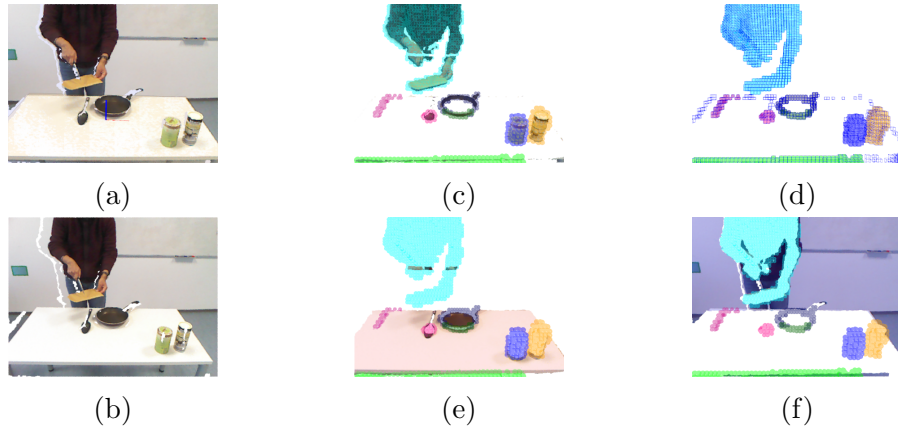


Figure 5.4: Exemplary RGB-D images for processing and analysis of scenes. a, b) Warped and non-warped RGB-D image, note how well the Depth-silhouettes fit the contours of RGB-data (objects), c) raw 3D segmentation and clustering of foreground regions, d) voxel-occupancy of 3D segmentation, e) 3D segmentation and supporting plane, and f) 3D segmentation and points $>$ z-depth (e.g., wall).

5.2.1 Building and Storing Times

In order to analyze and characterize the behavior of our particular octree-DS implementation first we obtained its computational performance regarding the building and storing times; For the building time we tested the DS with three different sizes of point sets k at different number of nodes, as we can observe the curves in Fig.5.5a the building behaviour resemble those presented in Fig. 3.4 (Sec.3.2.2). The corresponding obtained storing-time performance tested also at different voxel sizes is shown in Fig.5.5b.

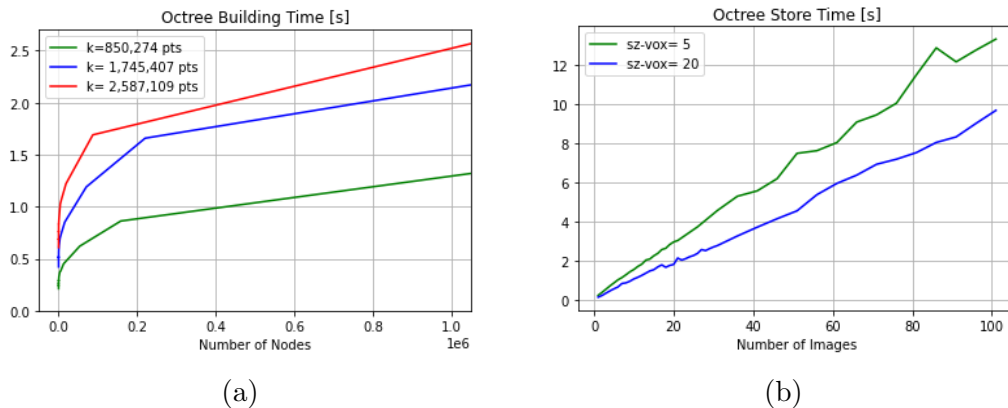


Figure 5.5: Building and storing performance of implemented octree DS. a) Octree building times in function of the number of nodes with three different data sizes, b) octree storing times in function of the number of images at two different voxel sizes.

5.2.2 3D Scene Mapping Test

In a second experiment we want to test the DS' behaviour and to know how some of its parameters alter when mapping a 3D scene. For this we employ the scenes

shown in Fig. 5.6 corresponding to an indoor scenery. Table 5.1 shows the results of collating the first eight (of nine) 3D scene images of Fig. 5.6a and building an octree with this data at different voxel resolutions. Since the 3D scene images overlap, as shown in the collated scene in Fig. 5.6b, many points of newly collated images might not generate neither new nodes nor leaves. We can observe in the table, as expected, the octree building time, the number of nodes and the number of leaves decrease as the size of the voxels (leaves) increases, as shown graphically by the curves in Fig. 5.7. We can also observe that for some consecutive and multiple voxel sizes the cubical size and volume of the octree root node remain the same.



Figure 5.6: Exemplary mapping sequence for characterization of octree-DS performance. a) Individual 3D scenes, b) collated scene showing the camera poses from where each individual scene image in (a) was taken.

Table 5.1: Test Results of Octree-DS mapping sequence at different voxel sizes: building-time, number of nodes/leaves, size/volume requirements at different voxel sizes.

For 8 captured images with a total of 2,292,618 pts					
Voxel size [mm]	# Nodes	# Leaves	Root size [m]	Root volume [m ³]	Building Time [s]
1	2,325,222	2,209,382	4.096	68.7195	3.487930
2	266,195	899,111	6.144	231.928	2.215324
5	83,934	317,872	5.12	134.218	1.557559
10	18,182	65,687	5.12	134.218	1.106374
15	7,667	26,362	7.68	452.985	0.952324
20	4,262	13,787	5.12	134.218	0.919194
40	1,072	3,197	5.12	134.218	0.807496
80	288	787	5.12	134.218	0.731326

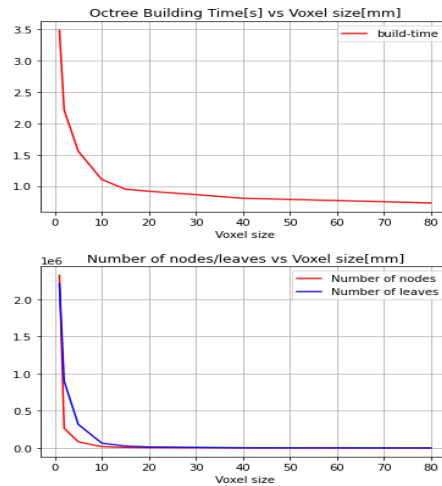


Figure 5.7: Building time and number of nodes/leaves vs voxel size for the mapping test of images in Fig.5.6 and Table.5.1.

5.3 3D Object Completion

5.3.1 Z-Buffered Fusion & Confidence Value

For the test and evaluation of the proposed blob fusion in Sec.3.4 based on the z-buffered re-projection approach and a confidence value (CV) assignments to each fused-blob point we present the scene shown in Fig.5.8a; we took a sequence of observations at different camera poses around this scene to obtain the 3D segmented fore- and background at each captured observation as shown in Fig.5.8d. The final blob map in Fig.5.8b shows the final fused blobs, in which each point has assigned a color-coded CV from 0 to 7 corresponding to : white, yellow, light brown, orange, green, violet, purple and red, respectively. Close-up views of two of these fused blobs is shown in Fig.5.8c. As we can observe in Fig.5.8c the points colored with lower CV

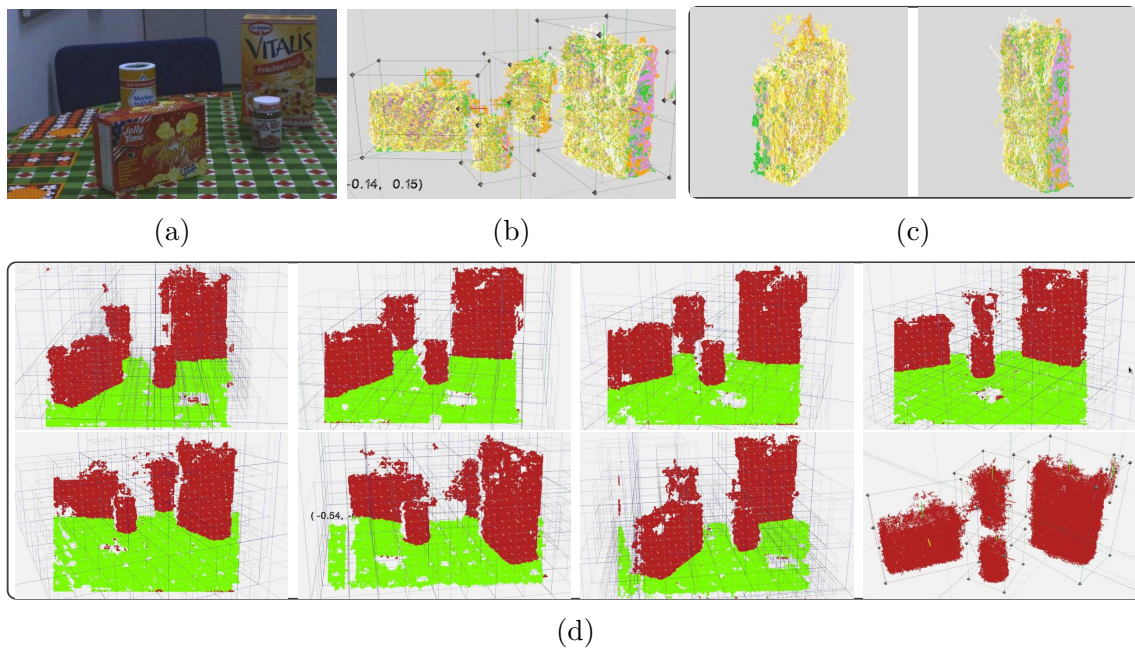


Figure 5.8: Scene sequence for 3D object completion and confidence value (CV) assignments. a) 2D RGB image of scene, b) 3D final map of encapsulated and fused blobs with color-coded CV, c) 3D close-up views of two fused blobs corresponding to the object boxes at the scene sides (left) pop corn box, (right) cereal box; d) sequence of registered and segmented 3D fore- and background at different camera poses that are to be fused.

are or must be assigned to noisy or rather more 3D *superficial* points to the blob hull; In a first evaluation we are interested to determine how precise the points with different CV assignments describe the actual size of a fused object. We present the results for the two boxes of the scene presented in Fig. 5.8c, which are the objects whose blob point readings undergo the widest range of error observations due to the variations in the proximity of the objects to the cameras as these moved around the scene. In tables 5.2 and 5.3 we summarize the results obtained with the valuated points of the cereal box blob and pop-corn box blob respectively. The tables show by CV assignment: the percentage of points that belong to that value, the measured size of the box these points define, and the root mean square deviation as a measure of error fitting between the valuated blob points with its corresponding actual-size

box model; this error in fitting is achieved by the iterative closest point (ICP) algorithm [7]. Visually these fittings can be observed in the pictures of Fig. 5.9.

Table 5.2: Confidence-Value evaluation for 3D object completion of a cereal-box blob.

Cereal Box, actual size= 13.6 x 5.3 x 21.2 cm				
CV	Points [%]	Measured Size [cm]	RMS Error [cm]	Figure
7	0.52	13.67x4.83x20.23	0.000081	5.9a
6	2.17	15.36x8.21x22.21	0.000098	5.9b
5	12.7	16.01x12.13x22.43	0.000092	5.9c
4	56.53	18.36x11.38x23.16	0.004374	5.9d
3	12.92	18.98x13.14x23.12	0.005539	5.9e
2	6.41	18.37x10.06x24.29	0.005946	5.9f
1	4.29	18.26x10.38x24.38	0.006908	5.9g
0	4.46	19.1x9.19x24.45	0.005781	5.9h

Table 5.3: Confidence-Value evaluation for 3D object completion of a pop-corn box blob.

Pop Corn Box, actual size= 16 x 5.75 x 11.7 cm			
CV	Points [%]	Measured Size [cm]	RMS Error [cm]
7	1.19	15.23x7.697x10.75	0.002548
6	3.61	17.69x8.91x11.01	0.002792
5	10.09	18.61x11.15x13.45	0.003715
4	54.46	20.09x12.29x15.90	0.004805
3	14.41	21.34x12.58x15.24	0.007764
2	8.9	19.45x11.76x14.60	0.006681
1	4.77	19.28x12.41x12.40	0.003378
0	2.55	18.90x11.26x12.45	0.003580

5.3.2 Blob State Propagation under Partial Visibility

Continuing in the blob update context, in a second test we want to observe how the updated state of a captured blob propagates under partial occlusion, i.e., although only a part of the blob is visible and only this part can be updated with current information, the updated state of the blob is propagated to all its structure. For this we map an object with a sequence of measurements before the occlusion; Fig.5.10a shows the 2D RGB image of this object along with its CV-valuated fused points that are fit to its 3D model observed from the front and above; the object is now clockwise rotated and partially occluded, and a sequence of scene observations during this condition was also taken as shown in Fig.5.10b; as we can observe the state of the object's 3D model fit to the partially occluded-blob points corresponds to this new state of the blob structure.

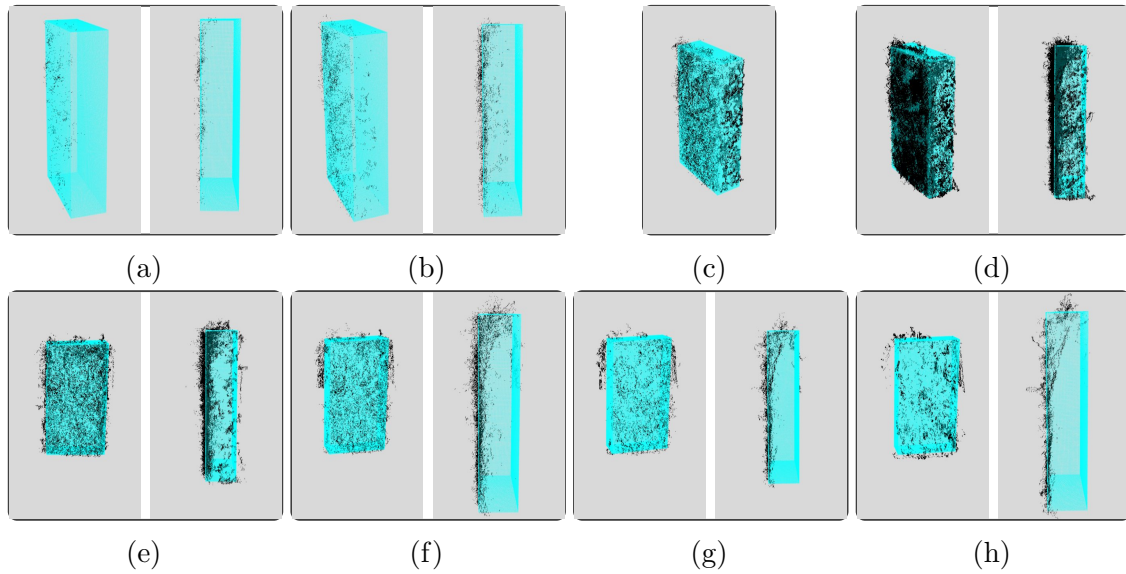


Figure 5.9: ICP fitting of fused points (black points) with different CV to a 3D model (cyan points). a-h) The blob points correspond to the fused cereal box of the scene in Fig.5.8, with different CV assignments of 7 to 0 resp; (from Table 5.2)

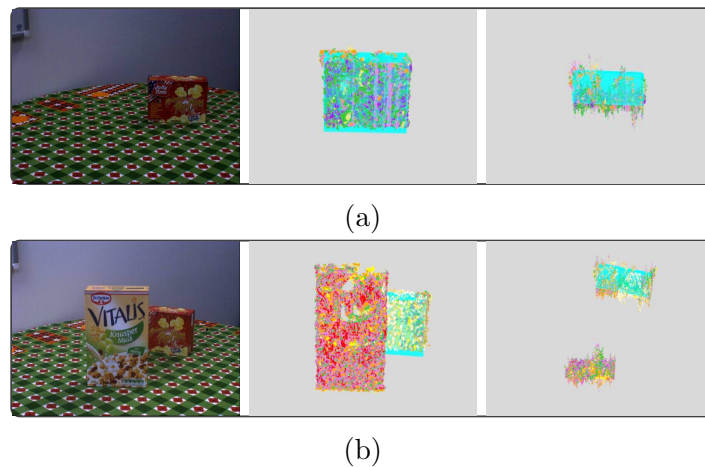


Figure 5.10: Blob state propagation under partial visibility: the pop-corn box is mapped before and during the partial occlusion; a) shows the scene before the occlusion (left), along with its CV-valuated fused points fit to the object's 3D model (cyan) from two different views: front (middle) and above (right); b) although the object is slightly clockwise rotated and now partial visible, after a sequence of mapped measurements we can observe how the object's 3D model fit to the updated blob points corresponds to this new blob state.

5.4 Visual Estimation of Independent Foreground Motions

5.4.1 Ego- & Independent-Motion in Non-Static Scenes

In order to test our approach for map update in non-static environments (Sec.3.5) based on independent ego- and object-motion estimations of 3D segmented foreground structures we set the following scenario up. Our vision system is now mounted in a wheeled robot Pioneer³ that moves through a series of fixed poses to captures and update its map at each spot. As in a non-static environment, the scene is constituted by some movable, graspable objects (see Fig.5.12a), that was modified as the robot travels from one spot to the next.



Figure 5.11: Pioneer 3-DX

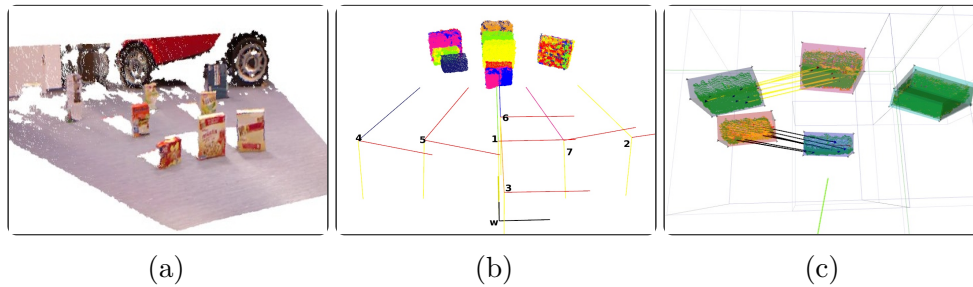


Figure 5.12: Test scene for visual estimation of independent foreground motions. a) 3D RGB image of the scene at its initial state, b) robot ego-motion frames obtained at each setpoint (see Table 5.4) and 3D overlapped blob registrations, c) example of feature-matching flows of two detected object motions.

In Table 5.4 we enumerate the list of the seven robot's pre-programmed poses (set-points) and the measured pose estimates we obtained corresponding to a sample trial. Considering that in each trial the ego-poses reached by the robot are, or might be, biased by the built-in odometry system of the Pioneer, we also report in Table 5.4, as reference, the estimated ego-poses that were obtained by running the test but keeping the scene static. As described in Sec.3.5, since the visual estimation of a pose transformation depends on the quantity as well as the quality of the matching points, we also include for the dynamic test the Mean Squared Error (MSE) of each transformation (MSE Tr) as a reference of the reliability or precision of the estimation, and in order to have a statistics of the accuracy of the process we show the MSE of the Euclidean distance (MSE Eu) between the estimated poses and the reference positions corresponding to 40 measurement for each pose.

Fig.5.12b shows the enumerated pose frames obtained during the trial, we can also observe the overlapped registrations of the moved blobs corresponding to the first three boxes in Fig.5.12a. Fig.5.12c shows an example of the motion detection in two mapped blobs moved during the same sequence interval.

³<https://robots.ieee.org/robots/pioneer/>

Table 5.4: Test results of Pioneer ego-motion estimation in a non-static scene

		Pose (X [cm], Y [cm], Angle [°])			
Set Point	Static Scene	Dynamic Scene	MSE $\text{Tr}(\mathbf{x}^{-3})$	MSE Euclidean	
1	(0,0,0)	(0,0,0)	(0,0,0)	—	—
2	(40,0,10)	(41.21,-0,9.7)	(42.25,0,10.46)	1.019	6.9104
3	(0,-20,0)	(-0,-19.54,0.88)	(0,-20.12,0.15)	1.029	1.5678
4	(-45,0,15)	(-46.19,-0,14.71)	(-45.16,-0,14.0)	0.119	3.3030
5	(-24,0,14)	(-23.62,-0,14,32)	(-24.72,-2,13.78)	0.282	5.9464
6	(0,10,0)	(-0,9.65,1.1)	(1.1,8.32,0.44)	0.688	3.6485
7	(20,0,10)	(20,0,9.7)	(22.22,-0,11.35)	0.316	5.8045

The last two columns of Table 5.4 indicate that our visual motion estimation system is more precise than accurate, i.e., we can not certainly determine the absolute pose of each mapped object in the world but rather determine that the measured geometric relations inside the map are the closest values to the actual ones. In Table 5.5 we report the estimated pose values that were obtained with the moved objects.

Table 5.5: Test results of object-motion estimation in a non-static scene

		Pose (X [cm], Y [cm], Angle [°])		
Set Point		Cereal Box	Pop-Corn Box	
		Meas. Pose	Set Point	Meas. Pose
1	(0,120,0)	(0,117,1.52)	(-33,115,10)	(-33.67,114,8.1)
2	(0,110,0)	(0,106.3,0.28)	(-33,115,10)	(33.84,113.42,9.52)
3	(0,130,0)	(0,126.3,1.92)	(-33,115,10)	(33.78,113.01,8.78)
4	(0,130,0)	(0,126.52,2.39)	(-27,107,10)	(-27.42,104.95,7.40)
5	(-33,120,10)	(-32.77,118.28,4.81)	(0,100,0)	(0,97.15,1.16)
6	(-33,120,10)	(-32.74,118.27,5.46)	(0,94,0)	(1.38,92.84,1.24)
7	(-33,120,10)	(-32.83,118.22,6.04)	(0,94,20)	(0,92.42,22.47)

5.4.2 3D Object Completion in Non-Static Scenes

We now present the results of collating a sequence of range data of the front figurine shown in Fig. 5.13a. As before, the cameras and some objects were moved to different spots during the test (Fig. 5.13b). In order to analyze how precise a given set of CV points of a fused 3D blob images the actual object in this kind of environments, we present the results of the ICP fittings of each obtained CV point set to the figurine’s 3D model; the CV assignments ranges from 0 to 7 and some of the fittings can be observed in Fig. 5.13. We also present the magnitude of the matrix rotation (Eq. 5.1) that was needed in this trial for each fitting: $\{\text{CV_pts}\} \rightarrow \{\text{model_pts}\}$;

$$\begin{aligned} \|\mathbf{R}_F\| &\equiv \|\{\text{valuated_pts}\} - \{\text{model_pts}\}\|_F \\ &= \sqrt{\text{trace}(\mathbf{R}^T \cdot \mathbf{R})} \end{aligned} \quad (5.1)$$

Since the 3D object-model frame and each CV-point frame were set aligned before running ICP, this value will give us a measure of the amount of correction that was needed to obtain the corresponding RMS error value for such fitting. The results are shown in Table 5.6; Although the amount of correction might be similar for the points with extreme CV, we can observe that the points with larger CV present smaller RMS errors; this means that these fused points were better spatially corrected/fused in their local frames before the ICP fitting and therefore describe better the actual size of the object. We can also remark at this point that in spite of the independent sensor and object motions, plus the non-simple geometric shape of the figurine, the percentage distribution of the resulting CV points in this trial resembles to those obtained for the object boxes in static scenes (Sec.5.3.1) as shown in Fig.5.14; this can be an indication of the consistency not only of the fusion approach against these factors (relative motion, shape) but also of the coherence with the independent motion estimation process.

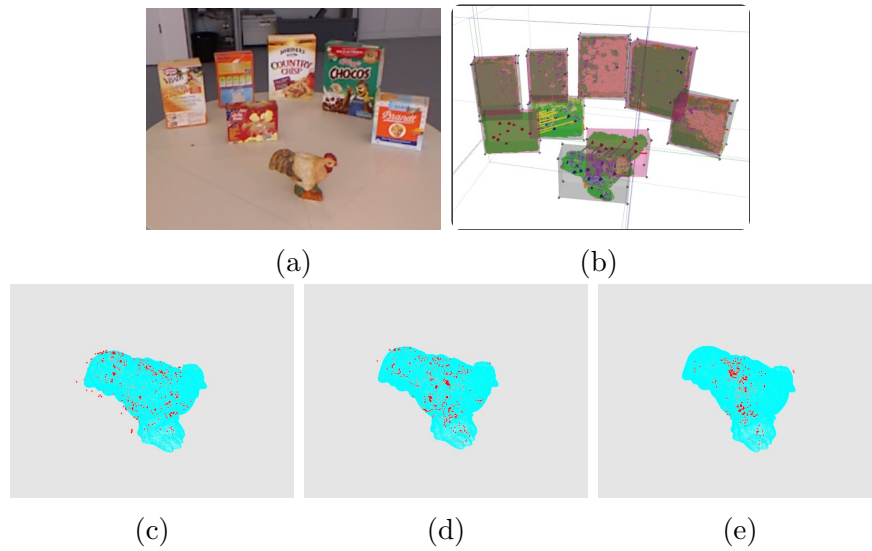


Figure 5.13: Confidence Value and 3D object completion in non-static scene. a) Initial 2D RGB image of the scene, b) example of 3D feature-flows detection corresponding to motion of two mapped blobs, c-e) sets of fused-blob points with highest CV (5, 6, 7, resp.), see Table 5.6.

Table 5.6: Confidence-Value (γ) Evaluation for 3D Object Completion of a figurine in a non-static scene

Chicken Blob				
γ	Points [%]	Rotation Norm	RMS Error [cm]	Fig.
7	1.97	0.257931	0.001407	5.13e
6	2.85	0.279540	0.001439	5.13d
5	3.24	0.334356	0.002410	5.13c
4	74.66	0.411679	0.004266	—
3	4.92	0.339462	0.004003	—
2	4.14	0.255960	0.002779	—
1	3.01	0.260456	0.002608	—
0	5.22	0.251197	0.002689	—

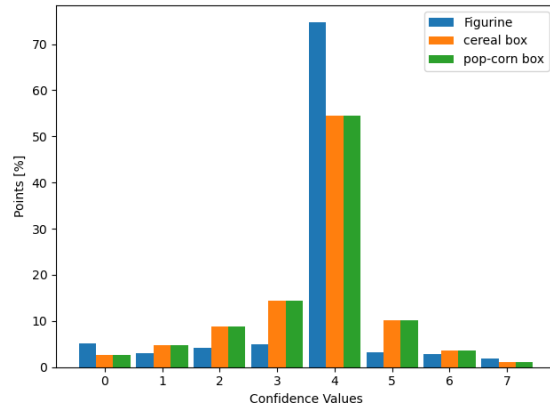


Figure 5.14: Percentage distribution of CV fused points obtained during 3D object completion in non- and static scenes

Summary

In this chapter we presented experiments and test results to validate and determine the performance of the geometric approaches (blob-fusion based on z-buffered re-projection, CV assignments, independent foreground motion estimation) that were proposed and described in Chap.3; we also presented different tests to observe the parameter behaviour and performance of our implemented octree-DS tool. We also described some practical aspects of our stereo-camera setup and showed the corrected, warped RGB-D images.

Chapter 6

Results - Abstraction Layer

Sections

Overview	97
6.1 Identification and Analysis of Object Manipulation Ac- tions	99
6.1.1 Identification of Object from Data in the Semantic Structure	99
6.1.2 Identification of Motion from Data in the Dynamic Model	104
Summary	114

Overview

In Chap. 4 we demonstrated how the geometric information captured and organized at the geometric layer of the framework can be further processed and analyzed at a higher level of perception. We introduced the framework components that interface both layers, *knowledge containers*, and the concept of *object container* as an abstraction of their corresponding geometrical tokens in the scene foreground: the object candidates. We presented three examples in which these actors can have different attributes and functions: *i*) as a hand: for this highly dynamic actor a plugin was developed for the modeling of physical interactions using objects with different functionalities; *ii*) as vehicles: as static, mapped structures, their dimensions, positions and orientations were relevant to support RoMo's motions during an autonomous parking maneuver; *iii*) as pedestrians: whose motions and path predictions are more context-relevant than obtaining their accurate 3D models.

The proposed extension -plugin- for the *characterization of humans interactions* described in Chap. 4 is tested and the results are presented in this chapter. Interaction scenes with objects can present similar manipulation progressions although the objects in used are different, we model and analyze these kind of scenes in the first part of the experimental settings, Sec. 6.1.1; in the second part, Sec. 6.1.2, we analyze interaction scenes to disambiguate the use of an object: we observe each handling sequence, which should correspond to a particular functionality of the object, to obtain the motion traits or patterns that help to characterize and distinguish each action from the other. Our plugin helps to extend the prediction horizons of geometrically described motions to a level of manipulation gestures, like writing, hammering, hand turns, etc.

6.1 Identification and Analysis of Object Manipulation Actions

The objects we employed in the experiments offer a diverse set of motion properties. In Table 6.1 we present the first set of experimental scenes, in which, as described in Sec.4.1, we identify the object in used based on the analysis of the observed motion although these motion trajectories describe similar patterns; whereas in a second set of scenes, shown in Table 6.2, we identify the motion, i.e., the current functionality, for each object in used, as described in Sec.4.2. The implementation code is written in *c++* programming language supported by external libraries like *OpenCV*, *OpenGL*, *qt*, *vtk*, and *eigen*, running on a linux system. For each given scene we present besides the plots of the obtained model some related images of the running experiments corresponding to some supporting functions of the framework previously described, like hand segmentations by either OpenPose or Kalman-Filter detections, 2- or 3D hand-blob optical flow detection, hand paths or encapsulations of projected patterns.

Table 6.1: Object Identification despite Similar Motions Patterns

Object	Action	Motion Attributes	Projected Pattern	# Frames
knife	chopping	translational, random	period-ish, oscillating	140
pen	writing	predominantly translational w/ curved segments	structured, period-ish: <i>sawtooth-ish</i>	267
fork	eating	translational w/ curved segments	Period-ish: repetitive curved patterns	187
glass	drinking	translational w/ curved segments	Period-ish: repetitive curved patterns	188

6.1.1 Identification of Object from Data in the Semantic Structure

The next two scenarios we analyze, set and convey similar manipulation progressions, probably very distinguishable to humans, but not that obvious to artificial agents. The first scenario '*chopping*' vs '*writing*' over a table are two tasks confined to a fixed area over which the hand moves relatively free but close to the interacting surface; the second presented scenario, '*eating*' vs '*drinking*', offers pretty similar manipulation paths as both motions extend back and forth from the table surface to the person's face; Both scenarios, however, leave distinctive manipulation traits as it is shown below. Additionally, in many cases the grasped objects are not detectable by the sensory systems since they are small, thin or partially covered by the hand, like the knife, pen or fork in these particular cases, leaving the hand motions as the

Table 6.2: Motion (funcionality) Identification for each Object in Used

Object	Action	Motion Attributes	Projected Pattern	# Frames
Whiteboard	writing	predominantly translational w/ curved segments	structured, period-ish: <i>sawtooth-ish</i>	482
	erasing	translational w/ curved segments	random, chaotic, unstructured	187
Screwdriver	tightening	predominantly rotational w/ short translations	period-ish, oscillating	250
	poking	translational	spiky, jerky, random	240
Hammer	hitting	translational w/ curved segments	period-ish: repetitive spiky patterns	317
	pulling	translational w/ curved segments	Period-ish, oscillating	269

only source of information to take apart these actions; we focus here again on the hand, the doer of the action motions.

Knife and Pen: Chopping *vs* Writing

At first sight we can observe on the 3D path in Fig.6.2a the '*chopping*' manipulation presents a *quasi*-periodic spiral-*ish* motion that can be also perceived on the oscillatory patterns of the y-axis and distance-to-plane motion profiles in Fig.6.2b, where the x-axis indicates the progression of the chopping spots, which seem to be arbitrary across the chopping area. Regarding the '*writing*' 3D trace in Fig.6.2c we can observe how it differs from the previous writing 3D traces in Fig.6.6a, for this reason in this approach a 3D path is not considered as a direct indication of what action is performed but it could be analyzed to rather infer how such action was performed; The obtained motion profiles in Fig.6.2d, however, are quite similar to the ones obtained for '*writing on the WB*' in Fig.6.6b. The sawtooth-*ish* wave also indicates with each tooth slope the progression and speed of the writing at each row i.e., the steeper the slope, the faster the writing, the length of each slope corresponds to the length of each row and the sharp drop at the end of the slope indicates the change of row. In the same way, the implicit tendency in the y-axis profile gives the vertical advance of the writing from up to down with the size of spikes corresponding in proportion to the size of the actual written letters; finally the big spikes on the distance-to-plane plot can be associated to the gaps between written words or changes of row when the hand separates a bit away from the writing plane. In the framework images of Fig.6.1 we present some views of these scenes with the supporting functions like skeleton-hand detection, hand blob segmentation, 2/3D OF detection, etc.

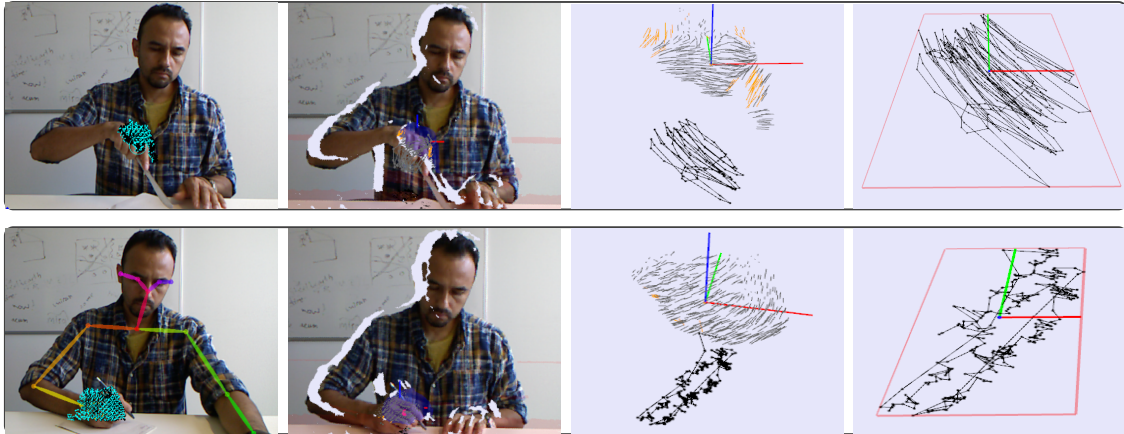


Figure 6.1: Framework-augmented images of *'chopping with a knife'* (up) and *'writing with a pen'* (down). Each row shows the detected 2D OF over the hand blob for the chopping action (up) and also the skeleton-hand detection for the writing action (down), the 3D-RGB image with the hand blob segmentation (purple), a close-up image of the hand OF inliers (black) and outliers (orange) vectors and some tracks of the projected pattern, and the last image shows the final 3D encapsulated pattern of the projected hand motions.

Fork and Glass: Eating *vs* Drinking

We can observe in Fig.6.3a and Fig.6.3c the motion paths of both actions are very much alike, both outline similar curved motion segments of the hand, spanning from the table to the person's face; we can bring, however, some key differences that help us to take them apart. Comparing the motion profiles of Fig. 6.3b and Fig.6.3d we can observe that in both actions the captured motion component on the x-axis corresponds to minor movements of arbitrary or random hand positions along its table-face path which is better described, in these trials, by the y-axis profiles; the length of the flat segments on the top of these latter plots indicate, in turn, the duration of the hand remaining closer to the table surface (whose location is assumed to be on the upper part of each projected pattern). Independently of the number of half cycles or *positive wave crests* on the distance-to-plane plots we can observe that the heights and durations of the peaks during the eating with a fork are smaller and shorter than the peaks during the drinking action. According to the plots, during the eating the heights are around 250 mm, which would correspond to the distance between the table and the person's face, or they could be even smaller (like the 4th peak in Fig.6.3b), which would mean that the person's face also moves closer to reach sooner the fork in a typical eating movement. On the other side, we can also notice the wave peaks drawn during *'drinking'* are in general higher ($> 300mm$) and present a kind of inflection point approximately $\sim 250mm$; This point indicates the first contact of the glass with the mouth and the continuation of the hand motion upwards in order to place the glass a bit higher and tilt it toward the person (while the person leans the head slightly backwards). On the slope down of the peaks we can also notice another subtle inflection point indicating the end of the mouth-glass contact and returning the glass to the table. These points can be observed in Fig.6.3d and Fig.6.4b. The differences in head and hand postures between these actions can be also observed in the 2/3D images presented in Fig.6.4.

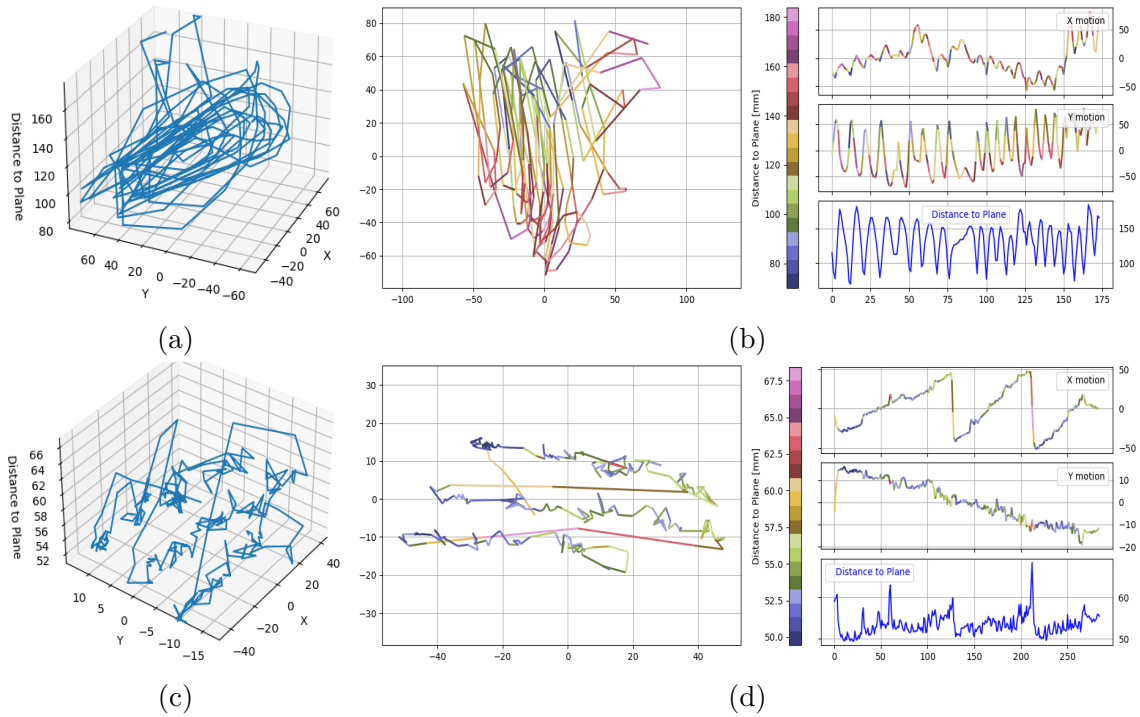


Figure 6.2: Manipulation pattern and motion profiles of *'chopping with a knife'* (up) vs *'writing with a pen'* (down). (a, c) show an exemplary 3D path trace for each action, (b, d) show the obtained projected patterns (left) and motion profiles (right). The 3D curled-loop patterns in (a) is mainly reflected as oscillatory variations on the y-axis and on the distance-to-plane motions and also indicated as cutting line strokes along this axis on the projected pattern in (b), while the x-axis motion component indicates arbitrary cutting spots across the chopping area. Although the 3D trace in (c) might look different to the ones in *'writing on board'* (Fig.6.6a), we can observe that the obtained motion profiles in (d) are consistent with the patterns previously obtained in (6.6b) independently of the scale or context where the action takes place; a sawtooth-ish-wave component is also identified as a major distinctive feature in this manipulation, corresponding the y-axis motion to the vertical progression of the writing and the amplitude of its spikes to the proportional size of the written signs; larger peaks in the distance-to-plane plot indicate bigger separations of the hand to the writing plane, which occur between words or changes of writing rows.

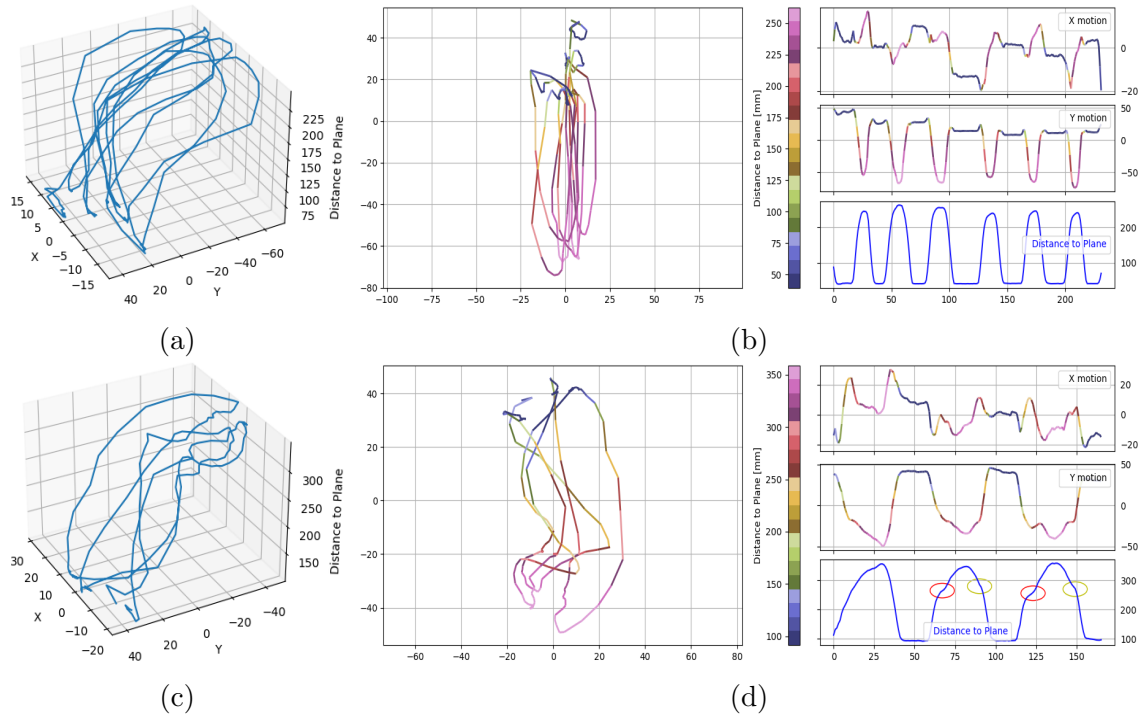


Figure 6.3: Manipulation pattern and motion profiles of 'eating with a fork' (up) vs 'drinking with a glass' (down). (a, c) show the 3D path traces of each manipulation, (b, d) show the projected patterns or fingerprints (left) and the motion profiles (right) for each action. We can notice in both actions repetitive movements of the hand going back and forth from the table to the person's face, which are better described in our trials on the y-axis profile, and where also arbitrary repetitions and durations of this motion can be seen, while the x-component can be considered as a minor motion produced by random or arbitrary hand positions along the major path. Although both actions outline these similar manipulation progressions the most distinctive feature can be observe on the additional peak in each crest of the distance-to-plane motion during drinking; this additional segments are produced by the further motion of the hand to get a bit higher and tilt the glass toward the person' mouth and whose head is leaned backwards as shown in Fig.6.4a. The inflection points (red, yellow) mark the start and end of these peak segments in the distance-to-plane in Fig.6.3d and Fig.6.4b.

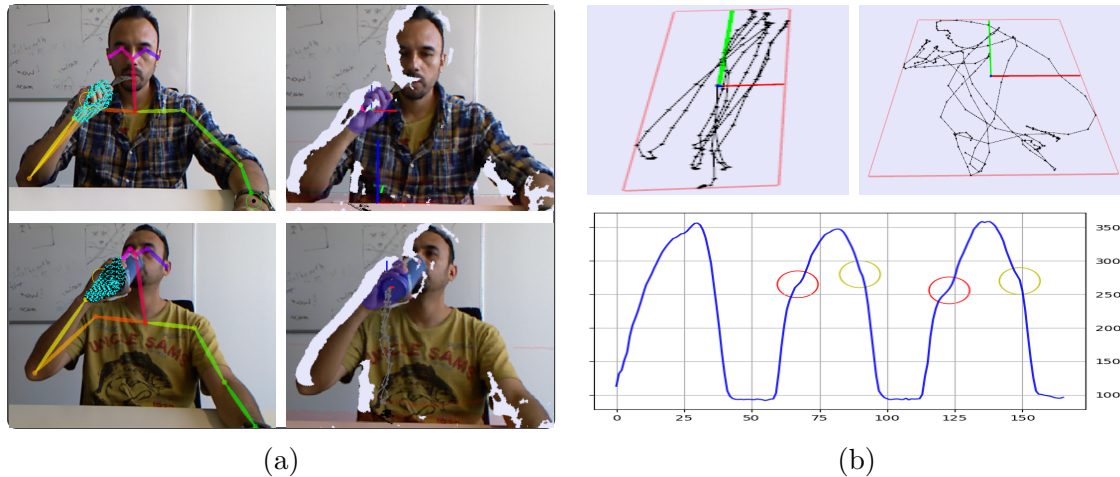


Figure 6.4: Framework-augmented images of 'eating with a fork' and 'drinking with a glass'. a) shows for the 'eating' (up) and 'drinking' (down) trials 2D OpenPose skeleton-hand images with OF detection and 3D-RGB images with hand-blob segmentations (purple). We can observe in these snapshots the difference in the hand-head postures; the hand approaches the person's head during eating and the head remains still or moves a bit forward to also approach the hand, this highest point of the hand at each loop is $\sim 250mm$ as can be seen in 6.3b, while during drinking around this height the first glass-mouth contact is made (red circles in b), the hand then moves on in order to put the glass higher and tilt it as the person leans back the head a bit, as shown in (a); this state continues until the glass is separated (yellow circles) to be returned to the table. (b) also shows the 3D encapsulated projections of each action motions (up).

6.1.2 Identification of Motion from Data in the Dynamic Model

Whiteboard (WB): writing *vs* erasing

We start with a particular case since instead of considering two different objects like a WB-marker or a WB-eraser for writing or erasing we select as primary object the item that also acts as the interaction background, namely the whiteboard; this trial clearly illustrates how the motions projected onto the background can help to distinguish the manipulation. Both of these actions are examples of translational rather than rotational motions since the positions of the hand are more perceptible and descriptive of the actions than the small turns or rotations the hand wrist can exert during the manipulations.

In Fig.6.5 we show some snapshots of the executed scenes corresponding to the occidental (up) and oriental (down) writing styles; for both cases we present the 2D images of the OpenPose skeleton with hand-blob and OF detection (left), the 3D-RGB images of hand-blob segmentations and projected paths (middle) and the 3D encapsulations of projected manipulation patterns onto the WB (right). In order to have a first impression on how the 3D motion paths of these actions look like, we show in each of Fig.6.6a and Fig.6.6c two examples of the obtained 3D traces corresponding to the writing and erasing on WB respectively; the upper image in Fig.6.6a shows the obtained 3D hand path of the occidental writing style i.e., horizontal

rows with left-to-right progressions and adding new rows down, it also shows in the lower image the obtained 3D path of the oriental writing style i.e., vertical rows with up-to-down progressions and new rows to the left; In Fig.6.6b we can see the corresponding manipulation fingerprints on the left and the decomposed, projected motions (right) of these 3D traces. Fig.6.6c shows two different erasing sequences, one with mostly short, sharp strokes (up), the other with rather long and rounder moves, yet both sequences keep in general the hand closer to the WB than during writing; their projected fingerprints and decomposed motion patterns are shown on the left and right of Fig.6.6d respectively. On the obtained motion patterns of the manipulation modeling (right of Fig.6.6b) we can observe that both writing actions present a rather structured, periodic motion sequence, whereas the motion modeling of erasing shows arbitrary, chaotic motion strokes (right of Fig.6.6d). Observing the *sawtooth-ish* wave outlined on the x-motion for the occidental writing style in Fig.6.6b we understand each slope of the plot as the left-to-right advance of the writing in each of the three rows, where the steep of the slope indicates the speed of the writing i.e., the steeper is the slope the faster is the writing, and the length of the slope corresponds to the length of the written row; we can attribute the sharp drop at each end of the slopes to the change of writing line i.e., the returning of the hand to start a new row. On the y-motion we can first notice the vertical progress of the action going downward to the bottom of the board, we can also associate the variations in amplitude of the y-spikes to the size of the written signs. In the distance-to-plane plot (blue) we can observe the big spikes correspond to the separation of the hand from the board either due to the changes of rows or the gaps between written words; the steady increase presented in the hand-board separation can be taken as an indication that the hand centroid gets closer to the board on the upper areas i.e., the forearm is rather parallel to the board with the elbow also close to the board, and gets away on the lower areas during writing, with the forearm somewhat perpendicular to the board. Analogous observations can be deduced about the obtained patterns of the oriental writing style (down in Fig.6.6b).

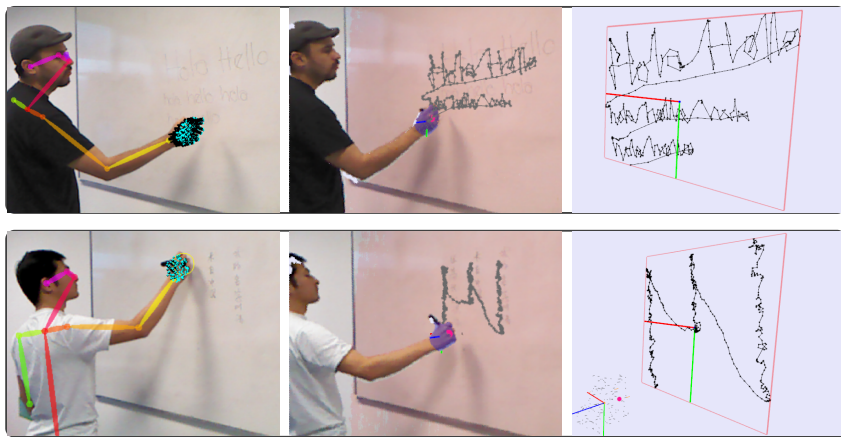


Figure 6.5: Framework supporting functions for the WB-writing scenes. For occidental (up) and oriental (down) writing styles: (right) Openpose skeletons with hand-blob and OF (cyan) detections, (middle) 3D-RGB images of hand-blob segmentations (purple) and projected paths (black) during writing, (right) 3D encapsulations of projected manipulation patterns.

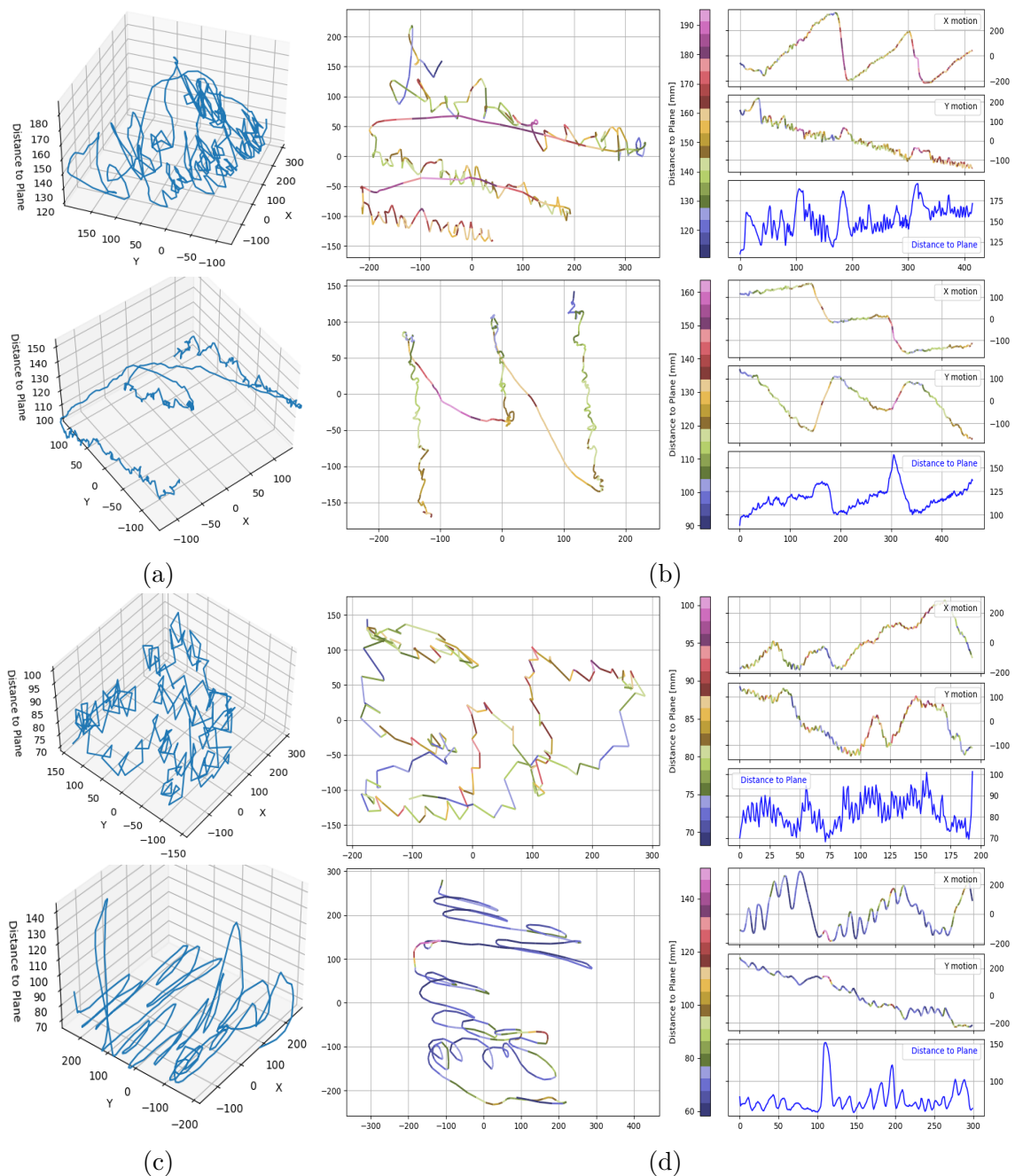


Figure 6.6: Manipulation modeling of actions on WB, 'writing' vs 'erasing': a) although the obtained 3D hand traces of the performed actions for the occidental (up) and oriental (down) writing styles look very different we can observe on the right of b) a *sawtooth-ish* motion modeling in both sequences, where each outlined tooth fundamentally corresponds to the writing progress along a row with the change to a new row, (left) shows the corresponding fingerprints or projected patterns of both writing styles. c) Shows two different erasing sequences, one with short, sharp strokes (up), the other with round, longer motions; we can observe in the obtained manipulation modeling in (d) that these actions do not reveal any pattern or structure; erasing a WB consists of mostly arbitrary motion strokes.

Screwdriver: *tightening vs poking*

Although both actions are executed predominantly around one spot, this is, the screw’s head and the poked point, the x- and y-projected motions they describe are very distinctive from each other; Fig.6.7 shows some images of the experiments carried out augmented with some framework functions. Fig. 6.8a and Fig.6.8c show two corresponding 3D traces of the executed experiments. As can be observed on the projected pattern and motions in Fig.6.8b the roughly oscillating wave on the x-axis during *tightening* describes the change of position of the hand centroid during the back-and-forth twists of the wrist, the obtained span of these x-displacements are a bit larger than the ones presented on the y-axis, which indicates a larger motion impulse on the x component; we can also observe that the hand distance to the plane presents small up-and-down variations corresponding to the vertical components of the twists and with a constant trend downwards as the screw is getting tight. Opposite to the tightening we can observe in Fig.6.8d the most distinctive motion of the poking action is a vertical movement corresponding to the sudden up-and-down poking strokes which are mainly depicted with the abruptly spiky lines in the distance-to-plane plot and also reflected on the x- and y-axis with the zigzagged motions around the poked point.

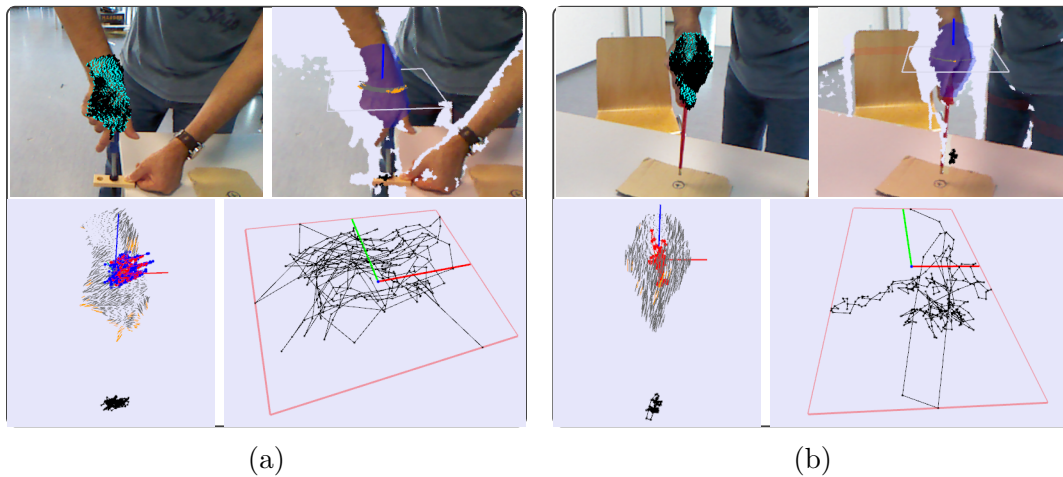


Figure 6.7: Augmented framework images of a) *'tightening'* and b) *'poking with a screwdriver'*. For each action the upper row shows the 2D image with OF detection over the hand and the 3D-RGB image with hand blob segmentation (purple) with an additional Normal-plane centered at hand centroid for hand-twists calculations; each left image in the lower row shows the corresponding 3D hand OF inliers (black)/outliers (orange) vectors with the KF predicted (blue)/corrected (red) path points and their projected point path below (black); the last image shows this encapsulated projection for each action.

As mentioned *'tightening'* also contains a characteristic rotational motion component due to the hand’s wrist turnings; in order to represent this rotational motion we estimate the rotation angles about the hand’s z-axis, which is parallel to the background’s normal plane and also aligned-*ish* to the screw’s long axis; for this the obtained 3D OF inliers of the segmented hand blob are projected onto the plane defined by the hand’s z-axis centered at blob centroid (see Fig.6.7) and the rotation angle at each frame can be estimated as described in Sec.4.2.4. Fig.6.9a and

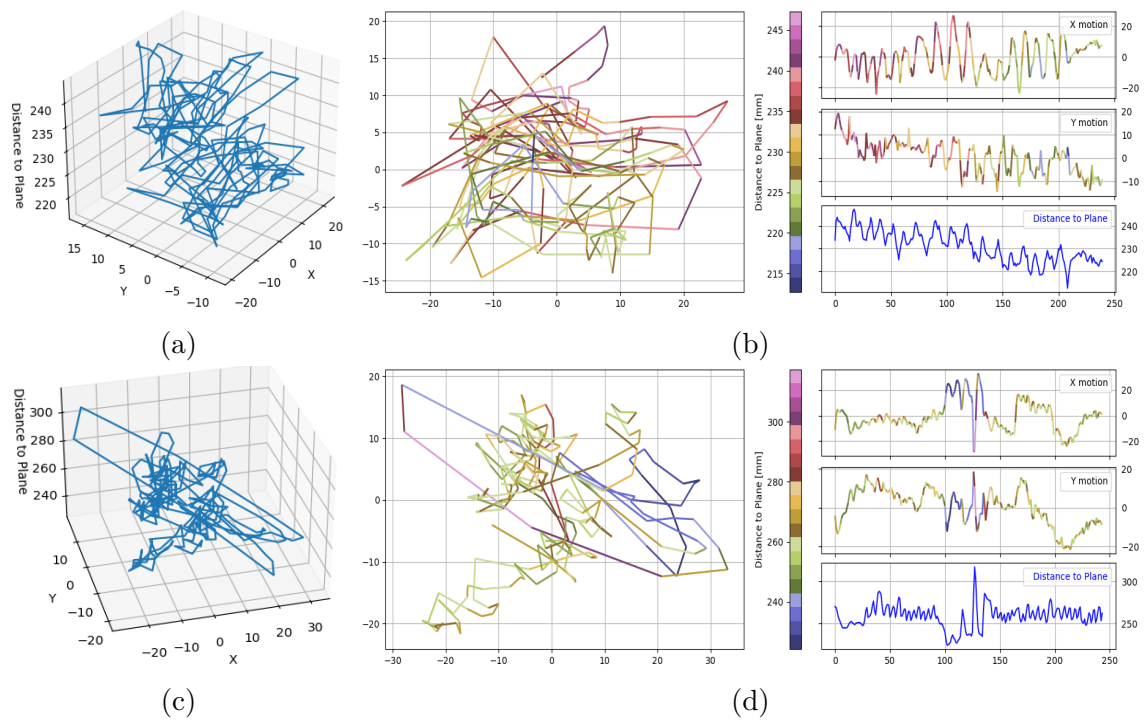


Figure 6.8: Manipulation modeling of actions with a screwdriver: *'tightening'* (up) vs *'poking'* (down). (a, c) show the 3D path traces of the manipulation motions and (b, d) their corresponding obtained patterns (left) and decomposed motion modelling (right). The irregular oscillating wave on the x-axis in (b) corresponding to the hand's tightening twists contrasts the spiky motion components in (d) corresponding to the abrupt poking strokes. The hand distance-to-plane in (b) tends downward as the screw gets tight, while in (d) flutter around a distance.

Fig.6.10a present two exemplary sequences of each action that show the projection of detected 3D OF vectors for the estimation of rotation angles at different frames and in Fig.6.9b and 6.10b the evolution of estimated rotation angles at each frame along with their standard (middle) and cumulated (right) distributions are shown; we can observe in the latter figures that although the standard distributions of both actions produce similar angle ranges the *tightening* is better distributed, less centralized as the *poking* one; additionally we can also notice the cumulated distribution of *tightening* spans a wider range of angles, which means that more consecutive angles support or reinforce the rotational motion of the hand and do not counteract as in case of *poking*; the shift to the right in the cumulated histogram in Fig.6.9b could indicate a *bias* in our measurements, which can be produced by several factors i.e., the irregular, not symmetric surface of the hand, or that the angles are estimated about the hand centroid and not about the actual center of rotation, etc.

Hammer: hitting *vs* pulling

Snapshots augmented with some framework supporting functions for the analysis and modeling of these two actions are shown in Fig.6.11 and the obtained 3D traces can be seen in Fig.6.12a and Fig.6.12c, respectively. The three spiky plots in the motion modeling in Fig.6.12b give immediately the idea of an action with sudden and sharp motion strokes that correspond to the impulse movements of the hand during '*hitting*'; as shown in the projected pattern these motions are quite centralized and slightly symmetric around the point being hit, the nail head. In contrast, pulling with a hammer's claw delineates a rather oscillating wave that is mainly reflected in this trial on the y-axis and distance-to-plane plots in Fig.6.12d, which correspond to the back and forth swinging '*pulling*' motions, whereas the x-axis represents a minor motion basically produced by small and random variations of the hand position along the principal motion, as can be observed in projected pattern.

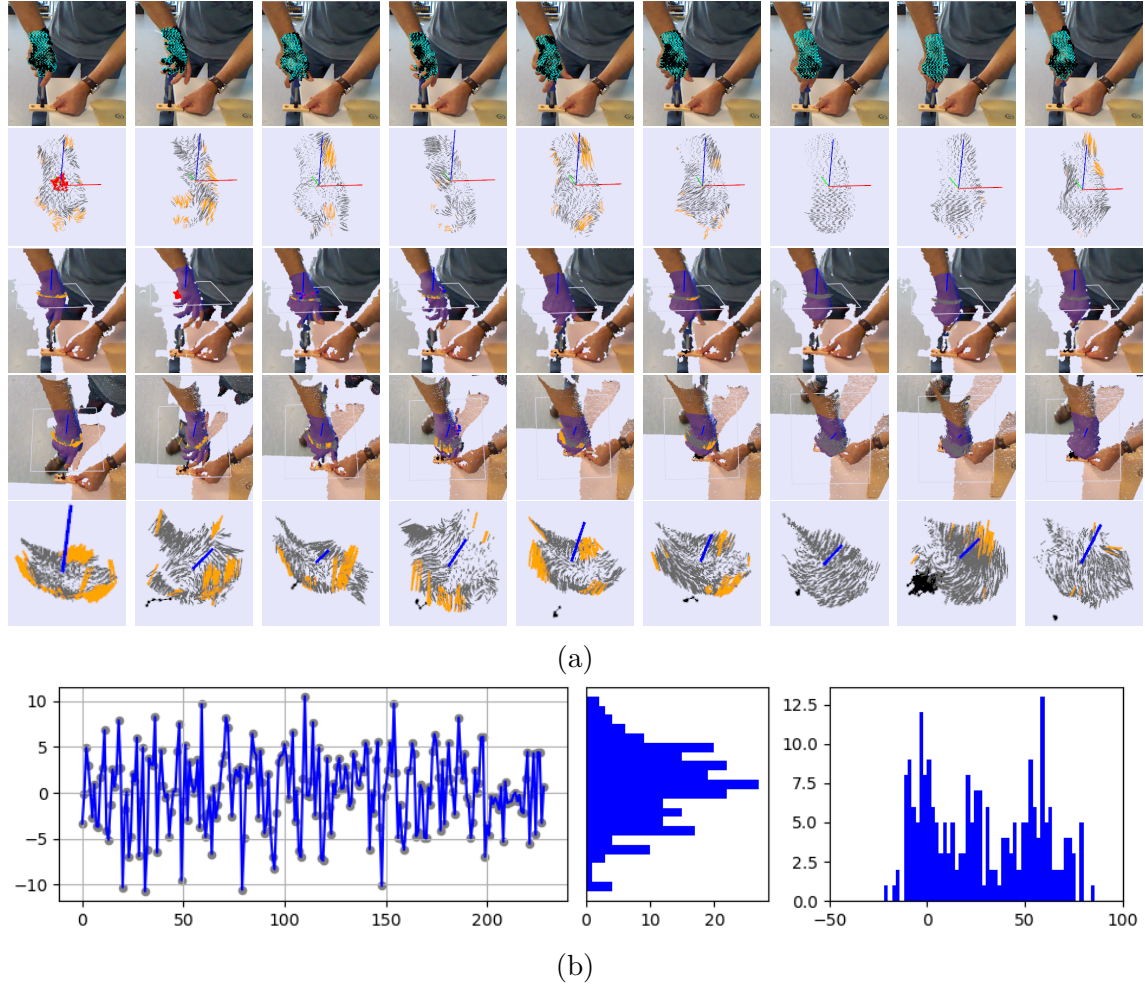


Figure 6.9: Representation of the rotational hand motion component during *'tightening with a screwdriver'*. a) Exemplary sequence of *'tightening'* frames for 3D OF projection and rotation-angles estimation: 1st row shows the 2D OF detected over the hand blob, in the 2nd row close-up views of 3D hand OF inliers (black) and outliers (orange) are shown, 3rd and 4th rows show two different views of the segmented 3D hand and augmented z-axis plane for 3D OF-vectors projection, last row shows the obtained projected sets of 3D OF inliers (black) and outliers (orange) onto the hand's z-axis plane for rotation angle estimates between frames. b) shows the evolution of the estimated rotation angles at each frame (left) and the distribution of these angles (middle); we can observe the cumulated rotations (right) spans a wider range of angles corresponding to full hand turnings.

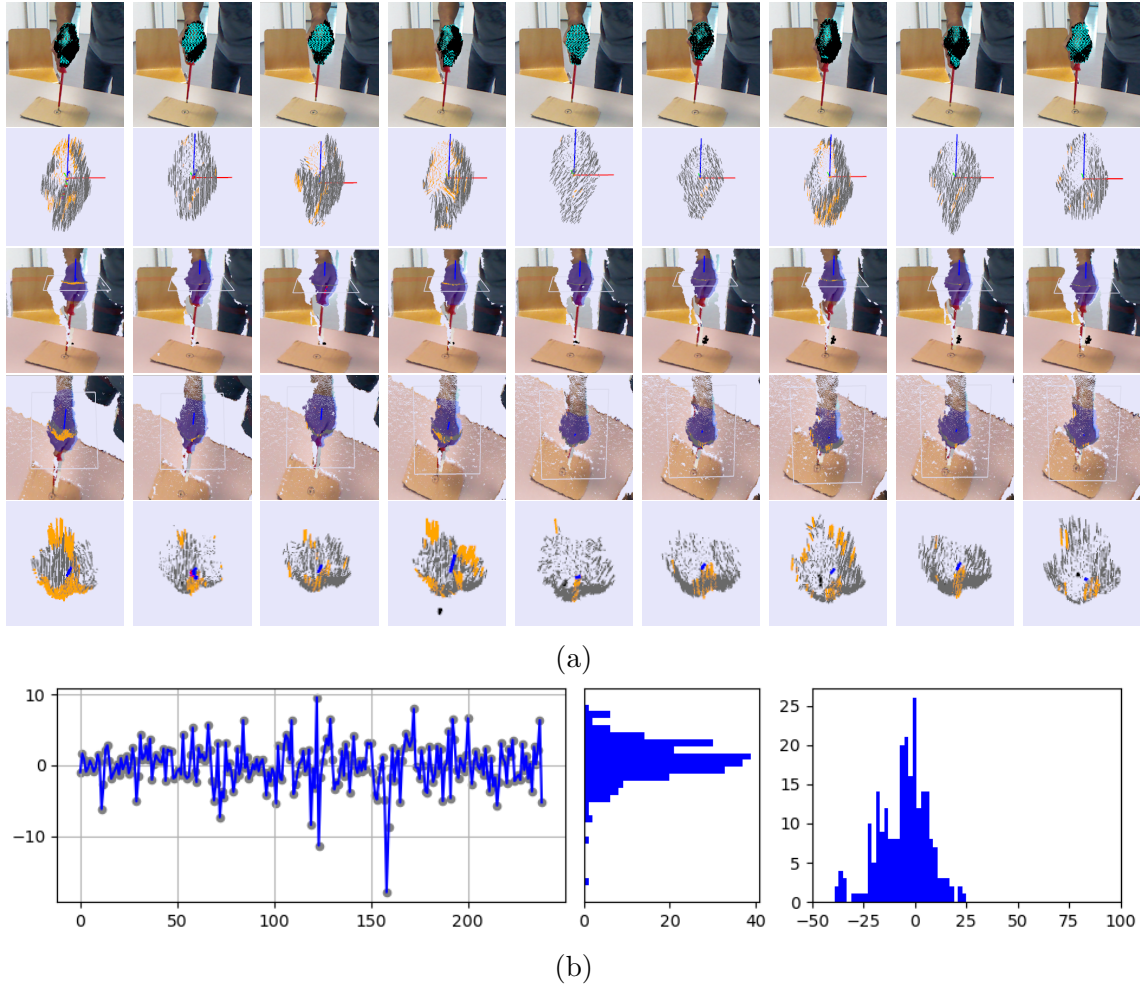


Figure 6.10: Representation of the rotational hand motion component during '*poking with a screwdriver*'. a) Exemplary sequence of '*poking*' frames for 3D OF projection and rotation-angles estimation: 1st row shows the 2D OF detected over the hand blob, in the 2nd row close-up views of 3D hand OF inliers (black) and outliers (orange) are shown, 3rd and 4th rows show two different views of the segmented 3D hand and augmented z-axis plane for 3D OF-vectors projection, last row shows the obtained projected sets of 3D OF inliers (black) and outliers (orange) onto the hand's z-axis plane for rotation angle estimates between frames. b) shows the evolution of the estimated rotation angles at each frame (left) and the standard distribution of these angles (middle); unlike '*tightening*' in this case we can deduce by observing the cumulated distribution (right) that consecutive angles counteract to each other resulting in a narrow distribution, this is, no rotational motion is supported.

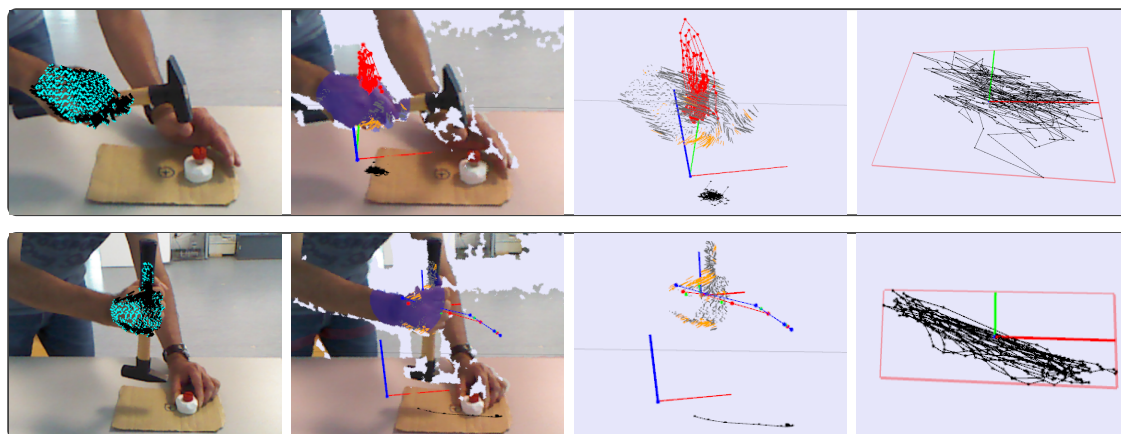


Figure 6.11: Framework augmented images of *'hitting'* (up) and *'pulling'* (down) with a hammer. path of tracked hand centroids; the next image presents a zoomed-in view of the 3D hand OF inliers (black) and outliers (orange) vectors of the hand blob with the KF predicted (blue) and corrected (red) path points, the last image shows a close-up view of the 3D encapsulated pattern of projected motions onto the background plane.

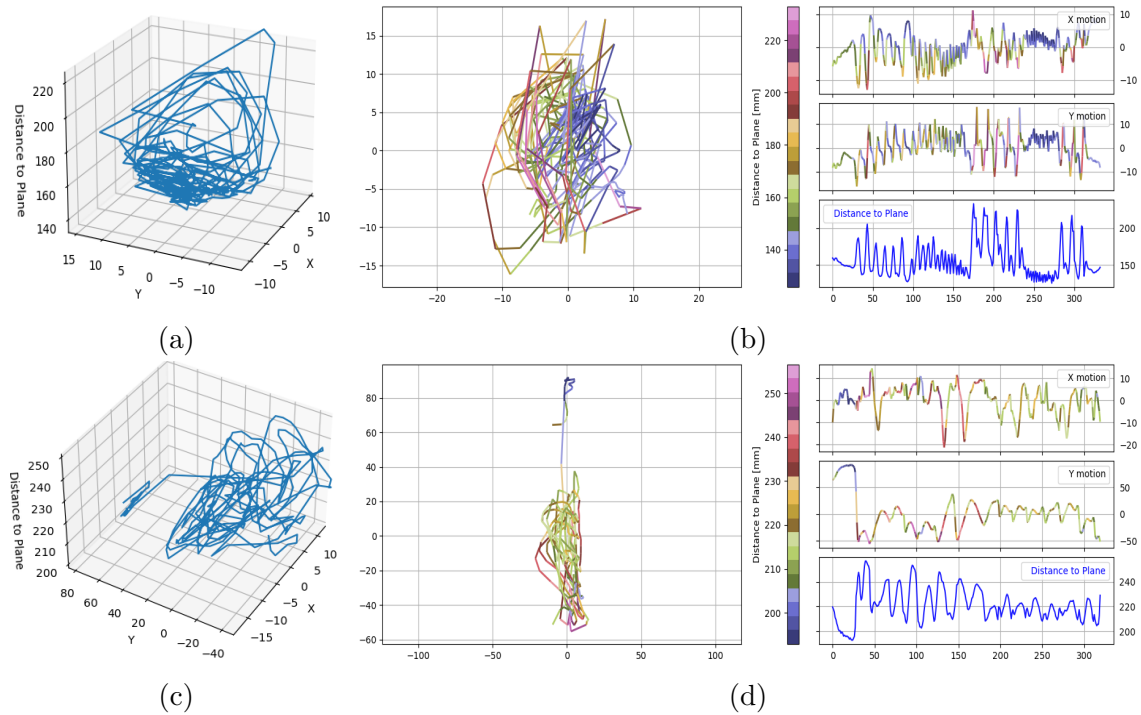


Figure 6.12: Manipulation modelling of actions with a hammer: *'hitting'* (up) vs *'pulling'* (down). (a, c) show the 3D path traces of the presented trials for each action and (b, d) their corresponding obtained patterns (left) and motion modeling (right). Although the hitting motion impulses are better perceptible in the distance-to-plane plot, these sudden and sharp strokes are also reflected by the spikes on the x- and y- components produced around the point being hit as shown in the plots and projected pattern in (b); the pulling action is essentially characterized by the back-and-forth swinging motion of the hand, which is mainly described on the y-axis and the distance-to-plane motions, on the x-axis we can notice a minor motion basically generated as random deviations of the hand position along the pulling trajectories as shown in (d).

Summary

In this chapter we presented a new modeling for physical interactions of manipulation sequences. For this, the 3D tracked, motion trajectory of a person's hand are projected onto the immediate supporting plane and the distances of the hand to this plane are also registered; this results in a characteristic fingerprint or manipulation pattern of the action that is projected upon where the manipulation progression directly takes place. 3D Optical Flow estimation over the hand blob also assists to the modeling of possible hand turns or twists during the interactions. The presented plugin is tested for the modeling and labelling of actions in manipulation scenes using the same object but with different functions, and also to distinguish similar manipulation progressions but that correspond to different actions.

Chapter 7

Conclusions

Sections

Overview	115
7.1 Summary	117
7.2 Future Works	118

Overview

In this final chapter we summarize and make some final comments and observations about our presented work; we also present some suggestions for the extension of the framework.

7.1 Summary

In this work we presented a scene analysis framework based on a bottom-up hierarchical concept of data abstraction that organizes the incoming sensory information (sensor-level) in essentially two coupled layers of abstraction, therefore, the framework is conceptualized as **hybrid**, linking a geometrical and an operative or functional layer of perception; At the lower level of perception, the geometrical layer, the captured structural data obtained from a rigid 3D point cloud is processed and organized firstly in back- and foreground regions, where the former is mainly composed of large structures in the scene, like (part of) tables, walls, windows, furniture, etc; and that are modelled in the framework as supporting surfaces and/or functional areas to indicate those scene zones where an action, task or activity takes place, while the latter (foreground regions) are further processed to obtain a set of individual structures, **3D blobs**, representing tentative object candidates; At this level we also analyze independent sensor and object motions and apply a *z-buffered re-projection* procedure for a consistent 3D object completion and mapping; diverse tracking mechanisms as well as a specialized human-body detector were also implemented, adjusted and integrated at this framework level to observe and captured the geometric motions of dynamic actors, the **doers** of action(s), inside the scene. At the higher-layer of perception functional attributes as well as dynamic properties are observed and recollected, and assigned to objects and actors; In this way, relevant geometric components inside the scene act as *physical tokens*, each of them *indexing* not only to own geometric properties but also to its own *augmented* vocabulary of functional/operational knowledge; Consequently, the object candidates can take particular context-depending meanings or roles different to small, grasping objects as shown in the RoMo implementations, where they can be considered as vehicles or pedestrians depending on the application. The framework provides with the required components and procedures to keep and maintain this connection, e.g., the *knowledge containers*, which store relevant information that is locally recollected in the current scene, and the *Atlas*, which store a-priori or gained-by-experience information about general class-object attributes and features. A higher agent of perception or *plugin* for analysis and modeling of physical interactions that utilizes the recollected information at the geometric level was also implemented and presented in this work; as shown, our plugin is able to model and differentiate manipulation sequences when handling the same object or observing similar actions using different objects; the agent extracts distinctive motion traits of the observed manipulation sequences that expand the **prediction horizons** from a geometric scope to a level of action or task, e.g., from tracked hand motion sequences to a *writing-* or *tightening-*action labelling.

From the proposed framework concept and presented work we draw and summarize the following contributions.

Flexible and structured 3D data. The sensory data, as a captured 3D rigid geometric representation of the world is split and re-organized firstly in two main scene regions: back- and foreground, which are, in turn, also broken down into a set of independent bodies that represent tentative object candidates in foreground embedded into a set of larger structures in background, like tables, walls, etc. This flexibility in the geometry of the world representation allows us to focus, if required, on individual analysis and update of objects or regions of interest.

Knowledge-augmented geometry. The purely geometric information corresponding to tentative objects is augmented with functionalities and attributes at the higher layer of perception in the framework, while background geometries are also identified and labelled in *functional areas* corresponding to areas where objects acquire certain or special states or functions inside the scene; particularly in this work plane surfaces are identified as *supporting or interacting planes* to indicate those specific areas where an action or object manipulation takes place.

Foreground geometries as tokens for data inquiry. Derived from the previous remark we can also point out that operative/functional knowledge is anchored and indexed by the geometry of the segmented bodies in the scene: the *knowledge containers*, recollect locally-gained information through current scene observation, while the *Atlas* is the database that contains general, a-priori or gained-by-experienced information of the world that can be used to complete local data or to predict and hypothesize about states, functions, shape or behavior of objects in the current scene.

Boost of geometric-level motions to task/action-level labelling. A common prediction horizon obtained at a geometric level by a typical tracking mechanism that follows a target ranges in general from fractions to the order of seconds ahead of the current target state. With our plugin we were able to expand these prediction horizons where tracked hand motions at the geometric scope were boosted to a level of action or task representing object-manipulation sequences, e.g., a set of hand translation paths to a *writing-* or *chopping-*action progression, or a series of hand turnings to the distinction of a either *tightening-* or *poking-*action manipulation.

Extensible and adjustable hybrid architecture. The presented framework is conceived and structured on data-processing layers that correspond to increasing levels of data *perception*, i.e., how the information is processed and what meaning is given to it at each level, e.g., at a lowest level, *sensor layer*, we dealt merely with two shifted arrays (for 3D points and colors) that were adjusted (calibrated) for a proper point-color correspondence, while in our main two **hybrid** layers we regroup the data in a set of 3D structures or series of points representing tentative scene objects, regions or motion paths at the *geometric* layer, and that were able to take different meanings and behaviours, like vehicles, pedestrians or actions at the higher *abstraction*-layer. With this concept we did not only break an scene analysis requirement into smaller problems but also organize these smaller issues into abstraction-levels of data processing; this makes the framework extensible and customizable in each layer, e.g., by integrating more tracking modalities, specific object detectors, etc; or by adding higher agents of perception.

7.2 Future Works

In the presented work we proposed a hybrid framework that aimed to focus mainly on the two central levels of perception that we called the geometric and abstraction layers; we presented in Chap.2 an additional stage of data processing at sensor level; although this stage could also be part of the geometry-layer we considered this separation an adequate arrangement to distinguish sensory data/signal conditioning issues from rather the core of higher data analyses, like scene segmentation, object completion, etc.

In the same way, many other additional features, tools and improvements can be proposed and integrated to the framework; below we listed some potential improvements and extensions we consider can be directly applicable and suitable to the proposed concept and work.

Additional geometry-layer tools. In the geometry layer we define **what** and **how** data of interest is extracted and represented; this means that several tools can be implemented and integrated in order to capture and encode this relevant and functional information. In this work we presented some tools that were also integrated as alternative, plausible solutions to some of the encountered issues e.g., the edge-based tracker to detect and track the pose of objects based on 3D models, the Particle Filter (PF) to track objects or hands (wearing flashy gloves) based on color distribution, or the B-Spline to detect and track human contours; in the same way, other tools like the SIFT- and YOLO-based object detectors that were implemented but not yet integrated could facilitate the 3D segmentation, clustering and completion of foreground objects; Additionally, other new tools can be integrated to contribute with the presented work e.g., a semantic mapping could help to an automatic labelling of functional areas in background regions.

Interaction analysis of two-hand motions. This can be an evident extension of the presented plugin in which with the analysis of a single hand's motions some object manipulation sequences are modeled and labelled as actions, like writing, drinking, chopping, etc; the integration of the second hand in the motion analysis would contribute not only to cover more manipulation sequences and reassure the already analyzed actions, but also could help to find manipulation variants inside particular actions e.g., whether an action using the same object, a knife, corresponds to chopping, peeling or slicing manipulation, since in these cases the guide (2nd) hand would give us additional indications how or where the object is held.

Modeling and labelling of activities. Since an activity comprises the execution of a consecutive series of actions, this proposal can be also considered a natural extension of the presented work; This extension would imply the implementation of an external plugin with a supervisory mechanism to register the *non-* and *observed* actions and decide based also on their chronological order the labelling of the executed activity; For example in Fig.7.1 we show a sequence of three actions with their respective projected patterns.

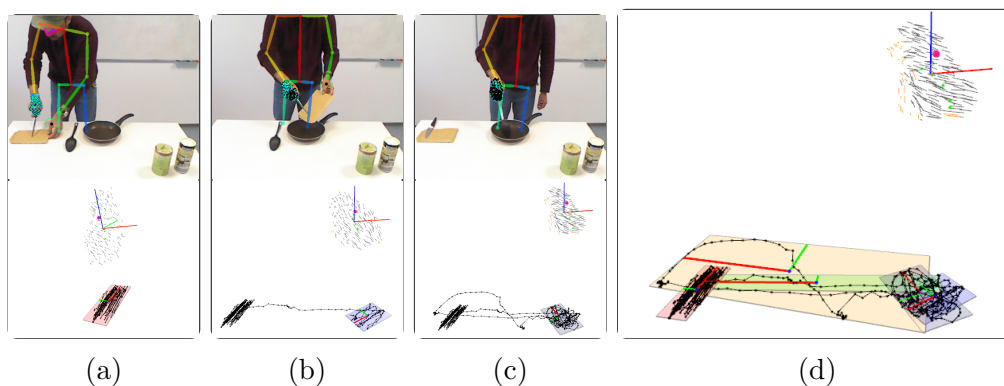


Figure 7.1: Modeling and Labelling of Activities. An exemplary sequence of three actions and their PCA-encapsulated patterns of: a) chopping, b) pushing, c) stirring; d) shows all the PCA-encapsulated pattern of the sequence, including the transportation paths, from chop to push area (green) and from push to stir (orange).

Cognition/Inference layer. The presented framework covers from a lowest -sensor-level, passing through a geometrical data representation up to a higher layer - modeling level- of perception, the next step will be the integration of a **cognition** layer following this bottom-up hierarchical concept of abstraction layers in the scene analysis; In this highest layer a *learning* machinery would be expected to **infer**, predict and/or classify similar action patterns based on the manipulations models obtained in the previous layer.

To conclude this chapter and work we would like to mention two alternative, related projects which can be derived from this work, or the framework can be part of: **Helping recognizing objects by manipulation rather than by shape**, as mentioned in this work, an object is designed to fulfill a specific function based on its shape, hence, shape and manipulation are strongly associated and complemented, knowing one can help to infer the other; **Transfer of functional properties to similar-shaped objects** Going deeper in a human way of thinking, the absent of a required object or tool makes us to search for possible, similar-shaped substitutes in our surroundings based on valid geometrical and functional criteria.

Bibliography

- [1] Martin Armstrong and Andrew Zisserman. “Robust Object Tracking”. In: 2001.
- [2] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-squares fitting of two 3-D point sets”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 9.5 (Sept. 1987), pp. 698–700. ISSN: 0162-8828.
- [3] T. Bailey and H. Durrant-Whyte. “Simultaneous localization and mapping (SLAM): part II”. In: *IEEE Robotics Automation Magazine* 13.3 (2006), pp. 108–117.
- [4] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An Introduction to Splines for Use in Computer Graphics & Geometric Modeling*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. ISBN: 0934613273.
- [5] Herbert Bay et al. “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142.
- [6] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (Sept. 1975), pp. 509–517.
- [7] Paul Besl and H.D. McKay. “A method for registration of 3-D shapes. IEEE Trans Pattern Anal Mach Intell”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 14 (Mar. 1992), pp. 239–256.
- [8] Tim Bodenmuller et al. “Tackling multi-sensory 3D data acquisition and fusion”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 2180–2185.
- [9] T.W. Boyer, J. Maouene, and N. Sethuraman. “Attention to body-parts varies with visual preference and verb-effector associations.” In: *Cognitive Processing*. 2017.
- [10] John W. Boyse. “Interference Detection Among Solids and Surfaces”. In: *Commun. ACM* 22.1 (1979), pp. 3–9.
- [11] Matteo Bregonzio, Shaogang Gong, and Tao Xiang. “Recognising action as clouds of space-time interest points”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 1948–1955.
- [12] Minjie Cai, Kris M. Kitani, and Yoichi Sato. “Understanding Hand-Object Manipulation with Grasp Types and Object Attributes”. In: *Proceedings of Robotics: Science and Systems*. AnnArbor, Michigan, June 2016.

- [13] Fude Cao and Qinghai Bao. “A Survey On Image Semantic Segmentation Methods With Convolutional Neural Network”. In: *2020 International Conference on Communications, Information System and Computer Engineering (CISCE)*. 2020, pp. 458–462.
- [14] Z. Cao et al. “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).
- [15] Zhe Cao et al. “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *CVPR*. 2017.
- [16] Ee Chen and Darius Burschka. “Object-Centric Approach to Prediction and Labeling of Manipulation Tasks”. In: May 2018, pp. 6931–6938.
- [17] P. Corke, D. Strelow, and S. Singh. “Omnidirectional visual odometry for a planetary rover”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 4. 2004, 4007–4012 vol.4.
- [18] Thomas H. Cormen et al. *Introduction to Algorithms*. 2nd. The MIT Press, 2001. ISBN: 0262032937.
- [19] Davison. “Real-time simultaneous localisation and mapping with a single camera”. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1403–1410 vol.2.
- [20] A. J. Davison and D. W. Murray. “Simultaneous localization and map-building using active vision”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 865–880.
- [21] A. J. Davison et al. “MonoSLAM: Real-Time Single Camera SLAM”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067.
- [22] P. Dollar et al. “Behavior recognition via sparse spatio-temporal features”. In: *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. 2005, pp. 65–72.
- [23] T. Drummond and R. Cipolla. “Real-time visual tracking of complex structures”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 932–946.
- [24] Tom Drummond and Roberto Cipolla. “Application of Lie Algebras to Visual Servoing”. In: *International Journal of Computer Vision* 37 (June 2000), pp. 21–41.
- [25] Z. Duric, J.A. Fayman, and E. Rivlin. “Function from motion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (1996), pp. 579–591.
- [26] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110.
- [27] A. Elfes. “Using occupancy grids for mobile robot perception and navigation”. In: *Computer* 22.6 (1989), pp. 46–57.
- [28] F. Endres et al. “An evaluation of the RGB-D SLAM system”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 1691–1696.

-
- [29] Litong Fan et al. “A survey on multiple object tracking algorithm”. In: *2016 IEEE International Conference on Information and Automation (ICIA)*. 2016, pp. 1855–1862.
- [30] Hao-Shu Fang et al. “Pairwise Body-Part Attention for Recognizing Human-Object Interactions: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X”. In: Sept. 2018, pp. 52–68. ISBN: 978-3-030-01248-9.
- [31] Gunnar Farneback. “Two-Frame Motion Estimation Based on Polynomial Expansion.” In: *SCIA*. Ed. by Josef Bigün and Tomas Gustavsson. Vol. 2749. Lecture Notes in Computer Science. Springer, 2003, pp. 363–370. ISBN: 3-540-40601-8.
- [32] P. Fieguth and D. Terzopoulos. “Color-based tracking of heads and other mobile objects at video frame rates”. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1997, pp. 21–27.
- [33] Roman Filipovych and Eraldo Ribeiro. “Recognizing primitive interactions by exploring actor-object states”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–7.
- [34] M.A. Fischler and R.C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782.
- [35] F. Fraundorfer and D. Scaramuzza. “Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications”. In: *IEEE Robotics Automation Magazine* 19.2 (2012), pp. 78–90.
- [36] Guillermo Garcia-Hernando et al. “First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 409–419.
- [37] Shaogang Gong, M. Walter, and A. Psarrou. “Recognition of temporal structures: Learning prior and propagating observation augmented densities via hidden Markov states”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. 1999, 157–162 vol.1.
- [38] Mohit Goyal et al. “Human Hands as Probes for Interactive Object Understanding”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 3283–3293.
- [39] G. Grisetti et al. “A Tutorial on Graph-Based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.
- [40] A. Gupta, A. Kembhavi, and L. S. Davis. “Observing Human-Object Interactions: Using Spatial and Functional Compatibility for Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.10 (2009), pp. 1775–1789.
- [41] Gregory D. Hager and Kentaro Toyama. “X Vision: A Portable Substrate for Real-Time Vision Applications”. In: *Computer Vision and Image Understanding* 69 (1996), pp. 23–37.
- [42] I. Haritaoglu, D. Harwood, and L.S. Davis. “W/sup 4/: real-time surveillance of people and their activities”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 809–830.
-

- [43] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.
- [44] D. M. Helmick et al. “Slip compensation for a Mars rover”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 2806–2813.
- [45] Peter Henry et al. “RGB-D Mapping: Using Kinect-Style Depth Cameras for Dense 3D Modeling of Indoor Environments”. In: *International Journal of Robotic Research - IJRR* 31 (Apr. 2012), pp. 647–663.
- [46] Alexander Hermans, Georgios Floros, and Bastian Leibe. “Dense 3D semantic mapping of indoor scenes from RGB-D images”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2631–2638.
- [47] Heiko Hirschmuller. “Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information”. In: *Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR '05. USA: IEEE Computer Society, 2005, pp. 807–814. ISBN: 0769523722.
- [48] Heiko Hirschmuller. “Stereo Processing by Semiglobal Matching and Mutual Information”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 328–341.
- [49] Berthold K.P. Horn and Brian G. Schunck. *Determining Optical Flow*. Tech. rep. USA, 1980.
- [50] Armin Hornung et al. “OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees”. In: *Autonomous Robots* (2013).
- [51] G. Hu et al. “A robust RGB-D SLAM algorithm”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1714–1719.
- [52] Jian-Fang Hu et al. “Jointly learning heterogeneous features for RGB-D activity recognition”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5344–5352.
- [53] Michael Isard and Andrew Blake. “CONDENSATION – conditional density propagation for visual tracking”. In: *International Journal of Computer Vision* 29 (1998), pp. 5–28.
- [54] Neil Johnson and David Hogg. “Learning the distribution of object trajectories for event recognition”. In: *Image and Vision Computing* 14.8 (1996). 6th British Machine Vision Conference, pp. 609–615. ISSN: 0262-8856.
- [55] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), p. 35.
- [56] N. Karlsson et al. “The vSLAM Algorithm for Robust Localization and Mapping”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. 2005, pp. 24–29.
- [57] Kamaljit Kaur, Rajat Khurana, and Alok Kumar Singh Kushwaha. “Deep survey on visual object tracking in surveillance environment”. In: *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*. 2018, pp. 1–6.

-
- [58] Yan Ke, R. Sukthankar, and M. Hebert. “Efficient visual event detection using volumetric features”. In: *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*. Vol. 1. 2005, 166–173 Vol. 1.
- [59] Kourosh Khoshelham. “Accuracy analysis of Kinect depth data”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVIII-5/W12* (May 2012).
- [60] Dong H. Kim and Rae-Hong Park. “Analysis of quantization error in line-based stereo matching”. In: *Pattern Recognition* 27.7 (1994), pp. 913–924. ISSN: 0031-3203.
- [61] Georg Klein and David Murray. “Full-3D Edge Tracking with a Particle Filter”. In: Jan. 2006, pp. 1119–1128.
- [62] S. A. Krim, O. A. A. Elaziz, and R. Chatila. “A computationally efficient EKF-vSLAM”. In: *2008 16th Mediterranean Conference on Control and Automation*. 2008, pp. 511–516.
- [63] Lei Lai et al. “3D Semantic Map Construction System Based on Visual SLAM and CNNs”. In: *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. 2020, pp. 4727–4732.
- [64] John J. Leonard, Hugh F. Durrant-Whyte, and Ingemar J. Cox. “Dynamic Map Building for an Autonomous Mobile Robot.” In: *In Proc. IEEE Int. Workshop on Intelligent Robots and Systems*. (1990), pp. 286–298.
- [65] Vincent Lepetit and Pascal Fua. 2005.
- [66] S. Leutenegger, M. Chli, and R. Y. Siegwart. “BRISK: Binary Robust invariant scalable keypoints”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2548–2555.
- [67] Li-Jia Li and Li Fei-Fei. “What, where and who? Classifying events by scene and object recognition”. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8.
- [68] Y. Li and S. Lang. “A Stereo-Based Visual-Inertial Odometry for SLAM”. In: *2019 Chinese Automation Congress (CAC)*. 2019, pp. 594–598.
- [69] Yang Liu, Qingchao Chen, and Andrew Zisserman. “Amplifying Key Cues for Human-Object-Interaction Detection”. In: *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV*. Glasgow, United Kingdom: Springer-Verlag, 2020, pp. 248–265. ISBN: 978-3-030-58567-9.
- [70] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691.
- [71] Bruce D. Lucas and Takeo Kanade. “An iterative image registration technique with an application to stereo vision”. In: *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. IJCAI’81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.
- [72] Yi Ma et al. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003. ISBN: 0387008934.
-

- [73] G. Martinez. “Improving the Robustness of a Direct Visual Odometry Algorithm for Planetary Rovers”. In: *2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. 2018, pp. 1–6.
- [74] S. J. McKenna and S. Gong. “Gesture Recognition for Visually Mediated Interaction using Probabilistic Event Trajectories”. In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 1998, pp. 50.1–50.10. ISBN: 1-901725-04-9.
- [75] Donald Meagher. “Geometric Modeling Using Octree-Encoding”. In: *Computer Graphics and Image Processing* 19 (June 1982), pp. 129–147.
- [76] M. Montemerlo and S. Thrun. “Simultaneous localization and mapping with unknown data association using FastSLAM”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1985–1991 vol.2.
- [77] Richard A. Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 2011, pp. 127–136.
- [78] J. Nieto et al. “Real time data association for FastSLAM”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. 2003, 412–418 vol.1.
- [79] D. Nister, O. Naroditsky, and J. Bergen. “Visual odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*. Vol. 1. 2004, pp. I–I.
- [80] B. C. Ooi. “Spatial kd-Tree: A Data Structure for Geographic Database”. In: *BTW*. 1987.
- [81] B. C. Ooi, Ken J. McDonnell, and R. Sacks-Davis. “Spatial kd-tree: an indexing mechanism for spatial databases”. In: 1987.
- [82] Alan V. Oppenheim and Ronald W. Schaffer. *Discrete-Time Signal Processing*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0131988425.
- [83] Ben Packer, Kate Saenko, and Daphne Koller. “A combined pose, object, and feature model for action understanding”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1378–1385.
- [84] G. Panin. *Model-based Visual Tracking: The OpenTL Framework*. IT Pro. Wiley, 2011. ISBN: 9781118002131.
- [85] Chavdar Papazov and Darius Burschka. “Stochastic global optimization for robust point set registration”. In: *Computer Vision and Image Understanding* 115.12 (2011). Special issue on Optimization for Vision, Graphics and Medical Imaging: Theory and Applications, pp. 1598–1609. ISSN: 1077-3142.
- [86] P. Pérez et al. “Color-Based Probabilistic Tracking”. In: *Computer Vision — ECCV 2002*. Ed. by Anders Heyden et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 661–675. ISBN: 978-3-540-47969-7.
- [87] R. Polana. “Temporal texture and activity recognition.” PhD thesis. University of Rochester., 1994.

-
- [88] M. Pupilli and A. Calway. “Real-Time Camera Tracking Using a Particle Filter”. In: Jan. 2005, pp. 50.1–50.10. ISBN: 1-901725-29-4.
- [89] Balasubramanian Raman et al. “Quantization Error in Stereo Imaging systems”. In: *Intern. J. Computer Math* 79 (June 2002), pp. 671–691.
- [90] Juan Ramirez and Darius Burschka. “Dynamic 3D Mapping - Visual Estimation of Independent Motions for 3D Structures in Dynamic Environments”. In: vol. 2. Jan. 2013.
- [91] Cen Rao and Mubarak Shah. “View-Invariance in Action Recognition”. In: vol. 2. Feb. 2001, pp. II–316. ISBN: 0-7695-1272-0.
- [92] Cen Rao, A. Yilmaz, and Mubarak Shah. “View-Invariant Representation and Recognition of Actions”. In: *International Journal of Computer Vision* 50 (2002), pp. 203–226.
- [93] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. cite arxiv:1506.02640. 2015.
- [94] Rivlin, Dickinson, and Rosenfeld. “Recognition by functional parts [function-based object recognition]”. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1994, pp. 267–274.
- [95] T. Ro, A. Friggel, and N. Lavie. “Attentional biases for faces and body parts”. In: *Visual Cognition* 15(3). 2007, pp. 322–348.
- [96] J.J. Rodriguez and J.K. Aggarwal. “Quantization error in stereo imaging”. In: *Proceedings CVPR '88: The Computer Society Conference on Computer Vision and Pattern Recognition*. 1988, pp. 153–158.
- [97] Grégory Rogez, James S. Supancic, and Deva Ramanan. “Understanding Everyday Hands in Action from RGB-D Images”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 3889–3897.
- [98] D. Scaramuzza and F. Fraundorfer. “Visual Odometry [Tutorial]”. In: *IEEE Robotics Automation Magazine* 18.4 (2011), pp. 80–92.
- [99] Alexander Schaub, Juan Carlos Ramirez de la Cruz, and Darius Burschka. “Autonomous Parking using a Highly Maneuverable Robotic Vehicle”. In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 2640–2645. ISSN: 1474-6670.
- [100] Philip J. Schneider and David Eberly. *Geometric Tools for Computer Graphics*. USA: Elsevier Science Inc., 2002. ISBN: 1558605940.
- [101] Dandan Shan et al. “Understanding Human Hands in Contact at Internet Scale”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9866–9875.
- [102] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. 2014.
- [103] Nathan Silberman et al. “Indoor Segmentation and Support Inference from RGBD Images”. In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part V. ECCV'12*. Florence, Italy: Springer-Verlag, 2012, pp. 746–760. ISBN: 9783642337147.
- [104] Tomas Simon et al. “Hand Keypoint Detection in Single Images using Multiview Bootstrapping”. In: *CVPR*. 2017.
-

- [105] Arnold W. M. Smeulders et al. “Visual Tracking: An Experimental Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1442–1468.
- [106] Jan Smisek, Michal Jancosek, and Tomas Pajdla. “3D with Kinect”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 1154–1160.
- [107] Lindsay I Smith. *A tutorial on principal components analysis*. Tech. rep. Cornell University, USA, Feb. 2002.
- [108] H. W. Sorenson. “Least-squares estimation: from Gauss to Kalman”. In: *IEEE Spectrum* 7.7 (1970), pp. 63–68.
- [109] L. Stark and K. Bowyer. “Generic recognition through qualitative reasoning about 3-D shape and object function”. In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991, pp. 251–256.
- [110] A.N. Steinberg. “Data fusion system engineering”. In: *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on*. Vol. 1. July 2000, MOD5/3–MOD510 vol.1.
- [111] Josephine Sullivan and Stefan Carlsson. “Recognizing and Tracking Human Action”. In: *Proceedings of the 7th European Conference on Computer Vision-Part I. ECCV '02*. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 629–644. ISBN: 3540437452.
- [112] S. Takezawa, D. C. Herath, and G. Dissanayake. “SLAM in indoor environments with stereo vision”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 2. 2004, 1866–1871 vol.2.
- [113] Bugra Tekin, Federica Bogo, and Marc Pollefeys. “H+O: Unified Egocentric Recognition of 3D Hand-Object Poses and Interactions”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4506–4515.
- [114] Sinisa Todorovic and Narendra Ahuja. “Learning subcategory relevances for category recognition”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8.
- [115] Jiang Wang et al. “Learning Actionlet Ensemble for 3D Human Action Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.5 (2014), pp. 914–927.
- [116] Limin Wang et al. “Object-Scene Convolutional Neural Networks for event recognition in images”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2015, pp. 30–35.
- [117] Wei Wang et al. “Textured/textureless object recognition and pose estimation using RGB-D image”. In: *J. Real Time Image Process.* 10.4 (2015), pp. 667–682.
- [118] Zhou Wang and Alan C. Bovik. “Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”. In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117.

- [119] Ping Wei et al. “Modeling 4D Human-Object Interactions for Event and Object Recognition”. In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 3272–3279.
- [120] Shih-En Wei et al. “Convolutional pose machines”. In: *CVPR*. 2016.
- [121] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. rep. Chapel Hill, NC, USA, 1995.
- [122] Lu Xia, Chia-Chih Chen, and J. K. Aggarwal. “View invariant human action recognition using histograms of 3D joints”. In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2012, pp. 20–27.
- [123] Hao Xing et al. “Robust Event Detection based on Spatio-Temporal Latent Action Unit using Skeletal Information”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 2941–2948.
- [124] Yang Cheng, Mark Maimone, and Larry Matthies. “Visual odometry on the Mars Exploration Rovers”. In: *2005 IEEE International Conference on Systems, Man and Cybernetics*. Vol. 1. 2005, 903–910 Vol. 1.
- [125] Jianyu Yang, Xiaolong Zhou, and Youfu Li. “On trajectory segmentation and description for motion recognition”. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2015, pp. 345–350.
- [126] Yezhou Yang et al. “Grasp type revisited: A modern perspective on a classical feature for vision”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 400–408.
- [127] Bangpeng Yao and Li Fei-Fei. “Recognizing Human-Object Interactions in Still Images by Modeling the Mutual Context of Objects and Human Poses”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.9 (2012), pp. 1691–1703.
- [128] Cha Zhang and Zhengyou Zhang. “Calibration Between Depth and Color Sensors for Commodity Depth Cameras”. In: July 2011, pp. 1–6. ISBN: 978-3-319-08650-7.
- [129] Cheng Zhang et al. “Large-Scale 3D Semantic Mapping Using Monocular Vision”. In: *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*. 2019, pp. 71–76.
- [130] Z. Zhuang and O. D. Faugeras. “Three-dimensional motion computation and object segmentation in a long sequence of stereo frames.” In: *Intern. J. Comput* (1992), pp. 211–241.