Hand-pose-based Learning from Demonstration of Manipulation Tasks

handed in
BACHELOR'S THESIS

stud. Zeju Qiu

born on the 24.08.1997
living in:
Buergermeister-Haidacher-Strasse 4
82140 Olching
Tel.: 0175 3429125

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	Thomas Eiband, Shile Li
Start:	01.11.2019
Intermediate Report:	17.02.2020
Delivery:	20.03.2020

In your final hardback copy, replace this page with the signed exercise sheet.

Before modifying this document, READ THE INSTRUCTIONS AND GUIDELINES!

Abstract

A Learning from Demonstration algorithm has been created to transfer task knowledge to the robot by observing human demonstration with a depth camera. A two-tiered segmentation algorithm that employs Hidden Markov Model has been developed to extract motion primitives from the demonstration and gives them semantic meaning. With the help of the segmentation, the functionality of this program has been extended from direct reproduction towards a more optimized and generalized execution of the demonstration.

Contents

Chapter 1

Introduction

1.1 Introduction

Learning from Demonstration (**LfD**) is a popular technique to transfer task knowledge to a robot. The idea of LfD is to teach the robot new tasks without manually writing codes, but by showing the robot how to perform the task. Commonly, a teacher performs multiple demonstrations and generates a data set. A LfD algorithm tries to find a strategy based on the data set to reproduce the demonstration. The problem statement in LfD can be formulated as follows: the world consists of states and actions, the desired LfD strategy enables the learner to select an action based on the current state [ACVB09]. In traditional robot programming, a human programmer has to manually implement the desired behavior with codes: specifies how the robot should perform the action and how the robot respond if a certain scenario occurs. This way of creating robot applications has some major drawbacks. Firstly, the robot programming requires expert knowledge. Training is always required and therefore hinders robot programming to become accessible for all users. Secondly, even small reconfigurations might require time-consuming re-programming and the modification of the whole application if the circumstance changes. With the methods of LfD on the other hand, we can create a programming framework with which the creation of robot programs will become more intuitive and the created applications will be more generalizable.
So the question arises: how to create a programming paradigm so that even non-expert users can create robot applications. One intuitive way is to use a camera to observe and learn from the demonstration since we as humans also use our eyes to observe and learn new activities. This kind of demonstration technique is called shadowing. Shadowing is where the robot learner uses its own sensing device to record the demonstration and tries to match the motion of the demonstrator [ACVB09].
How to solve the correspondence problem? How to make the robot reproduce the human demonstration? This so-called correspondence problem can be solved by identifying a mapping function between the teacher (demonstrator's hand) and learner (robot gripper) [ACVB09]. The mapping function only has to be one-sided, because

there is no information transferred back from the learner.
With a recent hand-pose estimation algorithm [LL19], a human could directly demonstrate manipulation skills to a robot with hand glove and the robot learner is able to receive the demonstration data with a depth sensor. Furthermore, the demonstration involves object handling and therefore an AR tag tracking library could be used to track the position and orientation of the objects. With these methods, a mapping between human's hand and robot gripper can be established.
After the demonstration has been successfully reproduced in the simulator we want to separate the demonstration into several segments and label each segment with a skill name. It is further desirable for the robot to be more adaptive to the environment. For example, after the robot has been shown a demonstration and the object positions have been changed, the robot should be still able to perform the same set of actions. Task-parameterized models of movement primitives are often used in this context. The advantage is that the robot can automatically adjust its movements to new situations. Some standard approaches like Dynamic Movement Primitives (**DMP**) or Task-parameterized Gaussian mixture model (**TP-GMM**) are helpful to implement this. For example, we want to use the Dynamic Movement Primitive (**DMP**) which encodes a movement primitive into a second-order, non-linear dynamical system. We can think of it as employing two systems: a DMP system to do trajectory planning and a real system to execute. In our case, we want to plan the trajectory of our robot hand. In our DMP system we plan a path for the robot to follow, the result is a set of "forcing terms" as variables calculated to the DMP [Cal16]. While we can generate DMP with only one demonstration, we will need more demonstrations to generate TP-GMM. The TP-GMM is more generalized and can deliver better trajectories to adapt to the new object position.

Aim of my research: is to create a framework, which uses a motion sensing device as data input to teach manipulation skills, e.g. pick and place, to a robot via demonstration. The solution should further has the following qualities:

- **intuitive**: the usage of this framework should be easy to learn and reproduce
- **robust**: functions under unknown scenarios and can
- **extensible**: the user should be able to use the routines to extend the functionalities (e.g. define more skills)
- **reconfigurable**: if the object changes the position, the robot should still be able to perform the same task without any disturbances or collisions.

1.2 Problem statement

In this paper, the following aspects will be closer examined and they are also the major contributions of this work:

- How to implement learning by demonstration with a vision system: setting up the demonstration in a robot simulator which represents the demonstration in reality.
- How to solve the correspondence problem, i.e. how to find the mapping function from the demonstrator's hand to the robot end-effector meaningfully. The robot gripper is, antithetic to human hands, symmetric.
- How to segment the trajectory reasonably. After the segmentation, each segment should have a semantic meaning (skill).
- How to improve the optimize the execution by applying different motion planning scheme during the task execution to optimize the demonstration: ranging from simple linear movements (point-to-point motion) to Dynamic Movement Primitives (DMP).
- A programming framework for non-expert users

1.3 Procedure

in the following is the procedure depicted, which has been followed during this thesis:

1) Implementation of the hand-pose estimation algorithm by installing and modifying all the necessary ROS packages and writing launch files.
2) Observing human demonstration: Track the human hand pose and object poses in the workspace using ROS packages and a Kinect sensor.
3) Modify data; filtering, calculate transformation matrices, create mapping between robot gripper and hand
4) Training HMM models with multiple demonstrations and create segmentation procedure
5) Extracting Movement Primitives (MP) from human demonstration: Segment the demonstration data by identifying if an object has been grasped or released. The used segmentation mechanism is based on object proximity and without explicit human commands [6]. The robot should independently recognize e.g. if the gripper has successfuly grasped the object.
6) Define and parameterize robot skills using the segmented data.
7) Retargeting from human to robot: Define a mapping between finger poses (i.e. the finger positions when grasping the object) and robot gripper fingers in order to grasp different objects with the robot
8) Execution of learned MPs: Reproduce the demonstrated task within a robot simulator (**V-REP**)
9) Execution of the trajectory generated from DMP
10) Create a programming by demonstration framework

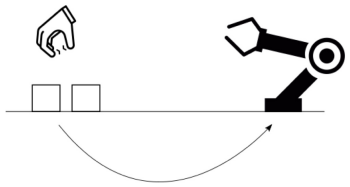


Figure 1.1: Introduction: Learning by demonstration

Chapter 2

State of the Art

In the following section the problem stated in this paper is been divided into several aspects and examined individually. For each aspect, it introduces and explains approaches in the past, some of them inspired this framework and some of them using a different methodology to solve a similar problem. A comparison and evaluation of other approaches with my proposed framework will expose the limitations and advantages of this framework.

2.1 Learning from Demonstration

Learning from demonstration is a popular technique to transfer task knowledge to the robot. The teacher creates demonstrations with data sets [ACVB09]. Although the basic principle of learning from demonstration is given, the approach can vary distinctively.

In this paper [CSFL19] the author presents a framework, which uses an attentional system and methods from kinesthetic teaching to teach the robot to execute demonstrated tasks. Kinesthetic teaching is a method to transfer tasks knowledge by physically guiding the robot to the desired pose. The proposed framework has two major components: a Robot Manager (RM), which is responsible for teaching and the execution of tasks, and an attentional system, which provides the possibility to coordinate the execution of complex tasks and the monitoring of human activities.

In this paper [DXW+16] the author presents a motion sensing-based framework to create robotic applications. This framework uses a Kinect sensor as motion-sensing device to perform gesture recognition and manipulates the robot. The proposed framework can track gestures including poses and joint angles, to perform the mapping from human demonstration with robot manipulation and is compatible with various hardware devices, including the motion sensor and the robot. In the core of the framework, the author designs ROS (robot operation system) modules to handle several tasks, including information and model management, control and visualiza-

tion.

In this paper [DZML12] the author uses a Kinect sensor to track the demonstrator's hand movement to achieve teleoperation. This approach sends whole task information to the robot, instead of sending limited commands like gestures. After processing RGB and depth images from the Kinect sensor, including the separation of the forearm, the position of thumb tip and index finger tip is used to generate a small coordinate system. This hand coordinate system is used to calculate 3D anatomical position and orientation of the demonstrator's hand.

In this paper [LC15] the author focuses on hand gesture recognition to teach the robot pick-and-place tasks. The author defines seven types of hand gestures and each one of them represents a basic skill (e.g. transport, grasp, release). With a CNN-based gesture recognition system, the robot can understand the skill sequence from a demonstration. After extracting the robot movement primitives, the author employs the Extensible Agent Behavior Specification Language (XABSL) programming platform to realize pick-and-place tasks.

2.2 Human Motion Segmentation

In this paper [CSFL19] the author applies a simple segmentation mechanism that comprises distance measuring and explicit human commands. Each object is attached with a proximity area, i.e. a sphere, and each time the robot enters or exits this area, the trajectory will be segmented. Human commands are also used to control the robotÂ´s components like the gripper. This segmentation method distinguishes between two control mode: while the robot is within the proximity area trajectory will be calculated more carefully (e.g. with Dynamic Movement Primitives) to guarantee precise reproduction of the demonstration, on the other hand, if the robot is far away from the object, a simple linear point-to-point motion is sufficient to fulfill the task.

In this paper [LK13] the author employs a two-stage recognition process to perform segmentation. One of the challenges in the segmentation procedure is the variability in human motions, i.e. the same motion can differ significantly between every demonstration, therefore, the author uses a probabilistic approach with HMM (Hidden Markov Model) to identify segments. The author trains different HMM models beforehand with exemplary data and performs template matching with new data. Further, the author monitors zeros-velocity crossings (ZVC), which will occur if the joint changes movement direction, to determine the framing windows for the template matching.

2.3 Robot skills

In this paper [PNA+16] the author introduces a task-level programming paradigm that enables non-expert users to create robot applications. The author divides the programming into three layers: the **primitive**, **skill** and **task** layer. The **primitive** layer consists of primitive motions like move or close gripper. **Skill** is defined as the combination of primitives and includes movements like **pick** or **place**. **Task** is always related to a specific goal, for example assembly or machine tending. After establishing the structure, the author argues that the skill should further have the following properties: parametric (always performs the defined action), able to pre-estimate whether the skill can be executed and able to evaluate the success after execution.

2.4 Summary

Our mapping function is inspired by this paper's idea to create a hand coordinate system [DZML12]. However, because of the novel hand-pose estimation algorithm [LL19] we can create a much more robust coordinate system using our index finger-tip, index finger ankle, and thumb tip. The execution of the demonstration in the simulator shows how stable correct our mapping function is.
In our approach the skill formalization is also object-centered, i.e. a skill is always defined with an object, The spatial relationship and interaction with an object are used to characterize a skill. However, unlike the approach introduced in [PNA+16] the movements are not further divided into **primitives** or **tasks**. I believe the separation into different layers will further increase the difficulty of the segmentation process.
Our segmentation process has been inspired by the two-tiered recognition process from [LK13]. However, we are integrating the concept of skills into the segmentation process, unlike the author who only monitors the zeros-velocity crossings but does not give the segments a semantic meaning. I believe by labeling segments with the concept of skills we can further improve the demonstration by incorporating expert knowledge.

Chapter 3

Main Part

3.1 Solution design

In the block diagram 3.1 there is the structure of the technical implementation depicted. Generally the solution can be divided into five sub processes: data generation, data processing, mapping, segmentation and execution.

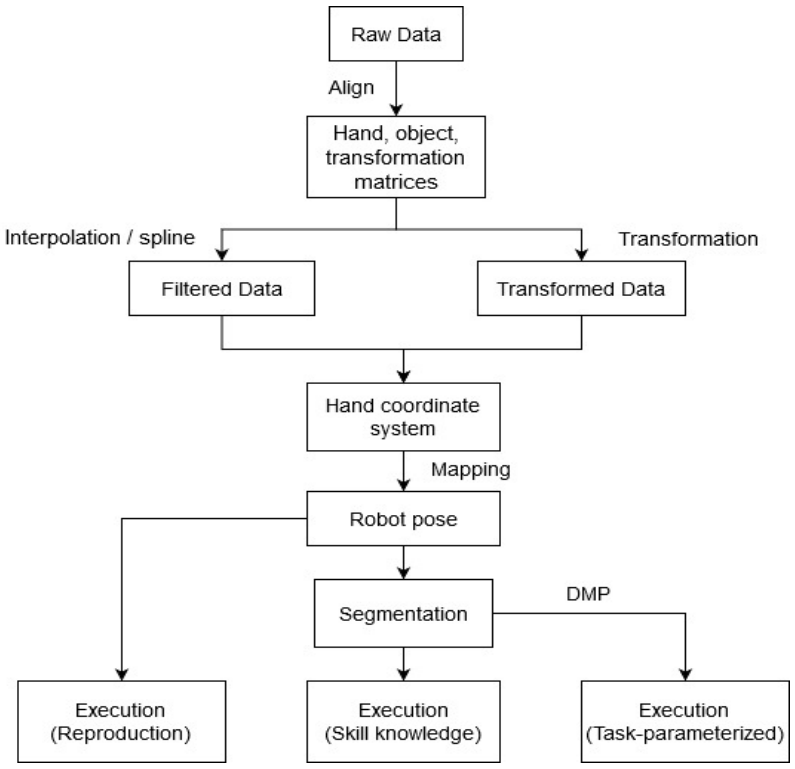


Figure 3.1: Block diagram: structure

Framework usage This section is designed to provide the reader with information about how to use the framework and how the solution is implemented. The reader will have a better understanding of the train of thought. In addition, a real demonstration has been made and this example is intended to give the reader a more vivid understanding of the implementation.

3.2 Environment

In this section, the mainly used tools and environments will be introduced to provide the reader the possibility to understand the process and reproduce the work.

3.2.1 ROS

Robot Operating System (ROS) is a modular, tools-based framework, which simplifies the development of robot applications by providing a structured communication layer. For example, the hand-pose estimation algorithm relies on ROS and programmer in general can profit from its open-sourced packages, libraries and tools. Some major components, which are used within this paper, are introduced here [QCG+09]:

- **Node:** a single computation process
- **Topic:** communication between nodes take place via topics
- **Message:** a strictly defined data structure
- **Launch:** a tool to start multiple nodes with parameters
- **Package:** a program, which can contain source code, launch files, message definitions etc. documentation

Ar_stack_alvar is a AR tag tracking library, which enables the user to generate AR tags and to identify and track the pose of the generated AR markers [NS]. A launch file has been created to pass the program the necessary information about the marker size, the camera calibration and the relative output frame to make a precise identification and calculation of the markers' pose.

dl_pose_estimation [LL19] Deep learning hand pose estimation algorithm using Permutation Equivariant Layer (PEL) to estimate the hand pose. After informations about indiviual points have been gathered, it uses a voting-based scheme to merge them into complete hand pose (21 points). **More Info!** The benefit of this algorithm is it provides the information about fingertips and finger ankles, which e.g. enables our program to calculate hand pose and hand gesture. This has proven to be a major advantage in the following mapping and segmentation process.

record_position is a package, which is created within this paper to provide the spatial pose and time information of the markers and the hand. This program

records the coordinate transformation from object frame to camera frame, the position of hand and all the objects (in camera frame). Three "callback" functions are implemented to subscribe the marker information (and modify them. The time information is calculated by setting the time stamp of the first recorded object as "start time" (value 0) and the time information is calculated by the difference between current time and start time converting from nano-seconds into seconds. For each time stamp, all the information about markers are saved. After the recording has finished, the program modifies the data into a form, which can be used directly to generate ".csv"-files.

3.2.2 MATLAB

The main part of this work has been implemented in **MATLAB R2019b**. **MATLAB** is a programming plattform designed to analyze data and develop algorithms. The reason for choosing **MATLAB** as the main programming platform is because it provides powerful tools to facilitate the data processing process and is compatible with the robot simulator **CoppeliaSim**. Several toolboxes were used to process the data, before it can be passed to the robot simulator: **Robotics System Toolbox:** which enables the program to convert between quaternions, transformation matrices and Euler anlges (quat2eul, makehgtform etc). **Curve fitting toolbox:** was used to generate spline functions and calculate the values (e.g. cscvn) and **Signal Processing Toolbox** was used to filter the data points. Further, a library called SMOOTHN [Gar] has been employed to generate a fast, automatized and robust discretized spline smoothing for data of three dimensions. One advantage of SMOOTHN is it can deal with missing (NaN) values, which often occur during the recording.

3.2.3 CoppeliaSim/V-REP robot simulator

V-REP, or **CoppeliaSim** is a versatile, scalable, general-purpose robot simulation framework and can simulate complete demonstrations, with embedded components including robots, grippers, vision sensors, force sensors, cameras etc. Our experiment includes the following components:

- a **KUKA LBR iiwa 14** robot
- a **Baxter gripper**
- a resizable table
- several items (primitive shapes, like cuboids)

V-REP programms can be implemented as **embedded scripts**, **add-ons**, **plugins**, **ROS nodes** and **remote API clients** [RSF13]. In this paper, the simulation is controlled via **remote API clients**, because it can be embedded as footprint code

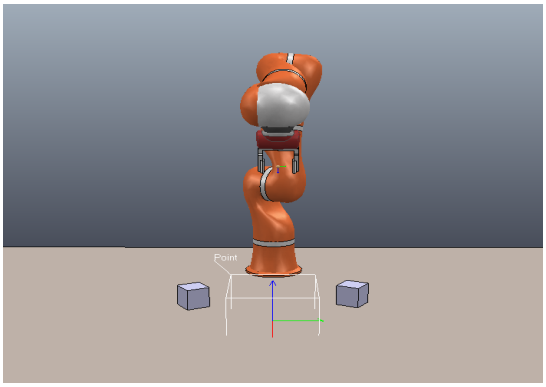


Figure 3.2: Demonstration: simulator

Figure 3.3: Demonstration: real

in other programms like Matlab programms and is therefore better integrated in our framework. The robot in the simulation can be easily controlled by calling matlab scripts with remote API functions (e.g. simxSetObjectPosition, simxSetObjectOrientation). However, using this kind of control method has the drawback that only a limited remote API functions are available and the delays in data transmission increases the duration of task execution.
Two scenes called "LBR4p_demo_scenario.ttt" and "LBR4p_demo_scenario_DMP.ttt" are used to run the simulation. The first scene is used to display the reproduced and segmented version of the demonstration, while the latter one is designed for DMP. The robot is mounted on a table and all objects are on the table to imitate the demonstration in the real world (see 3.2).

3.3 Data Generation

The first step is to perform movements while recording with a depth camera to generate demonstrations. In this thesis, an **ASUS RGB and Depth sensor** is used to do the recording. After the demonstration 3 files are generated separately: **hand_trajectory.csv**, **object_trajectory.csv** and **coordinate_frame.csv**, after importing these files three matrices are created **A**, **B** and **C**. These three matrices build the core of this framework and all the functions and subroutines, including the generation of DMP, the calculation of transformation matrices, the generation of simulation, the generation of HMM templates etc., are all based on these three matrices:

$$\mathbf{A}_{i,j} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i,1} & a_{i,2} & \cdots & a_{i,j} \end{pmatrix}, \ 1 \leq j \leq 105$$

$$\mathbf{B}_{k,l} = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,l} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k,1} & b_{k,2} & \cdots & b_{k,l} \end{pmatrix} \ , \ 1 \leq l \leq 40$$

$$\mathbf{C}_{m,n} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{pmatrix} \ , \ 1 \leq n \leq 64$$

Note that **A** has 105 columns and records the positions of all 21 hand points (5 columns per point: time stamp, numbering, coordinates) for each time stamp. **B** has 40 columns and records the positions of the markers numbering from 0 to 7 (5 columns per marker: time stamp, numbering, coordinates). If one marker has not been detected, then the belonging block is 0. C has 64 columns and records the transformation matrix ${}^{c}T_{o}$ of the markers numbering from 0 to 7 (8 columns per marker: time stamp, translation, quaternion). If one marker has not been detected, then the belonging block is 0. The user has to manually delete the entire row because zero is not a valid quaternion value to calculate the Euler angles.
Because of the deep learning algorithm that the hand pose estimation algorithm uses the A matrix will always have the least number of measuring points (rows). Therefore, the first step is to align the three matrices. For all time stamps in A, search the counterpart in **B** and **C** (the time stamp with the minimal deviation). After the alignment, A, B and C will have the same number of measuring points (number of rows), i.g. for every time stamp we will have the position of the hand, the position and transformation matrix of each object detected.

Important: Firstly, the minimum number of data samples recorded for the hand must exceeds 30 (because of the usage of **filtfilt-filter**. Secondly, the system expects at least 1 object during execution.

Notes on the demonstration: Several things should be paid with great attention during the demonstration to accelerate the demonstration process and increase the success rate. Firstly, the ambient light. I noticed that the recognition of markers is more stable under a dim environment while the hand pose estimation is more stable under a bright environment. Overall I noticed that the best recording time is in the morning or in the afternoon and natural light is better than artificial light. Another thing that has to be checked is to measure the camera with a spirit level before the recording to make sure that the camera is horizontal. It occurred during the experiment that because of the camera the whole recording was inclined and therefore could not be used.
The demonstrations are recorded with the **SimpleScreenRecorder** to create video

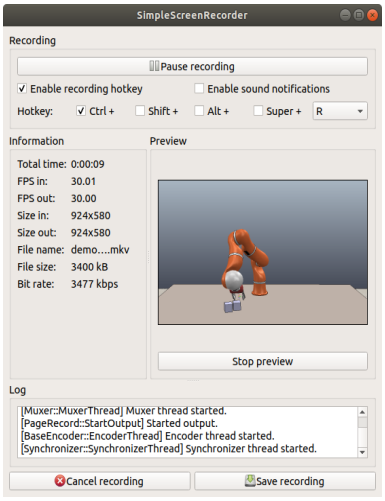


Figure 3.4: Screencast

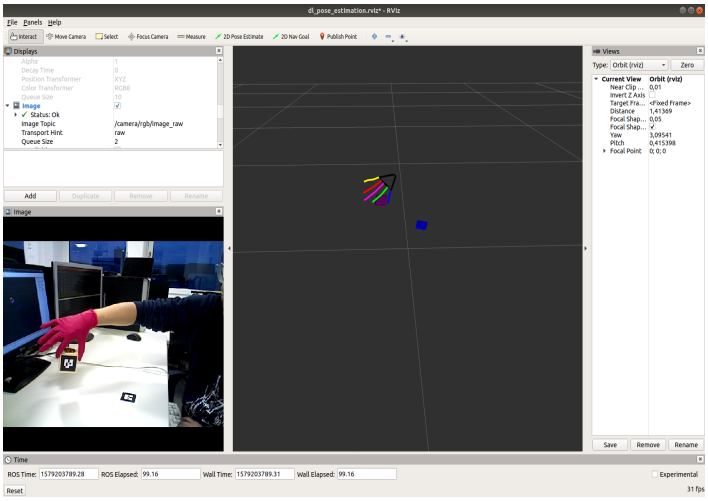


Figure 3.5: RVIZ

files and all the demonstrations data and recordings can be found in the attached CD.

3.4 Data processing

3.4.1 Transformation

Idea: The positions of the objects are in the camera frame and thus cannot be passed into the robot simulator directly. The data has to be converted into a defined reference frame first. There is a marker with the marker id 0 attached on the desk and serves as the reference frame of the system, i.e. the position of all the objects and the hand are described in this reference frame. We also define a point coordinate system in the simulation, which is attached to the table, as our reference frame.
Because the data recorded with the camera are within the left-handed coordinate system, but the frames in the robot simulator CoppeliaSim are right-handed, the conversion between those coordinate system has to be calculated first. The conversion from left handed coordinate system to right handed coordinate system is give by:

$$\begin{bmatrix} \hat{e_x} \\ \hat{e_y} \\ \hat{e_z} \end{bmatrix} = \begin{bmatrix} e_z \\ e_y \\ -e_x \end{bmatrix} \tag{3.1}$$

In the following there is an overview of all the transformations calculated and employed within this thesis:

The **Transformation** is given by:

1) Transformation matrix from object frame to camera frame:

$$ {}_c\mathbf{p} = \begin{bmatrix} {}^cR_o & \mathbf{r} \\ \mathbf{f}^\intercal & w \end{bmatrix} \cdot {}_o\mathbf{p} = {}^cT_o \cdot {}_o\mathbf{p} \tag{3.2} $$

2) Transformation matrix from camera frame to reference frame:

$$ {}^rT_c \subset {}^oT_c \tag{3.3} $$

3) Transformation matrix from hand frame to reference frame:

$$ {}_r\mathbf{p} = \begin{bmatrix} {}^rR_h & \mathbf{r} \\ \mathbf{f}^\intercal & w \end{bmatrix} \cdot {}_h\mathbf{p} = {}^rT_c \cdot {}_h\mathbf{p} \tag{3.4} $$

4) Transformation matrix from object frame to reference frame:

$$ {}_r\mathbf{p} = {}^rT_c \cdot ({}^oT_c)^{-1} \cdot {}_o\mathbf{p} = {}^rT_o \cdot {}_o\mathbf{p} \tag{3.5} $$

5) Transformation matrix from hand frame to object frame:

$$ {}_o\mathbf{p} = ({}^rT_o)^{-1} \cdot {}^rT_h \cdot {}_o\mathbf{p} = {}^oT_h \cdot {}_h\mathbf{p} \tag{3.6} $$

6) Transformation matrix from hand frame to reference frame based on ${}^oT_h$ (DMP):

$$ {}^rT_{\hat{h}} = {}^rT_o \cdot {}^oT_{\hat{o}} \cdot {}^oT_h \tag{3.7} $$

7) Inverse homogeneous transformation matrix:

$$ ({}^aT_b)^{-1} = \begin{bmatrix} ({}^aR_b)^\intercal & -({}^aR_b)^\intercal\mathbf{r} \\ \mathbf{f}^\intercal & w \end{bmatrix} \tag{3.8} $$

where R is the rotation matrix.

As mentioned in the previous section, we have the information about the translation and quaternion from **C**. Quaternion is a mathematical notation to describe the orientation and rotation of objects. To create the transformation matrix, the quaternions have to be converted to Euler angles first. The rotation matrix can be calculated with (3.13). After the rotation matrix has been calculated, it can be cascaded with the translation vector into the **homogeneous transformation matrix**:

$$ {}^aT_b = \begin{bmatrix} {}^aR_b & \mathbf{r} \\ \mathbf{f}^\intercal & w \end{bmatrix} \tag{3.9} $$

which converts a point form the coordinate system b to the coordinate system a.

The resulting transformation matrix is ${}^{c}T_{o}$ (3.2).
As mentioned earlier, the reference system is defined as the coordinate system of the marker 0, i.e. it's transformation matrix from the camera frame to the reference frame is a subset of the object transformation matrix (3.4). Other transformation matrices can be calculated with matrix multiplications.
The transformation matrix ${}^{r}T_{o}$ is important for the simulation: the initial transformation matrix is used to setup the demonstration in the simulator and others are used to set the target pose in the DMP.
The transformation matrix ${}^{r}T_{o}$ is only relevant for the calculation of DMP: after the demonstration has been segmented, the object will be moved to another position. The new target position requires the robot to grasp this object from a new pose (in a reference coordinate system perspective), but still remains the old relation (in a target coordinate system perspective). The new pose of the robot gripper can only be obtained by the multiplication of ${}^{c}T_{o}$ and ${}^{o}T_{h}$.

Additional information: If we want to test if the Euler angles have been calculated successfully we can multiply our general frame with the transformation matrix **R** and determine wether these two matrices are aligned. **R** is defined as the rotation around an arbitrary unit vector $\mathbf{u} = (u_x, u_y, u_z)$ by the angle θ [Col15]. Note the new coordinate system has to be in the origin because this function does not causes translation:

$$R = \begin{bmatrix} \cos\theta + {u_x}^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + {u_y}^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin s\theta \\ u_z u_x(1-\cos\theta) - u_y\sin s\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + {u_z}^2(1-\cos\theta) \end{bmatrix} \tag{3.10}$$

If we pass the calculated Euler angles directly into the robot simulator, we can observe an uncontrolled shaking of the robot gripper and the objects (see the plot 3.6), because the Euler angles jump sporadically from positive to the negative and back again. The gripper tries to follow the Euler angle values and therefore result in the uncontrolled shacking. The Euler angles have to be filtered before passing to the robot simulator. First, the Euler angles have to be consistent in the sign except near zero, this is based upon the assumption that the demonstration takes place gradually, i.e. without sudden twisting or shaking. We can observe the sporadic changing of signs for values around π. Secondly, sporadic variations are considered to be noises during the demonstration and therefore have to be filtered out.

Important: The Euler angles convention will be later explained but it is important to remember that the robot simulator uses the **XYZ** convention to use the Euler angles, therefore, it is important to calculate throughout the framework with the **XYZ** convention, including from quaternion to Euler angles and from rotation matrix to Euler angles.

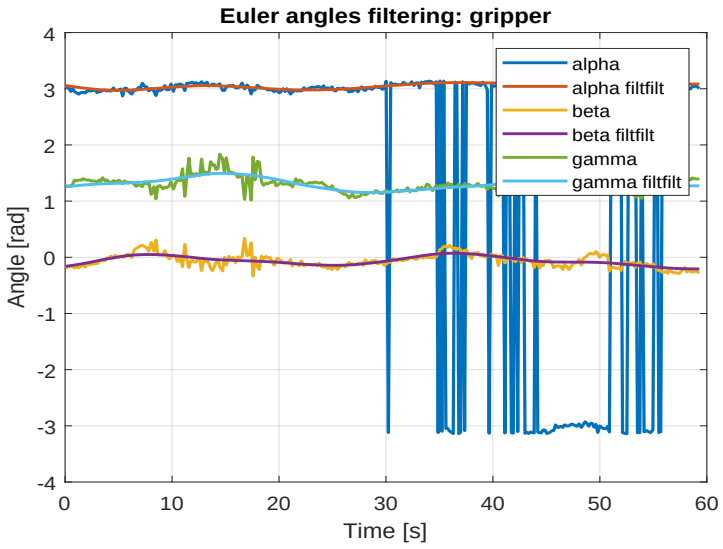


Figure 3.6: Euler angles filtering gripper

3.4.2 Mapping

Idea: The correspondence problem can be solved by identifying a mapping function between the teacher (demonstrator's hand) and learner (robot gripper) [ACVB09]. Our found mapping function is onesided, i.e. the information can only be transferred from the teacher to the learner. In order to control the robot in the simulator, the pose of the robot gripper has to be calculated beforehand. The idea is that the robot gripper in the simulator should reproduce the human hand demonstration as precise as possible. Because a pose normally consists of the the position and orientation, a hand coordinate frame has been defined to calculate the orientation of the hand. As mentioned before in the previous section, the data after transformation is in the reference frame, i.e. to reproduce the hand movement, a point reference has been stationed on the demonstration desk (please refer figure 3.3). The pose of the gripper is equivalent to the pose of the Tool Center Point (TCP) in reference frame. The hand is constructed with the positions of the index finger tip (I), the thumb tip (T) and the index finger ankle (B), with the index finger ankle as the origin. The middle point between the index finger tip and the thumb tip corresponds to the TCP of the robot gripper. The z-axis is defined as the line through the origin and the TCP, while the x-axis and y-axis are calculated through cross product. Together, they form a right-handed coordinate system 3.7.

The rotation matrix which rotates a reference coordinate system to a new coordinate system can be obtained with the unit vector of the new coordinate system:

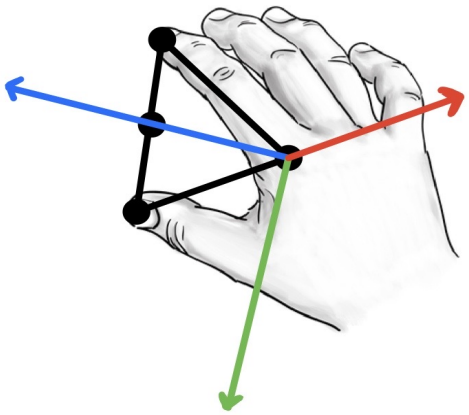


Figure 3.7: Hand coordinate system [Fus]

$$ {}^{b}R_a = \begin{bmatrix} u_{bx1} & u_{bx2} & u_{bx3} \\ u_{by1} & u_{by2} & u_{by3} \\ u_{bz1} & u_{bz2} & u_{bz3} \end{bmatrix} \tag{3.11} $$

The Euler angles convention in CoppeliaSim is defined as:

$$ Q = R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma) \tag{3.12} $$

$$ Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13} $$

For a rigid body, the transformation can be obtained by the rotation about the **absolute** reference frame in the following order: **ZYX**. This can also be understood as the intrinsic rotation around the Tait-Bryan angles **XYZ**. Intrinsic rotation is the rotation about the axes of the rotating coordinate system, which changes its orientation after each elemental rotation, unlike the original coordinate system. In our case, it is the rotation of α about own X, followed by a rotation of β about own Y, followed by a rotation of γ about own Z. We use this quality to test our Euler angles and to calculate the robot gripper pose in the simulator.

3.4.3 Data filtering

One of the major drawbacks of using a vision system to record the demonstration, e.g. in comparison with kinesthetic teaching, lies upon the instability of the recorded data. Not every demonstration after recording is suitable and therefore requires the framework to handle the exception and eliminate undesirable effects. There are three major problems, which often occur during the demonstration recording: detection problem of the markers, deformation of hand and noises.
If one of the markers cannot be detected for a short time span than it will result in an empty line within the "coordinate_transformation.csv"- or "object_trajectory.csv"-file. This phenomenon can be noticed by plotting the raw data: several straight lines occur pointing from the trajectory to the origin of the graph.
The hand can always be detected and therefore does not disappear during the demonstration, however, if the hand has not been detected properly, this will result in an uncontrolled deformation of the hand.
The noises-problem is the major challenge in the demonstration and has significantly impeded the functionality and progress.

The data filtering process uses the command **cscvn** from the **Signal processing toolbox** [KSL94] to calculate the periodic interpolating cubic spline curve out of data points in **ppform**. A univariate piecewise polynomial is defined through a break sequence and a coefficient array. The result can be understood as a set of different forth order polynomial equations modeling and each of them modeling one data point. The major benefit with the calculation of ppforms is its convenience and reliability in the calculation of velocity and acceleration and are to resample the data to meet the demand.

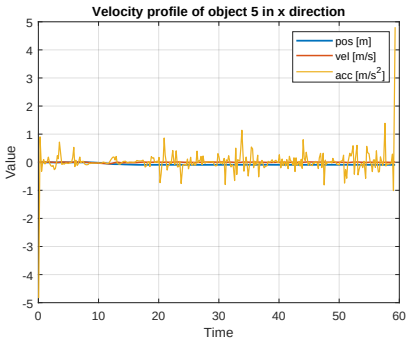
$$p_j(x) = \sum_{i=1}^{k} (x-\xi)^{k-i} a_{ji} \ , 1 \leq j \leq l \tag{3.14}$$

$$\sum_{j=1}^{k} a(j)x^{k-j} = a(1)x^{k-1} + a(2)x^{k-2} + \vdots + a(k)x^0 \tag{3.15}$$

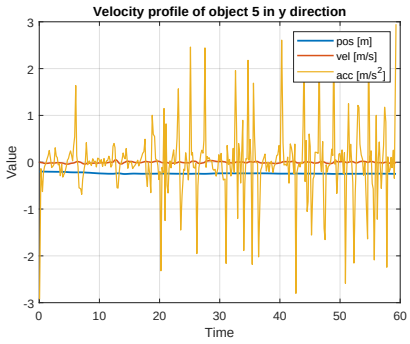
with ξ_1 as breaks and a_{ji} as the local polynomial coefficients.

In this thesis, **Butterworth** filter in combination with **filtfilt** has been repeatedly used to filter out noises for 2 dimensional data. **Butterworth** filter is an Infinite Impulse Response Filter (IIR) which has the advantage that it works recursively and therefore creates the same slope of a filter with a clearly lower order [ABRW14]. The MATLAB command **butter(n, Wn)** returns the coefficients of an nth-order lowpass digital filter with normalized cutoff frequency Wn. In this paper, the order and cutoff frequency are adapted to the concrete scenario. The major benefit of the **filtfilt** filtering is it eliminates phase distortions.

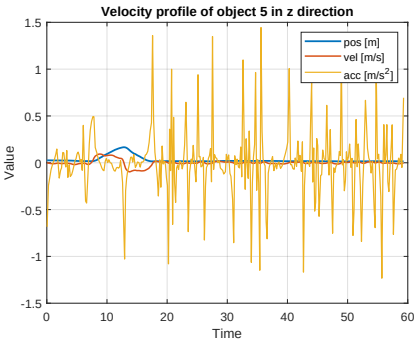
The calculation of the velocity and acceleration is required for the DMP:



(a) X: pos, vel, acc



(b) Y: pos, vel, acc



(c) Z: pos, vel, acc

Figure 3.8: Position, Velocity, Acceleration in X-, Y-, Z-direction

3.4.4 Summary data transformation and filtering:

Sample demonstration: The sample demonstration is to pick up the object 5 and place it on the table, afterward to pick up the object 3 and stacks it on the object 5.
A short summary of the data processing process is depicted in plots 3.9: at the beginning we can see the data floating within the graph and we are not able to tell what kind of demonstration we are analyzing. After the transformation process, the trajectory stands vertical above the XY-plane and we can observe the TCP approaches the objects and picks and places of the objects. At the end returning to the initial position. After the filtering, the TCP trajectory and the object trajectories are more smooth and has fewer variations. The processed data can now be passed to the segmentation process.

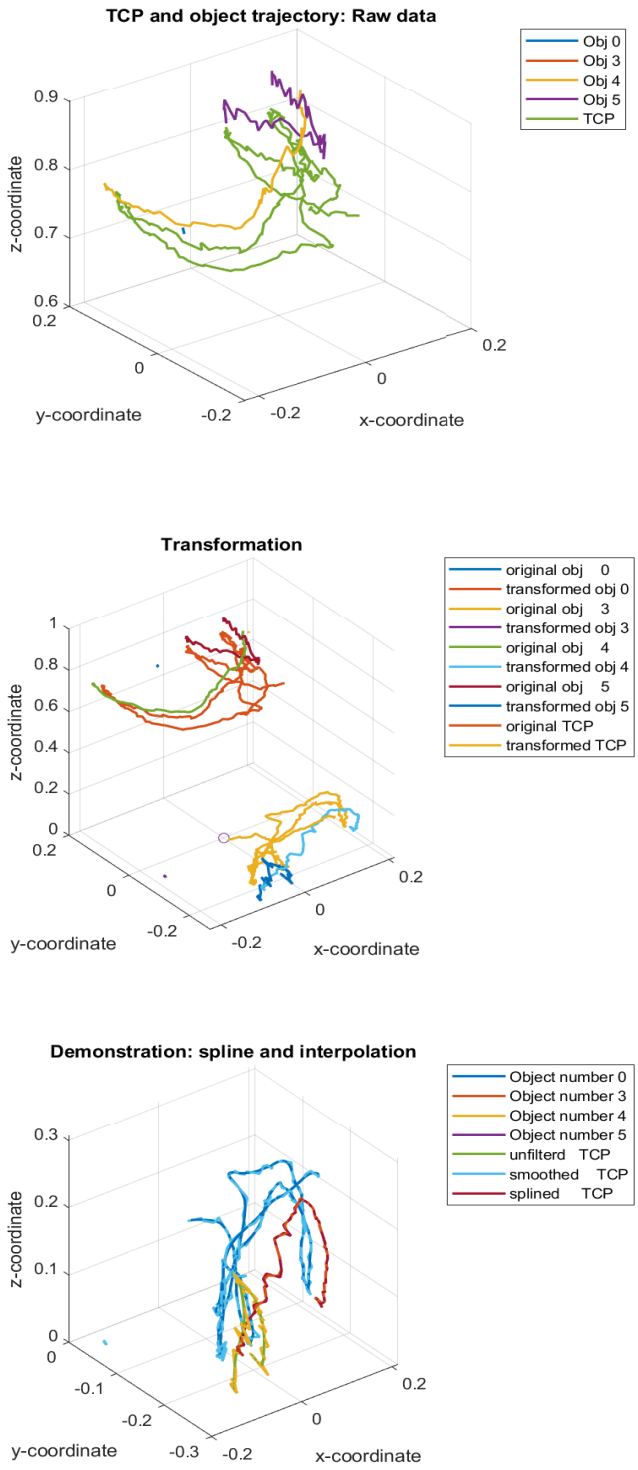


Figure 3.9: 1) Original raw data 2) Transformation from camera to reference frame 3)Transformed and filtered data

3.5 Data pre-processing for HMM

One of the major challenges in the classification is the temporal and spatial variations between the trained templates and the newly generated data. Therefore, all the data, including data to train HMM, has to be modified before the classification with HMM.

3.5.1 Dynamic Time Warping

Dynamic Time Warping (DTW) can eliminate the temporal distortions and creates non-linear alignments between time series that have similar characteristics but locally out of phase [RK04]. The idea is to construct a matrix out of the two sequences and try to find a path within this matrix, which minimizes the cumulative distance between them. In the following graph 3.28 there is the finger tip distance before and after the modification with DTW depicted: the red line is calculated by the means of the corresponding HMM model (from skill **Pick up**), the yellow line represents the demonstration data and the green line represents the data after modification. We can notice that the modified data is better adapted to our HMM model.

DTW [RK04] aligns the following two time series Q and C with different lengths:

$$Q = q_1, q_2, ..., q_i, ..., q_n \tag{3.16}$$

$$C = c1, c2, ..., c_j, ..., c_m \tag{3.17}$$

$$\gamma(i,j) = d(q_i, c_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \tag{3.18}$$

where $\gamma(i,j)$ is the cumulative distance and $d(i,j)$ is the current distance.

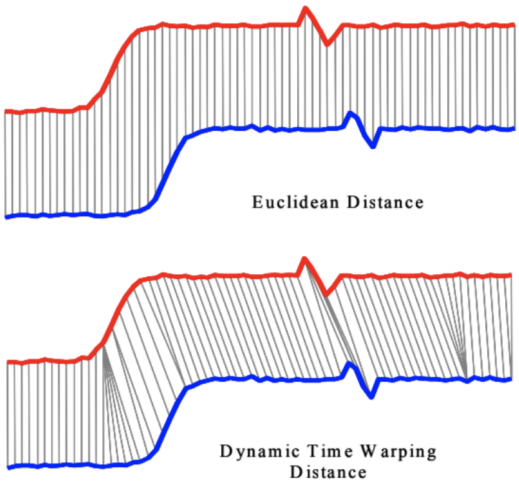


Figure 3.10: Dynamic Time Warping

3.5.2 Rescaling

The idea is to reduce the spatial variations by rescaling every point between 0 and 1. For example, different people have different hand size and stretches the fingers differently during demonstration (e.g. by grasping), so that the absolute value of the distance between the fingertips might vary significantly, even though they are in fact executing the same movement. After rescaling, the maximal stretching during the grasping will be uniformly 1:

$$\hat{x} = \left[\frac{x - x_{\min}}{x_{\max} - x_{\min}} \right] \tag{3.19}$$

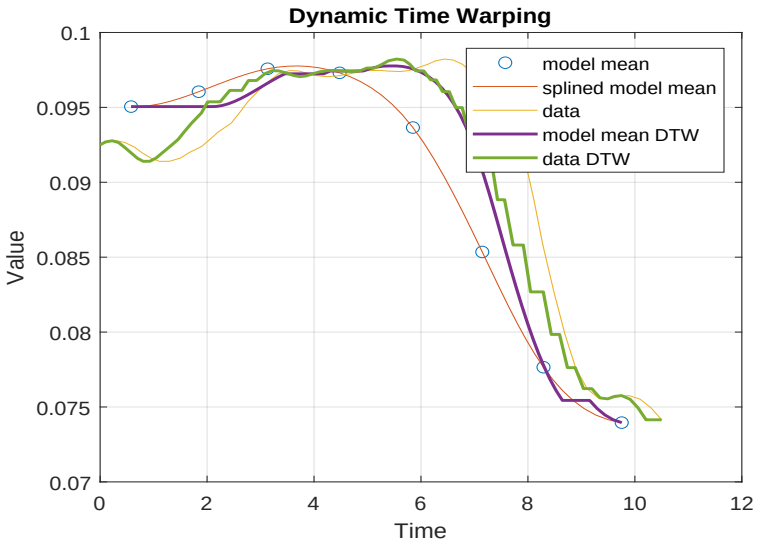


Figure 3.11: DTW data filtering

3.6 Features

To classify segments, 8 different features are defined and are used to characterize a skill, 6 of them are distance based and measures the position and 2 of them are velocity based and monitors the velocity. Like the implementation above, the filter is constructed with **Butterworth** and **filtfilt**, however, the cutoff frequency is higher, because it is desired to keep the characteristics of the features, unlike the filtering of euler angles, where a smooth transition is desired.

3.6.1 Distance Based Metrics

Distance between thumb tips and index fingertips: This diagram provides the information about the distance between the index fingertip and thumb finger-

tip during the demonstration. The fingertips distance provides information about whether an object has been held in hand. One basic assumption is that the finger can have two basic states: **closed** and **opened**. Because of the different hand sizes from demonstrator or different object sizes, the fingertip distance can vary massively. Therefore, no fixed threshold has been defined, but rather the program calculates the mean of the fingertip distance as threshold and divides the data points into **high** and **low** values (yellow line). The **closed** state is associated with **low** values. The positions, where the gripper changes from **high** to **low** or vice versa, indicates a closing or opening of the fingertips and might serve as possible segmentation points.

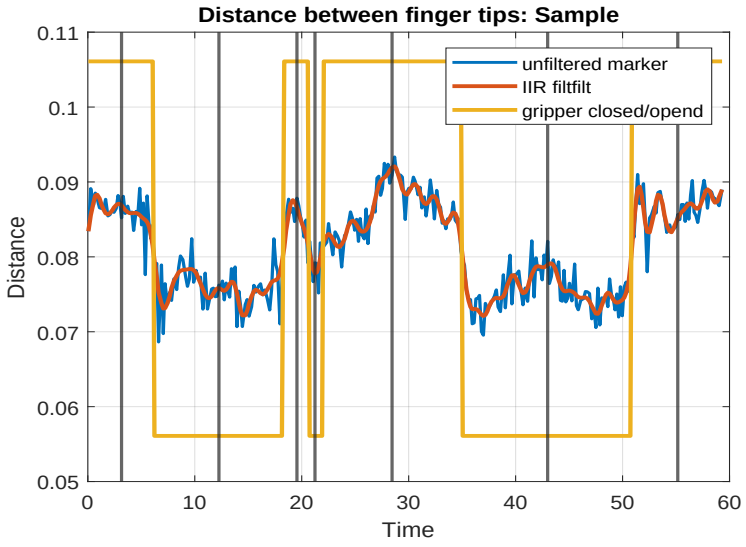


Figure 3.12: Distance between fingertips

Target object profile: This diagram provides the target object during the demonstration by measuring the distance between the reference marker and all other objects. The idea is based on the assumption that all the objects have two states: **static** and **dynamic**. Because the demonstrator performs the demonstration with the right hand, there can only be one object that can be handled at one time. The velocity, which is represented by the derivative of the trajectory, is calculated and also plotted in the diagram (the straight lines around zero). Only when the velocity significantly differs ($\dot{s} > 10^{-3}$) from zero for a time period (at least 15 out of 20 data points due to noise) we consider it as moving and label this section of the time vector t with the moving object as the target object. Other programs will use this information and only consider the characteristics of the target object, e.g. in the template matching.

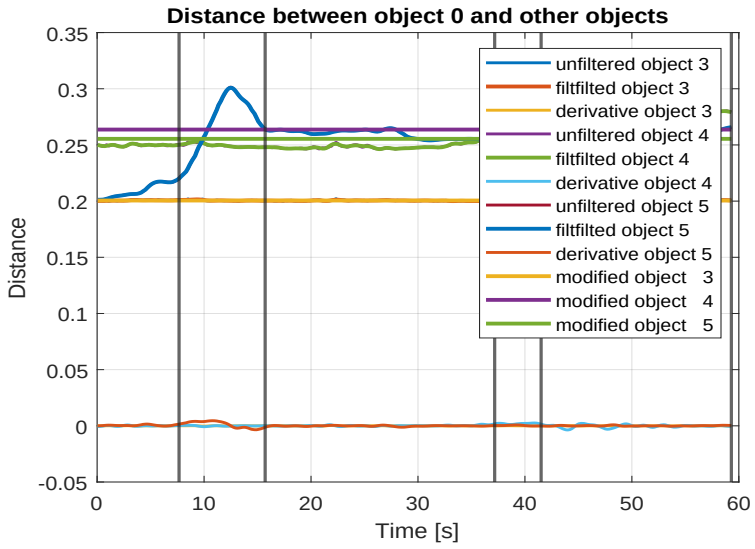


Figure 3.13: Target object profile

Proximity area profile: The diagram provides the distance between the TCP and all the objects during the demonstration. We can observe that the distance between the TCP and the is comparably small while it has been held by the demonstrator. This feature provides the user with the information about how close is the object near the hand and if the hand is approaching or leaving an object. However, this is one of the most unstable features and is therefore only suitable to a limited extent for the classification. We can observe in the following graph that even though the demonstrator only holds the object in the hand while moving, the value can still have a variance of several centimeters. Depending on the velocity of the hand movement, the variances can become even bigger.

TCP in object frame: The diagram provides the position of the TCP in the target object frame. This feature is designed as a supplement to the proximity area profile feature because it also provides the information about the approaching or departing direction.
Because of the limitations of the hand pose estimation algorithm, the object can only be grasped from above (e.g. **Pick up**) or approached from laterally (e.g. **Push**), it is sufficient to only evaluate the position in the X-direction to determine from which direction is the hand approaching. This is only valid if the object is has been placed with the y-Axis pointing above. An additional function should be implemented to determine the orientation of the object and depending on the orientation change the observing axis.

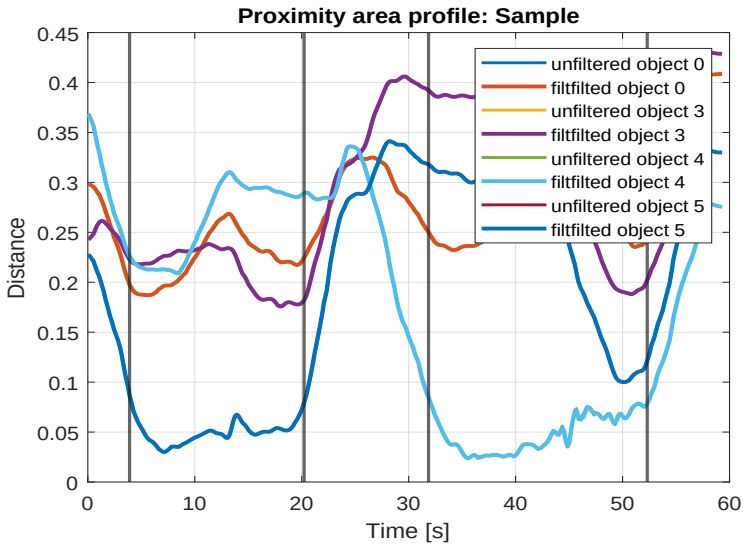


Figure 3.14: Proximity area profile

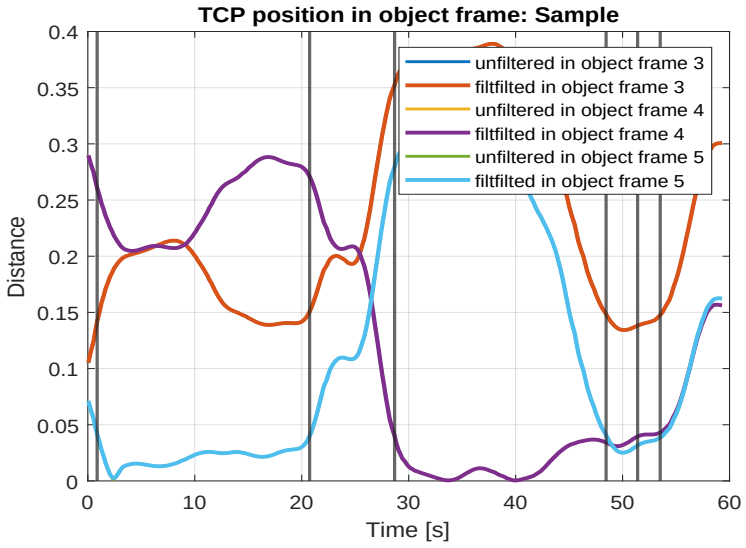


Figure 3.15: TCP position in object frame

Distance between objects and table: This diagram provides information about the distance between the object and the table during the demonstration. The object height is mostly constant and changes only if it has been handled by the demonstrator. This feature is primarily designed to separate the **Place**-skill and the **Stack**-skill from each other. In this demonstration, the object 4 has been stacked over the object 5 and therefore has a constant value around 0.07 m towards the end of the

demonstration. However, because the demonstrated objects are cubes, the height difference is not very clear-cut, it should be a more distinctive feature if the objects have other shapes (e.g. cuboid).

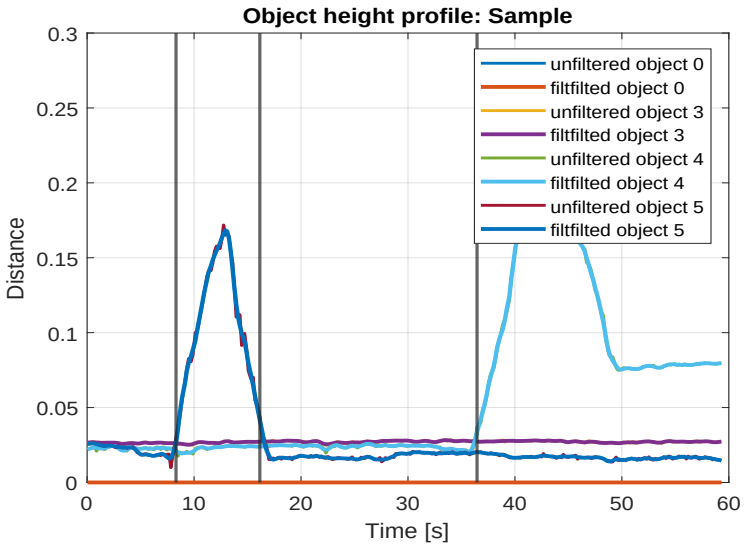


Figure 3.16: Object height profile

TCP rise and fall profile: The diagram provides the TCP height profile and TCP height changing profile during the demonstration and segments the trajectory if the hand changes the movement direction from rising to falling or vice versa. This feature is crucial in the separation of the skills **Move to** and **Move with** from other skills (see skill definition). The derivative of the movement has been calculated to determine the moving direction. The threshold is by zero.

3.6.2 Velocity Based Metrics

Velocity profile hand: The diagram provides the velocity profile of the hand during the demonstration by measuring the changes in the TCP position. This feature is originally designed to be the main segmentation, because of the assumption that the hand velocity will automatically slow down if the demonstrator tries to execute a delicate action or by the transition between two skills. In figure 3.18 we can observe the wavy course of the hand velocity.

Velocity profile objects: This diagram provides the velocity information of the all the objects during the demonstration by calculating the derivative of the object positions. In the previous data filtering process because the SMOOTHN library

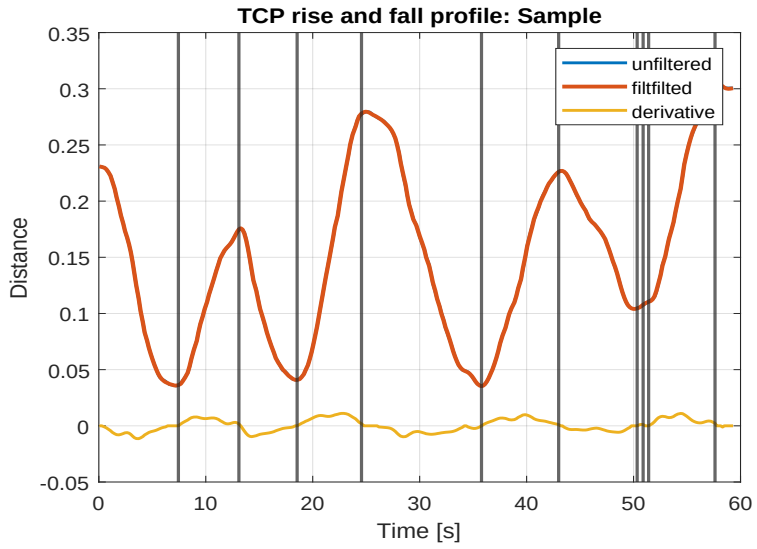


Figure 3.17: TCP rise and fall profile

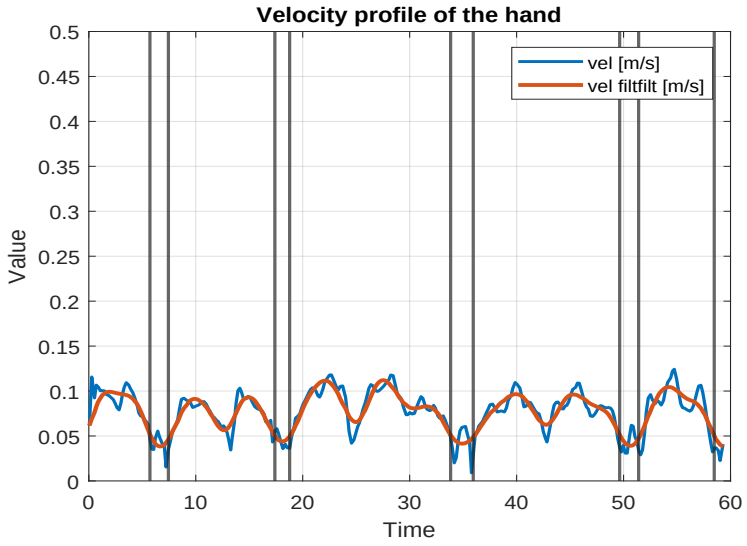
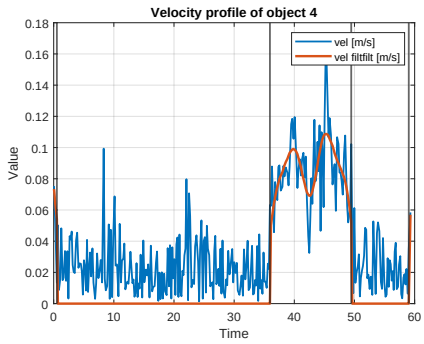


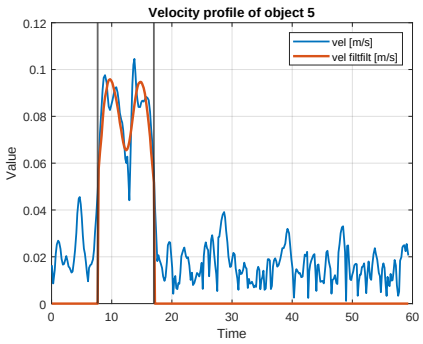
Figure 3.18: Velocity profile hand

cannot calculate the objects' position trajectory which barely moves, a threshold for displacement has been defined under which the velocity is considered to be zero. This will result that the velocity of the target object during a skill execution is often characteristically zero, because the object might not be touched (e.g. pick_up). However, in a demonstration which comprises of multiple skills, the object will often be moved. The result is a poor recognition rate in the later template matching

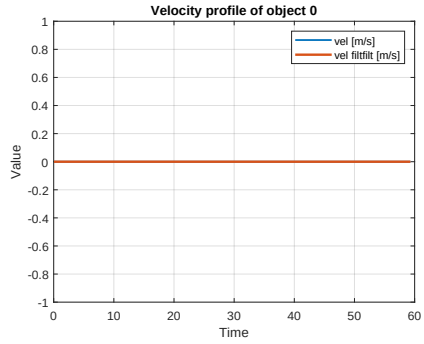
process. Therefore, in this section, the filtered velocity of objects under a certain threshold, which implies an inactivity, will be considered zero.



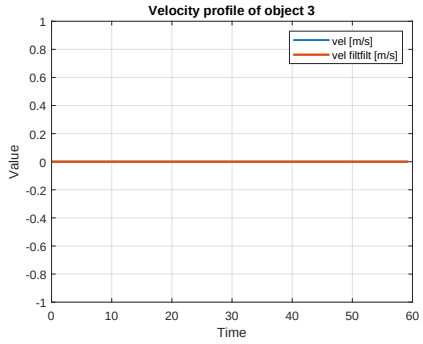
(a) Velocity object 4



(b) Velocity object 5



(c) Velocity object 0



(d) Velocity object 3

Figure 3.19: Object velocity profile

3.6.3 Features selection and weighting

The most important question after determining the features is how to select or weight the extracted features to improve the efficacy of segmentation. There are basically two approaches: only use several features or use all the features for each skill. Further, how to weight the selected features: do all the features have the same importance or are some features more distinctive than others and therefore should have a bigger influence on the classification? The reasoning behind this is intuitive: some features appear more important to a skill than others, some features appear nearly marginal to some skills. For example, the fingertip distance is crucial in the classification of the place-skill but does not characterize the move-skill. On the other hand, how do we compare the likelihood of the skills if they are examined under different conditions?

Skill segmentation							
	Pick_up	**Place**	**Move**	**Move (with)**	**Stack**	**Push**	**Locate**
Hand velocity profile	√	√	√	√	√	√	√
Object velocity profile		√	√	√			√
Target object profile	√	√	√	√	√	√	
Fingertips distance profile	√	√		√	√		√
Proximity area profile	√			√	√	√	√
TCP rise/fall	√	√	√	√	√	√	√
TCP in object frame		√		√	√	√	√
Object height profile	√	√	√		√	√	√

Table 3.1: Features consideration during classification

However, in order to use all the features, the features have to be defined globally and invariantly. In this thesis there is no systematic examination of this two ideas, but rather a small-scale comparison between these two strategies:
In the table 3.1 we have examined each skill individually with the template data (see next section) and have reached the conclusion that what features are important for each skill: The following three feature selection and weighting methods haven been compared:

- Equally weighted, all features
- Differently weighted, selection of features
- Differently weighted, all features

The best segmentation result was achieved by the third method. For example as stated earlier, the **Move to** and the **Move with** skill have the property that the TCP is actually rising while all the other skills have a constant TCP height profile or the height of the TCP is actually decreasing. This is an example for a distinctive skill in the segmentation and has to be examined first. Other skills, for example **Locate** also have distinctive features. After examining these features, some skills will be excluded from the examination and the for the rest of the skills the template matching will be performed for all features equally weighted.
An advantage is the comparison between different skills is possible and plausible because all the skills are examined under the same condition.

3.7 Skill definition

Idea: task is comprised of skills, each skill has a typical time period +- 2s from template. A skill can be understood as a simple atomatic motion and can be used to form more complex behaviours [PNA+16]. In this thesis, skills are used to describe the observed demonstration. However, unlike the approach introduced in [PNA+16] the motions are not further divided into **primitives** or **tasks**.

Template: Each skill has to be demonstrated multiple times in order to generate the model. In this framework, each demonstration lasts between 7 to 12 seconds with different velocity, height and approaching direction. One skill comprises of exactly 9 demonstrations.

Challenge: varying recording values During the implementation, the variation of the recorded data has massively impeded the progress and reliability of this program. For example, by recording the same object simply lying on the table and calculating the distance between object marker with the table, the height has an oscillatory behaviour and the value can vary between 1.5 cm to 3.8 cm, depending on the ambient light. Another problem occurred during monitoring the distance between TCP and object while moving the object holding in hand. Before the actual implementation it was assumed that the distance should be constant and the value should be near 0, however, because of the different velocity in the data processing, the actual value varies significantly depending on the velocity of the movement. Generally are the variances and distances much higher if the hand moves faster. On the other hand, because the data has been filtered in the previous step, the variances and instabiliy are reduced.
In total there are 7 skills defined, however, the user can use the same procedure to add more skills to the model:

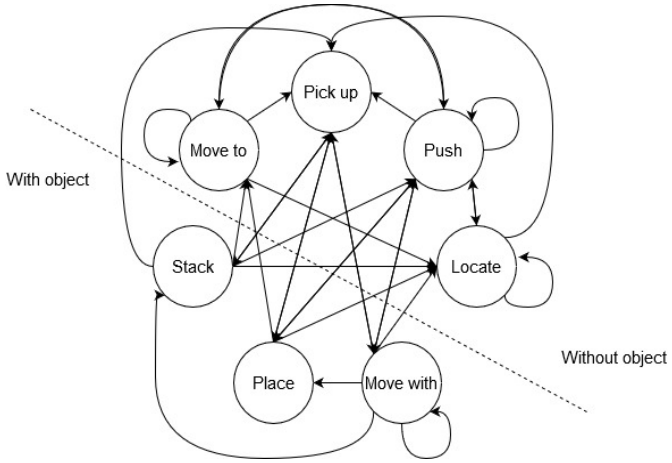
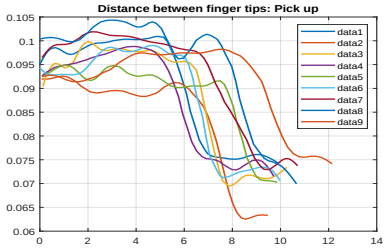


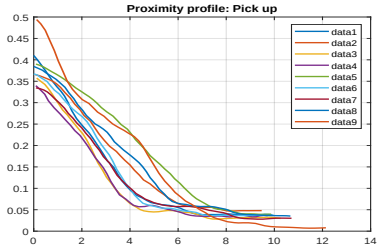
Figure 3.20: Semantic evaluation of skill sequence

Pick up: This skill is defined as approaching an object with stretched fingers from any position and pauses in the proximity of the target object. The demonstrator holds the grasping pose and moves the thumb and index finger slowing towards each other until the object has been grasped. This skill has the following characteristics:

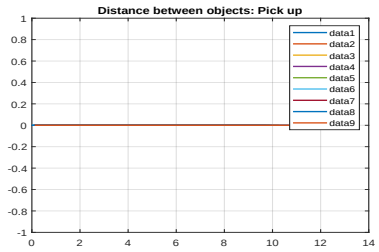
- Distance between fingertips: reducing at the end of the skill
- Proximity area profile: reducing all the time, steady towards the end
- Object movement profile: constant during the whole execution
- Object height profile: constant during the whole execution
- TCP rise and fall profile: negative all the time, steady towards the end
- TCP position in object frame: reducing all the time, steady towards the end



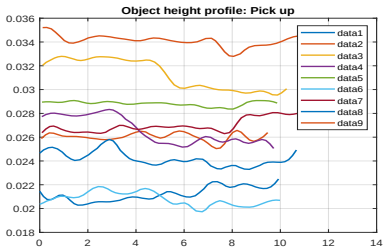
(a) Distance between fingertips



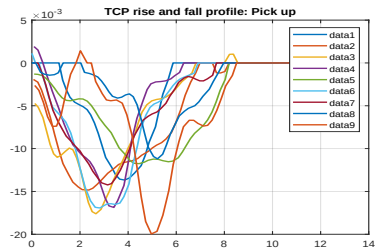
(b) Distance between TCP and target object



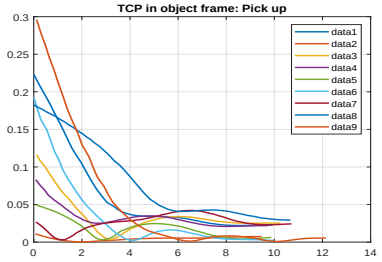
(c) Object movement profile



(d) Object height profile



(e) TCP rise and fall profile

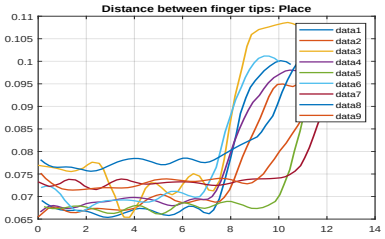


(f) TCP position in object frame

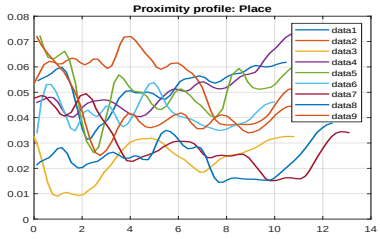
Figure 3.21: Pick up features

Place: This skill is defined as moving with closed fingers, while holding an object in the hand, and approaches the target position on the desk surface. After the demonstrator arrives the desired position and places the object on the desk surface, the demonstrator moves the thumb and index finger from each other while the hand remaining the same pose. This skill has the following characteristics:

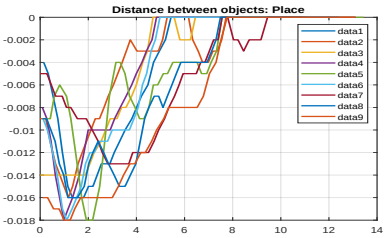
- Distance between fingertips: increasing towards the end of the skill
- Proximity area profile: nearly constant during the whole execution
- Object movement profile: reducing all the time, steady towards the end
- Object height profile: negative all the time, steady towards the end
- TCP rise and fall profile: negative all the time, steady towards the end
- TCP position in object frame: nearly steady all the time, within marker range



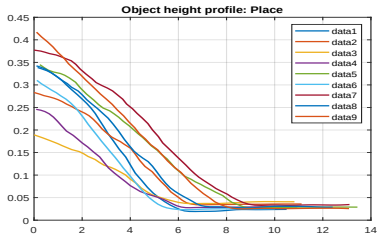
(a) Distance between fingertips



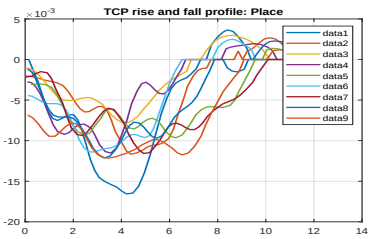
(b) Distance between TCP and target object



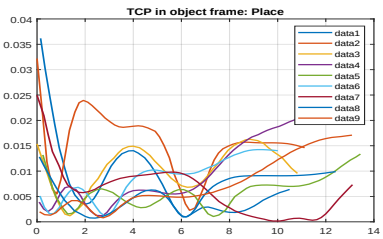
(c) Object movement profile



(d) Object height profile



(e) TCP rise and fall profile

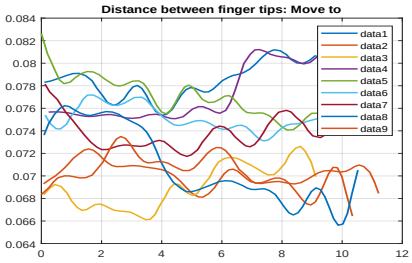


(f) TCP position in object frame

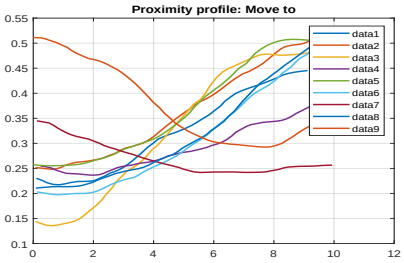
Figure 3.22: Place features

Move to: This skill is defined as the movement of the hand from a lower position to a higher position, without any objects held by the demonstrator. This skill has the following characteristics:

- Distance between fingertips: no obvious characteristic, mostly steady
- Proximity area profile: no obvious characteristic, mostly increasing
- Object movement profile: constant during the whole execution
- Object height profile: constant during the whole execution
- TCP rise and fall profile: positive all the time
- TCP position in object frame: no obvious characteristic, mostly increasing



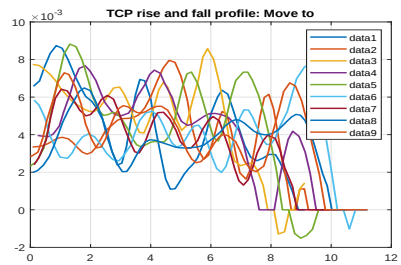
(a) Distance between fingertips



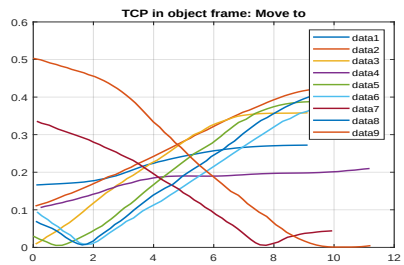
(b) Distance between TCP and target object

(c) Object movement profile

(d) Object height profile



(e) TCP rise and fall profile

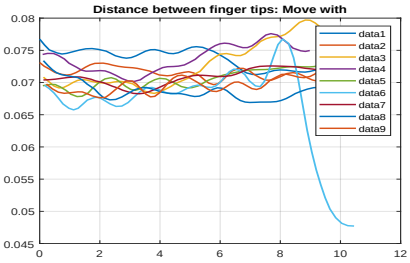


(f) TCP position in object frame

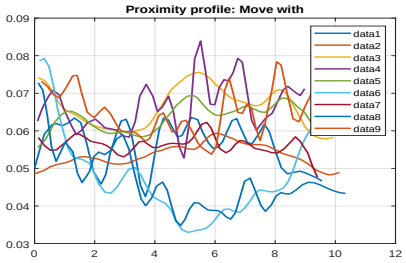
Figure 3.23: Move to features

Move with: This skill is simply defined as the movement of the hand from a lower position to a higher position, with an object held by the demonstrator. This skill has the following characteristics:

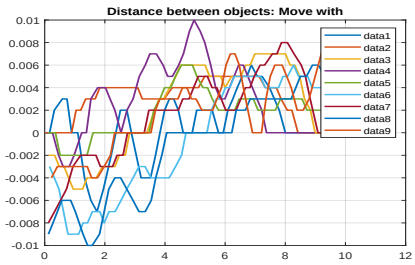
- Distance between fingertips: steady during the whole execution
- Proximity area profile: nearly steady all the time
- Object movement profile: no obvious characteristic, mostly increasing
- Object height profile: increasing during the whole execution
- TCP rise and fall profile: positive all the time
- TCP position in object frame: no obvious characteristic, within marker range



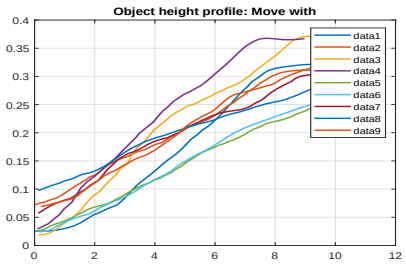
(a) Distance between fingertips



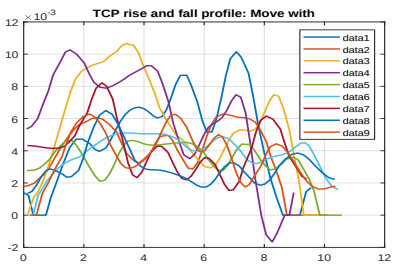
(b) Distance between TCP and target object



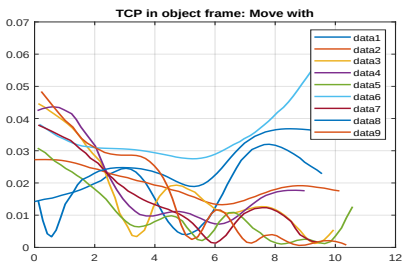
(c) Object movement profile



(d) Object height profile



(e) TCP rise and fall profile

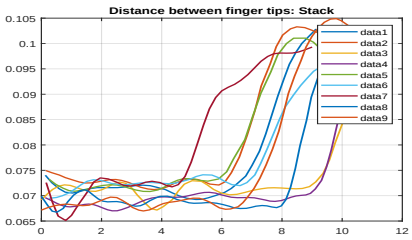


(f) TCP position in object frame

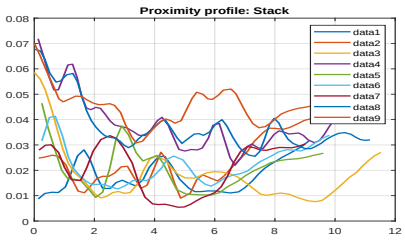
Figure 3.24: Move with features

Stack: This skill is defined as as moving with closed fingers, while holding an object in the hand, and approaches the target object. The demonstrator adjusts the hand pose to align the holding object with target object. After the object has been placed, demonstrator moves the thumb and index finger from each other while the hand remaining the same pose. This skill has the following characteristics:

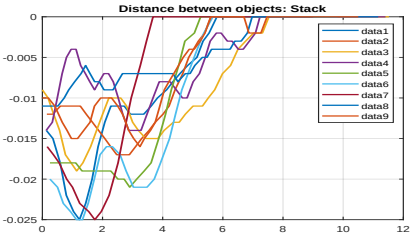
- Distance between fingertips: steady all the time, increasing towards the end
- Proximity area profile: nearly steady during the whole execution
- Object movement profile: negative all the time, constant towards the end
- Object height profile: falling all the time, constant towards the end
- TCP rise and fall profile: negative all the time, steady towards the end
- TCP position in object frame: nearly constant all the time, within marker range



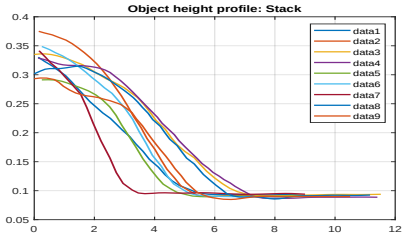
(a) Distance between fingertips



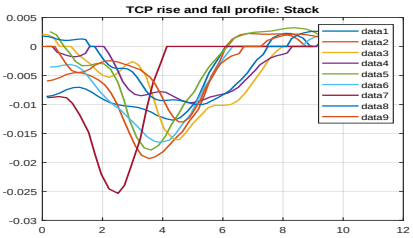
(b) Distance between TCP and target object



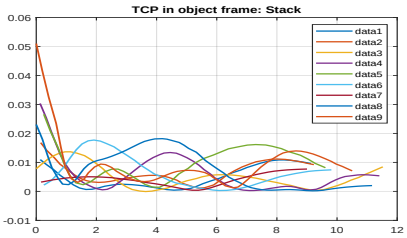
(c) Object movement profile



(d) Object height profile



(e) TCP rise and fall profile

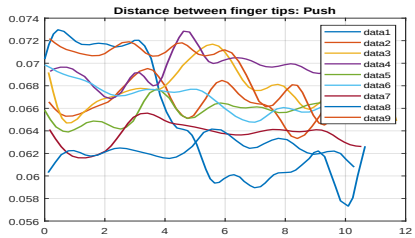


(f) TCP position in object frame

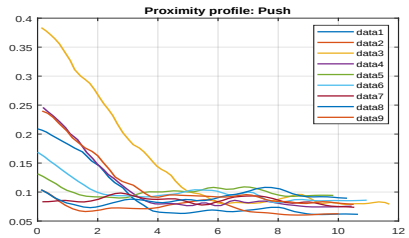
Figure 3.25: Stack features

Push: This skill is defined as approaching an object with closed fingers from any position and pauses in the proximity of the target object. The demonstrator pushes the target object from any position while keeping the same hand pose, this will result the object to glide along the desk surface. This skill has the following characteristics:

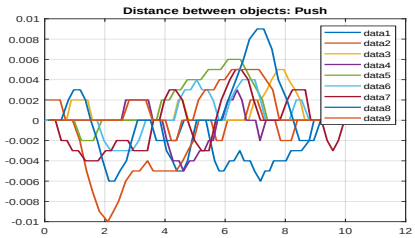
- Distance between fingertips: no obvious characteristic, mostly steady
- Proximity area profile: falling all the time, steady towards the end
- Object movement profile: no obvious characteristic, mostly steady
- Object height profile: mostly steady during the whole demonstration
- TCP rise and fall profile: steady all the time, around zero
- TCP position in object frame: falling all the time, steady towards the end, out of marker range



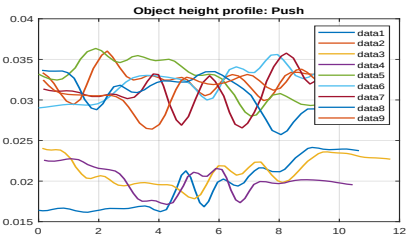
(a) Distance between fingertips



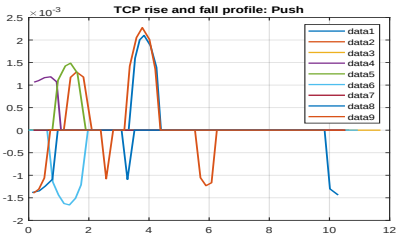
(b) Distance between TCP and target object



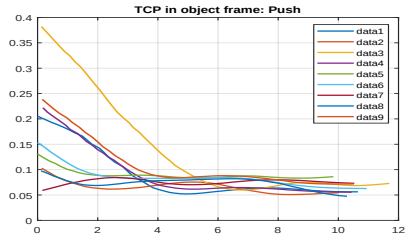
(c) Object movement profile



(d) Object height profile



(e) TCP rise and fall profile

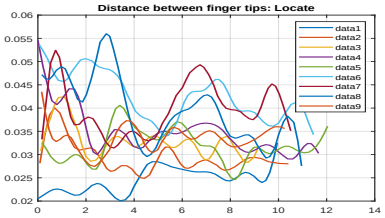


(f) TCP position in object frame

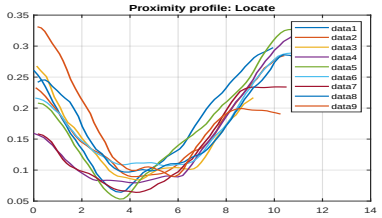
Figure 3.26: Push features

Locate: This skill is defined as gliding with closed fingers from any position along the desk surface, with approximately the same height and pose, towards the target object and avoids the object from above. After passing through the object the demonstrator glides along the surface for a short time period. This skill is intended to determine dimensions within the demonstration. For example, the camera only registers the position and orientation of the marker, but not the size of the object, on which the marker is attached. This information could be used in the obstacle avoidance.

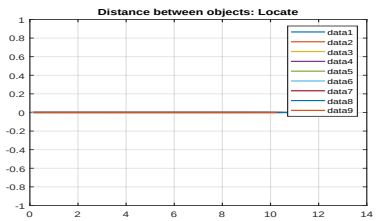
- Distance between fingertips: no obvious characteristic, mostly steady
- Proximity area profile: falling all at the beginning, increasing towards the end
- Object movement profile: constant during the whole execution
- Object height profile: mostly constant during the whole demonstration
- TCP rise and fall profile: steady all the time, around zero
- TCP position in object frame: falling all at the beginning, increasing towards the end



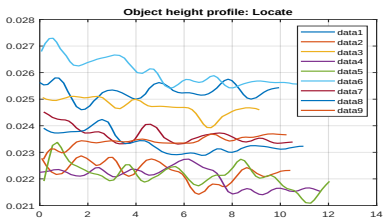
(a) Distance between fingertips



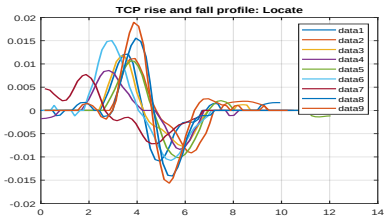
(b) Distance between TCP and target object



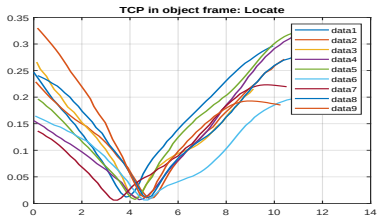
(c) Object movement profile



(d) Object height profile



(e) TCP rise and fall profile



(f) TCP position in object frame

Figure 3.27: Locate features

3.7.1 Skill semantic order

The idea is that a skill sequence should be semantic meaningful. For example, the algorithm has discovered a place skill, which implies that there is no object currently holding in hand. If the next recognized skill is a stack, this suggests a failure at least in the recognition. The relationships between skills are summarized in the following graph 3.20. An arrow indicates a logical meaningful next skill.
The knowledge about the skill semantic order can be used to reduce the computing effort or to evaluate the success of a segmentation. In this paper, the semantic order of skills has been implemented during the segmentation process to sort out certain skills before the classification and segmentation.

3.7.2 Template matching: Hidden Markov Model

Hidden Markov Model is a probability statistical classifier, which can return the probability that an observation can be assigned to a particular class. The idea can also be formulated as: find the class that can best reproduce the given observation. This is a great advantage of using HMM as the classifier: there is no fixed threshold defined, which determines whether the observation belongs to a certain class. For example, because the author generated the template for classification, it is natural that the author might get higher scores in the classification, considering the hand size or typical movement velocity. This will distort the result. A HMM-Model can be described with three elements [Rab89]:

1) State transition probability distribution matrix **A**:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i] \ , \ 1 \leq i, j \leq N \tag{3.20}$$

2) Observation symbol probability distribution matrix **B**:

$$b_j(k) = P[v_k \mathrm{att} | q_t = S_j] \ , \ 1 \leq j \leq N, \ 1 \leq k \leq M \tag{3.21}$$

3) Initial state distribution π:

$$\pi_i = P[q_1 = S_i] \ , \ 1 \leq i \leq N \tag{3.22}$$

We can apply the Baum-Welch algorithm to make an estimation of parameters which define a HMM model based on an observed data set. Baum-Welch Algorithm is a method designed to iteratively adjust the parameters A, B and pi to maximize the probability of the occurrence of the obeserved sequence O [Rab89]. The **Baum-Welch Algortihm** requires the forward variables and the backward variables, which are calculated through the **Forward Algorithm** and **Backward Algorithm**:

The **Forward Algorithm** [Rab89] is given by

$$\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = S_i|\lambda) \tag{3.23}$$

1) Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1) \ , 1 \leq i \leq N \tag{3.24}$$

2) Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \ , 1 \leq t \leq T-1 \ , 1 \leq j \leq N \tag{3.25}$$

3) Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) a_T(i) \tag{3.26}$$

where $\alpha_t(i)$ is the forward variable, which entails the probability of the observation sequence $O_1 O_2 ... O_t$ and state S_i at the current time t with the given HMM model λ.

The **Backward Algorithm** [Rab89] is given by

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T|q_t = S_i, \lambda) \tag{3.27}$$

1) Initialization:

$$\beta_T(i) = 1 \ , 1 \leq i \leq N \tag{3.28}$$

2) Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \ , \ t = T-1, T-2, ...1, \ 1 \leq i \leq N \tag{3.29}$$

where $\beta_t(i)$ is the backward variable, which entails the probability of the observation sequence $O_{t+1} O_{t+2} ... O_T$ and state S_i at the current time t with the given HMM model λ.

The observation symbol probability distribution $\mathbf{B}$ is generated by a set of multivariate Gaussians, where each Gaussian mean represents the position of the calculated state and the covariance matrix represents the variance and covariance [LK13]:

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^p \det(\sum)}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^\intercal \sum^{-1} (\mathbf{x}-\mu)) \tag{3.30}$$

The probability of the given sequence O: this HMM model leads with the probability $P(O|\lambda)$ to the given observation O. $P(O|\lambda)$ is crucial to determine to which

class belongs the observation. $P(O|\lambda)$ can be efficiently calculated with the **forward algorithm**.

The **Baum-Welch Algorithm** [Rab89] is given by

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1}\beta t+1(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(j)} \tag{3.31}$$

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j) \tag{3.32}$$

where $\beta_t(i)$ is the backward variable, which entails the probability of the observation sequence $O_{t+1}O_{t+2}...O_T$ and state S_i at the current time t with the given HMM model λ.

Summary: In this thesis, each feature from each skill has an own HMM model, making a total of 56 (7x8) HMM models. Each HMM model can provide the information about what is the possibility that the observation belongs to skill **X** if we only consider this feature.

3.8 Human Motion Segmentation

In the following there is structure of the segmentation process depicted. The segmentation procedure is a two-tiered segmentation process: the first step is the segment point modelling, which over-segments the motion and in the next step template matching will be applied to reduce the over-segmentation and gives the extracted segments semantic meaning.

3.8.1 Segment Point Modelling

The main segmentation forms the basic structure of the human motion segmentation (windowing) [LK13].

- single criteria, e.g. hand velocity profile
- multiple criteria

The problem of considering the segmentation based on single criteria is its lack of comprehensiveness. At the beginning, the author assumed that the velocity profile of the hand is sufficient to segment all the skills from each other. The idea is

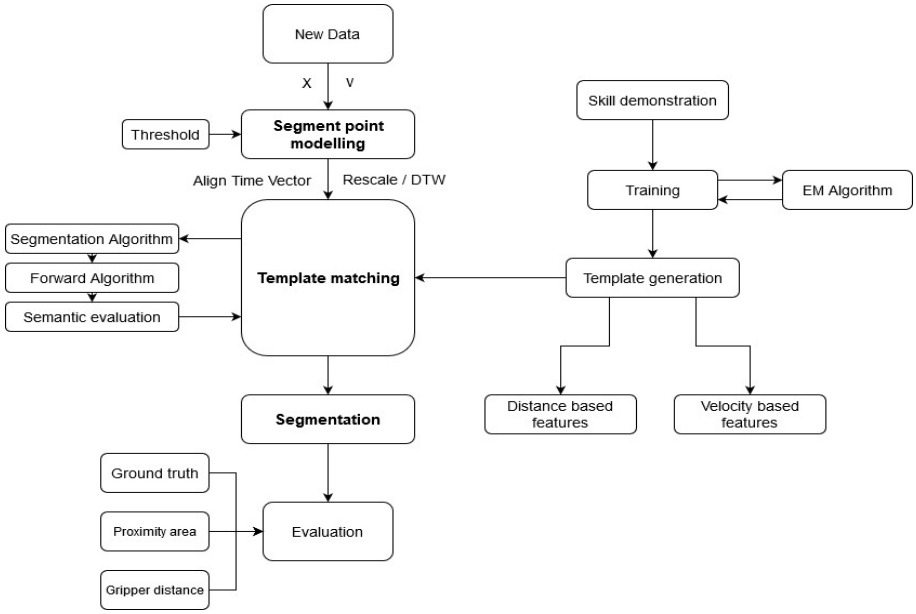


Figure 3.28: Block diagram: segmentation

that it is a natural way of behaviour to reduce the velocity while demonstrating delicate skills like "place" or "push" and possesses a much higher velocity during casual movements. However, a simple demonstration unveils the limitations of this single-criteria segmentation. A simple "pick_and_place"-task has performed and one might observe a small velocity decrease around 17 seconds 3.18, this is caused by the switching from a **move_with** skill to a **place** skill. However, a normal human demonstrator can easily switch swiftly between two skills, which are physically coherent. Even by adjusting the threshold value this way of segmentation is not sufficient to separate all skills robustly.
The main segmentation has to be determined by considering multiple criteria at the same time. This will automatically leads to over-segmentation, but the number of segments can be reduced with template matching.

3.8.2 Segmentation algorithm

After the main segmentation has been found, it is now the task to reduce the number of segments with a strategy. As mentioned before, the demonstration should be fully divided into skills after the segmentation. The first segment comprises the first data segment, the second segment comprises of the first and second data segment and so forth. The last segment comprises all the data segments. The algorithm applies the template matching procedure to all the segments and calculates the score of each skill for all the segments. Afterward each segment is labeled with the skill witch achieves the highest score. At the end, every segments will be compared and the segment with the highest score will be cut off and forms the first skill. This process

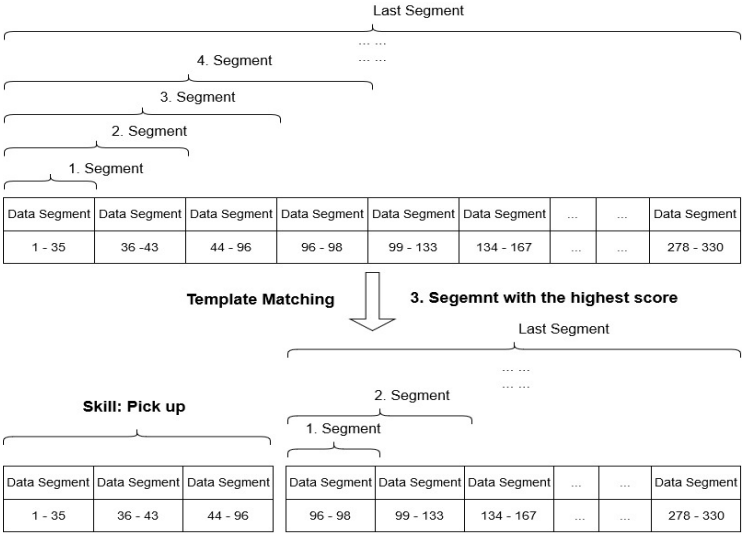
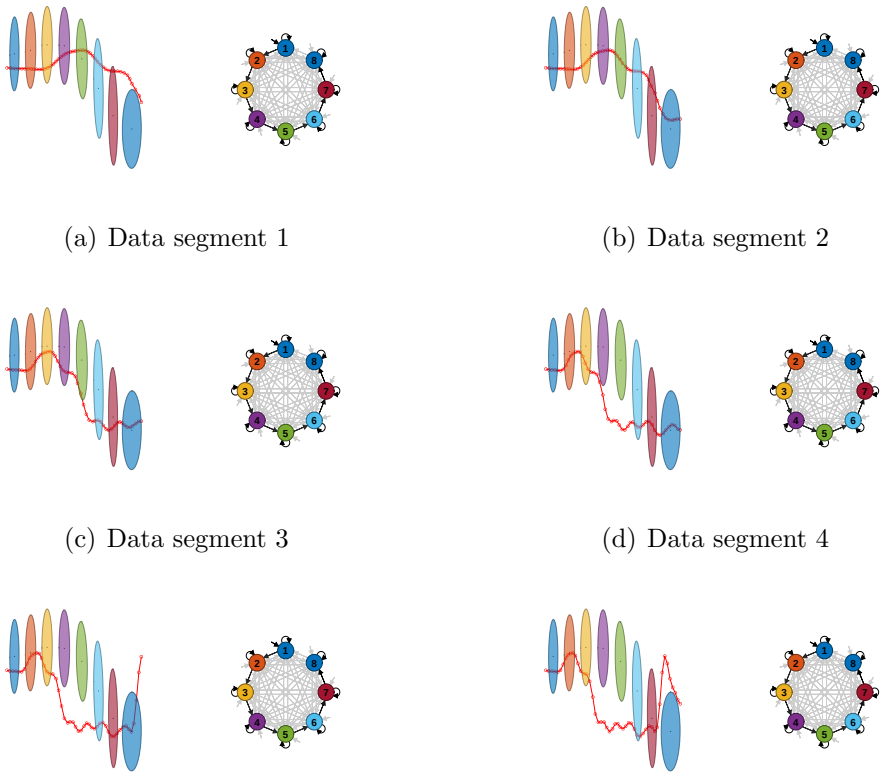


Figure 3.29: Segmentation algorithm

will repeat until the whole segment has been divided into skills.
Some additional steps have been implemented to facilitate this algorithm. First, the data should be modified before passing to the template matching process: as mentioned before, each skill is defined over its own property and the interaction with a target object. For example, during the place skill, we want to examine the data array "distance between the TCP and the object" (d_tcp_o), it will be misleading if we falsely calculate the distance between the TCP and a random object, which is not held by the hand. In the segment point modelling process we have labeled all the data points with an object number. We use this information to cascade our new d_tcp_o out of old d_tcp_o segments, i.e., the first 30 points we extract from the array distance between TCP and object 1, the last 50 points we extract from the array distance between TCP and object 3, etc. Further, in the definition of skills we have mentioned the semantic order of skills, we also take this idea into account when we apply the segmentation algorithm to save computing power.
Further the algorithm modifies the time vector of the demonstration data segment to be aligned with the template (mostly between 10s to 12s). Without this step, we may notice that the calculated score is extremely low, because the algorithm always tries to find values between 0 and approx. 10s, but our data segment might be outside of this time range.
The following graph depicts a section of the template matching process. The distance between fingertips data from the demonstration (red line) is passed into the corresponding HMM model. In this section the program calculates the score for the each data segment with the HMM model from the **Pick up** - skill. In this case, the data segment 2 will have the highest score. This result implies that the first two data segments combined are most likely than all other combinations of segments to be a **Pick up** skill, if we solely consider the course of the distance between finger-

tips. This procedure will be extended to the whole demonstration trajectory and for each skill multiple features are considered (see section 3.6.3). Possible results of the segmentation can be seen in the experiment chapter.



(a) Data segment 1

(b) Data segment 2

(c) Data segment 3

(d) Data segment 4

(e) Data segment 5

(f) Data segment 6

Figure 3.30: HMM Template Matching: Distance between fingertips - Pick up

3.9 Dynamic Movement Primitives

After the program has successfully segmented trajectory and extracted the skills, the robot can use this information to be more adaptive to the environment. In this framework, the user has the possibility to choose one of the objects and and changes the position and the robot will still be able to perform the same task.
One of the major challenge in the implementation is to calculate the target gripper position and orientation after the object has been displaced. The idea is to calculate the transformation matrix from the hand frame to the object frame and use this transformation matrix to calculate the new hand poses (3.7). One of the calculated new hand pose is the new grasping pose. The new grasping pose of the hand is used

as the target pose of DMP.

Dynamical movement primitives encodes a movement primitive into a second order, non-linear dynamical system [CSFL19]:

Spring-damper system with a nonlinear forcing term [CL17]:

$$\ddot{\mathbf{x}} = k^p(\mu_T - \mathbf{x}) - k^v\dot{\mathbf{x}} + \mathbf{f}(s) \ , \ \mathbf{f}(s) = \sum_{k=1}^{K} \phi_k(s) s \mathbf{F}_k \tag{3.33}$$

$$\widetilde{\phi}_k(\mathbf{x}_n{}^I) = \exp\left(-\frac{1}{2}(\mathbf{x}_n{}^I - \mu_k{}^I)^\intercal \mathbf{\Sigma}_k{}^{I\,-1}(\mathbf{x}_n{}^I - \mu_k{}^I)\right) \tag{3.34}$$

$$\phi_k(\mathbf{x}_n{}^I) = \frac{\widetilde{\phi}_k(\mathbf{x}_n{}^I)}{\sum_{i=1}^{K} \widetilde{\phi}_i(\mathbf{x}_n{}^I)} \tag{3.35}$$

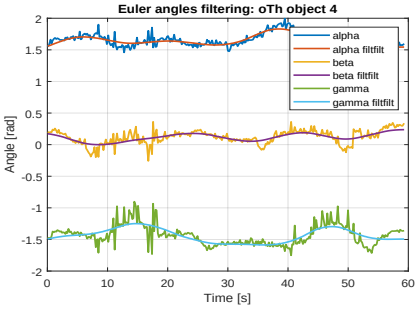
$$\mathbf{W}_k = \mathrm{diag}(\phi_k(\mathbf{x}_1^I), \phi_k(\mathbf{x}_1^I), ..., \phi_k(\mathbf{x}_1^I)) \tag{3.36}$$

$$\mathbf{X}^O = \mathbf{X}^I\mathbf{A} \tag{3.37}$$

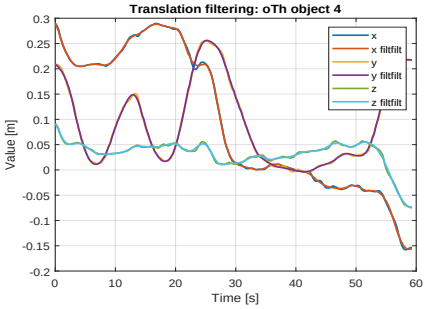
$$\hat{A} = (\mathbf{X}^{I\intercal}\mathbf{W}\mathbf{X}^I)^{-1}\mathbf{X}^{I\intercal}\mathbf{W}\mathbf{X}^O \tag{3.38}$$

where X^I is the input data, X^o is the output data, $\phi_k(\mathbf{x}_n{}^I)$ as the weighting functions, W as the weighting matrix, μ_T represents the end-point, k^p is stiffness, k^v is damping and $\mathbf{f}(s)$ embodies the forcing term with s as phase variable.

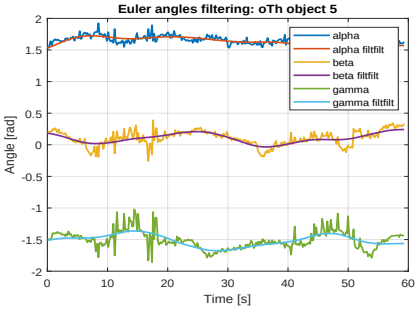
In (3.38) the phase variable s is represented by the expression $\dot{s} = -\alpha s$, which means that the phase variable converges towards 0. The forcing term can also be encoded with Gaussian Mixture Model (GMM), which uses a the EM algorithm to learn the means and covariances of the GMM and therefore does not require the basis function [PL18].
The Euler angles and the translation have to be filtered before passing to the robot. In this framework, the user has the possibility to change the position of an object and the robot will still be able to perform the same demonstration. Before modifying the object positions, the segmentation procedure has to be completed first.
In a simple user interface, the system provides the user with the information about the available objects and the total number of segments. After the user has specified a segment and an object, the user can relocate the target object with a x- and y-offset. In this demonstration, there are two sequentially ordered segments: **Pick up** - segment and the **Move with** - segment. The user chooses to change the position of an object in the **Pick up** - segment. The new pick up position, which can be



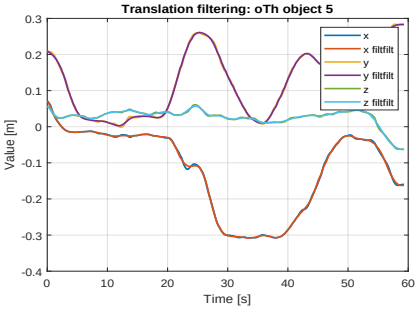
(a) Transformation from hand to object 4: Euler angles filtering



(b) Transformation from hand to object 4: translation filtering



(c) Transformation from hand to object 5: Euler angles filtering



(d) Transformation from hand to object 5: translation filtering

Figure 3.31: Transformation hand frame to object frame filtering

calculated with coordinate transformation from hand frame to object frame, becomes the new target position of the **Pick up** - segment and the new start position of the 'Move with'-segment. The original (grey color) and the calculated object position and euler angles trajectories are shown in the following graphs:

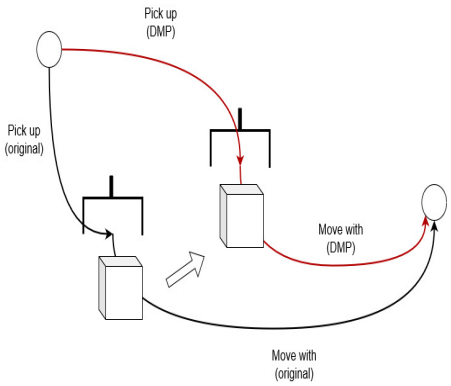
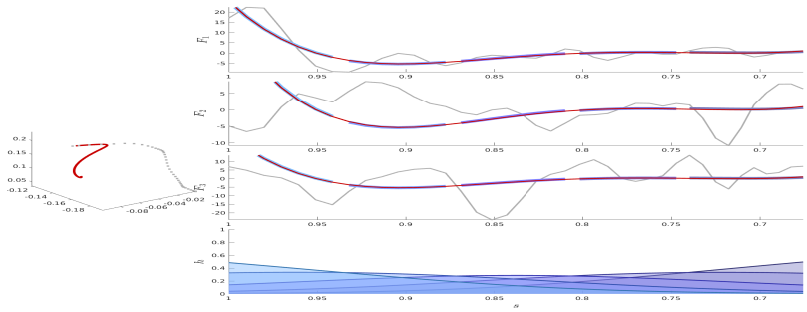
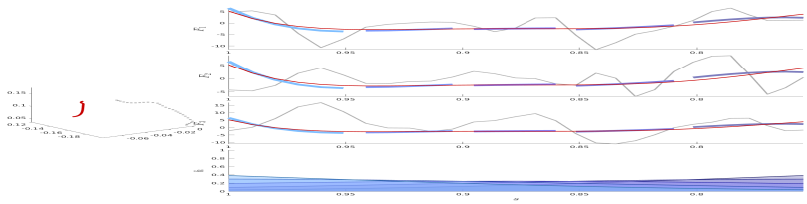


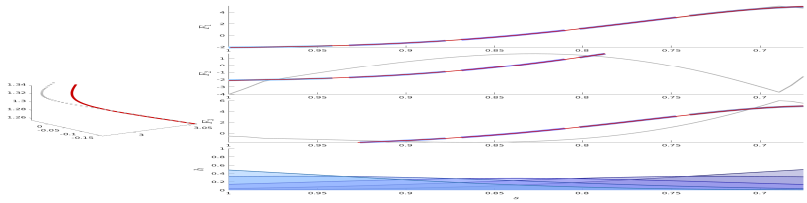
Figure 3.32: Application of DMP to generate new trajectory



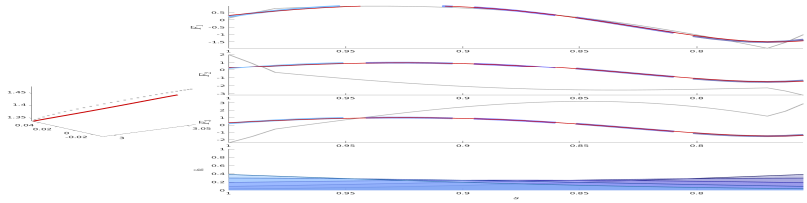
(a) DMP for segment 1: position



(b) DMP for segment 2: position



(c) DMP for segment 1: euler angles



(d) DMP for segment 2: euler angles

Figure 3.33: DMP position and euler angles

Chapter 4

Experiment

4.1 Experiment: Segmentation algorithm

In the previous chapter we have shown how the framework is used and how to understand the result in each step. In the second experiment we will apply the same procedure to test the proposed segmentation algorithm.

Idea: The idea is to test the reliability of the proposed algorithm by designing experiments, which can be segmented into skills according to the definitions. The development of this algorithm is time consuming, because it requires the demonstration

4.2 Experiment design

Four scenarios with each four demonstrations are designed to test the effectiveness and correctness of the proposed segmentation method. The proposed method and two other segmentation schemes are compared with the ground truth.

- finger-distance-based segmentation
- proximity-area-based segmentation
- **proposed approach**
- ground truth

The finger-distance-based segmentation monitors the finger tip distance and segments if the finger tip distance crosses a threshold. The proximity-area-based segmentation segments if the TCP enters or leaves the proximity region of an object. The ground truth method is a segmentation method based on human observation: the human segments there, where he thinks it represents a semantic meaningful segment. A meaningful segmentation is if the result resembles the ground truth segmentation.
The four scenarios are designed to cover all the defined skills and are complex enough

to require an effective segmentation strategy. Each skill is represented by a color and the legend of the color usage can be found in the figure 4.1. **Proximity area segmentation** and **Gripper distance segmentation** charts only have two colors: orange and blue. This is because in these two approaches no semantic meaning could be derived, these two colors only indicate the state of the robot gripper. A segment with the color yellow indicates a near object state or a closed gripper state and a segment with the color blue indicates a far object state or a opened gripper state.

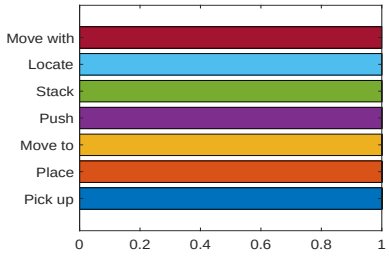


Figure 4.1: Skill legend

4.3 Experiment execution

As mentioned before, one of the major drawback using a vision system, e.g. in comparison with kinethetic teaching, is the instability of the recorded data. Even though the framework can compensate some of the undesired effects, nevertheless, the demonstrator should be extremely careful about the following aspects to have a higher success rate in the generation of data:

Detection of marker and hand: due to light reflections and the environment in which the the hand estimation algorithm has been trained, the demonstrator should always monitor the demonstration with **RIVZ** during the execution. If the demonstrator notices e.g. a marker has disappeared or the deformation of the hand pose, he should repeat the demonstration. **Filtered data has been used to calculate segmentation.**
Three cuboids are used to perform the demonstration. Each one of them has the size of approx. 6 cm x 6cm x 6cm and is attached with an AR track marker (5 cm x 5cm). Each marker has an individual number (3, 4, 5) and a reference marker (0) has been attached on the table. The camera is positioned in such way, so that the reference marker's vertice is parallel to picture frame. The demonstrations from the same experiment have been performed as different as possible, i.e. the duration of the demonstration, the target objects, the velocity of the hand movement etc.

Result evaluation: The similarity between a segmentation approach with the ground truth is an indication of the efficacy of this segmentation approach. However, a segmentation according to the ground truth is not the only way of segmenting human motions, other segmentation might also be valid. For example there are two skills ordered sequentially: **Movet with** and **Place**. The length of the skill **Move with** is not decisive for the success of the segmentation, a relative long **Movet with** skill followed by a relative short **Place** skill is from the reproduction perspective the same as a relative short **Movet with** skill followed by a relative long **Place** skill. Therefore, the results should be further tested in the simulator.

4.4 Evaluation

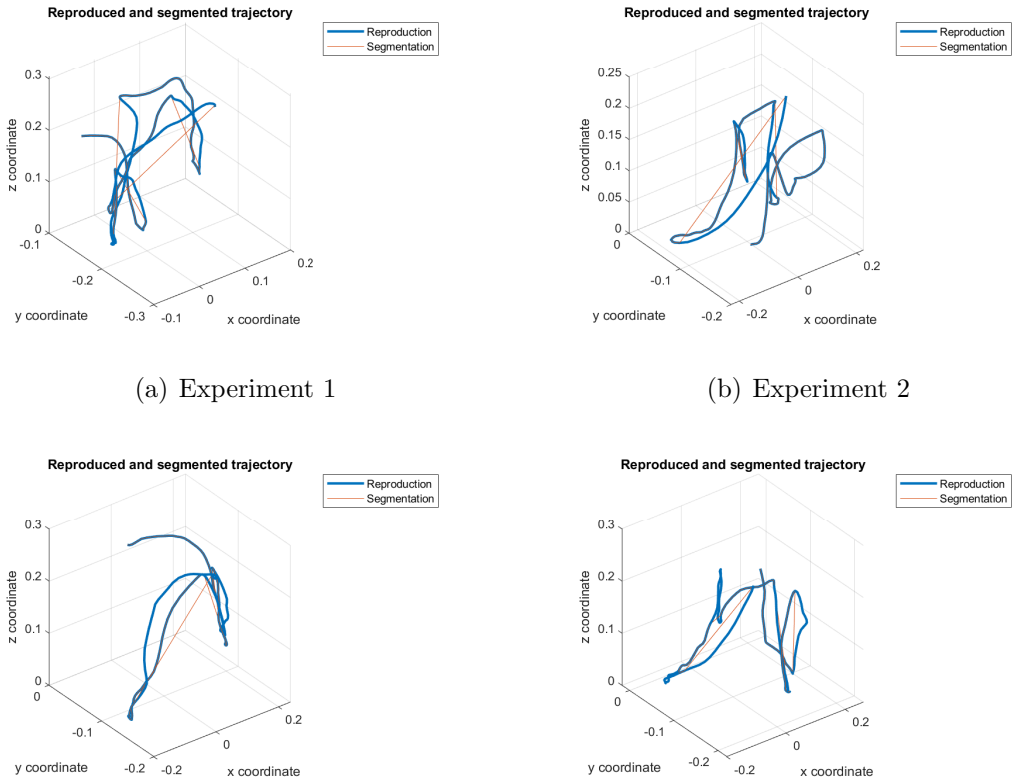
Experiment 1: Pick up, Place, Pick up, Stack The second experiment includes a **Pick up** - skill, a **Place** - skill, a **Pick up** - skill and a **Stack** - skill. This experiment is also designed to test the DMP calculations, because of the nature of the skills, it is very convenient to change the position of an object without affecting other objects.
The major difficulty during the demonstrations was the shadowing, i.e. if the shadow of the hand overlaps the marker, this will sometimes cause the marker to disappear for several seconds. We can see that the **Proximity area segmentation** and **Gripper distance segmentation** are suitable approaches to segment the human motions. Although sporadic gripper opened states within the gripper closed states might cause the robot to stop and manipulates the gripper, the overall result is acceptable considering the simplicity to implement these two approaches.

Experiment 2: Locate, Pick up, Stack, Push The second experiment includes a **Locate** - skill, a **Pick up** - skill, a **Place** - skill and a **Push** - skill.
This demonstration is the most difficult one among the four, not only the segmentation has the lowest success rate among the demonstrations, but was also the most difficult one to perform the demonstration. The algorithm some times cannot distinguish between a **Place** - and a **Stack** - skill, because the characteristics of **Place** and **Stack** are actually similar, apart from the object height profile. However, because the objects are relative small and the height of a stacked object does not differ significantly. However, the other two approaches are not suitable, because the distance between finger tips or distance to object are not the primary features to separate the skills **Locate** and **Push** from other skills.

Experiment 3: Pick up, Stack, Push The third experiment includes a **Pick up** - skill, a **Stack** - skill and a **Push** - skill.
The segmentation results from our approach of this experiment are rather stable and the segmentation often matches the ground truth. perhaps because of the

comparably simple scenario and the features are more distinguishable. The other two approaches are not suitable for the same reason as above.

Experiment 4: Push, Pick, Place, Push The fourth experiment includes a **Push** - skill, a **Pick up** - skill, a **Place** - skill and a **Push** - skill. This experiment is designed to test the algorithms's ability to recognize skills which occurs repeatedly. Our proposed approach has difficulty to find the segmentation point that separates the**Move to** skill and **Push** skill. The reason is in the definition of **Push** the segment before actually touching the object has similar characteristics with the **Move to** skill, however, this will not impede the actual execution in the simulator.



(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

(d) Experiment 4

Figure 4.2: Reproduction trajectory and Segmentation trajectory

4.4.1 Objective metrics

The experiment result was further evaluated under other objective metrics. However, because of the limitedness of the data, there is no systematic evaluation available.

Path length One of the advantage of the segmentation is switching between different control forms: in this thesis, the program replace all data segment labled with the **Move to** or **Move with** with linear point-to-point movement. Theoretically the robot can execute of these segments with a higher velocity to further facilitate the demonstration, however, because the robot becomes still after executing each segment to secure data transmission, monitoring the execution time is not meaningful in this case. Instead, the path lengths have been calculated to prove the advantage of switching between control schemes:

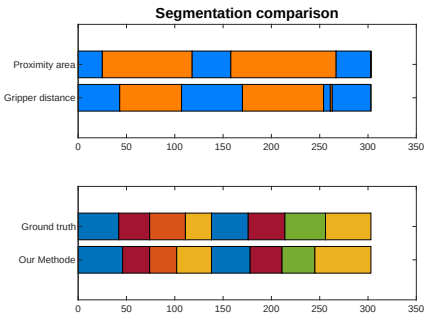
	Experiment 1	**Experiment 2**	**Experiment 3**	**Experiment 4**
Reproduction	2.0204	2.1774	1.5876	1.9319
Segmentation	1.8930	2.1187	1.5035	1.6281
Ratio	93.69%	97.30%	94.70%	84.28%

Table 4.1: Path length comparison

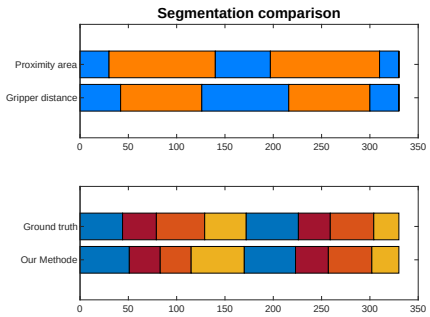
4.4.2 Subjective metrics

Simulator success rate After the segmentation process, the segmented motion trajectory with the skill information will be passed into the simulator to evaluate the efficacy and correctness of the segmentation algorithm. It was clear that the segmentation procedure has to be further improved. For example, if a data segment has been labled with the skill name **Pick up**, the program will follow this trajectory segment and send a signal to close the gripper. If the demonstration was fast which means the periode of time for closing the gripper is short, the robot might not be able to successfully grasp the object, because even if the segmentation

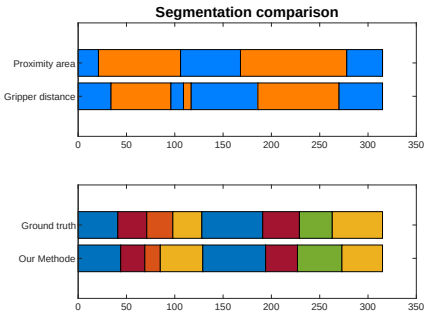
Understanding the movement The major advantage of our approach is it give the user a better understanding of the demonstration and based on the knowledge to create more sophisticated tasks. The extracted skills serve as the fundamental for an intuitive programming of robot application of non-expert users.



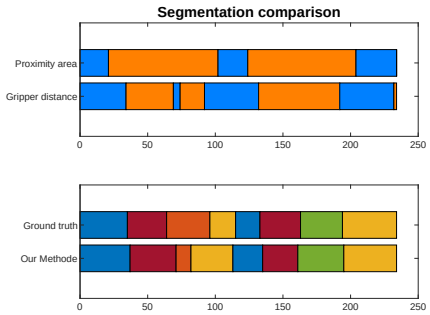
(a) Demonstration 1



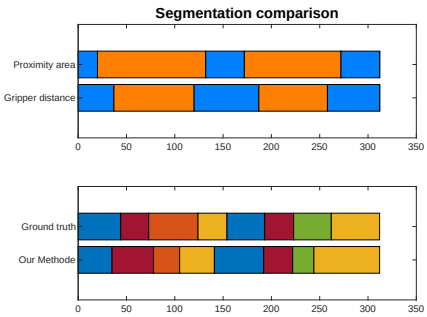
(b) Demonstration 2



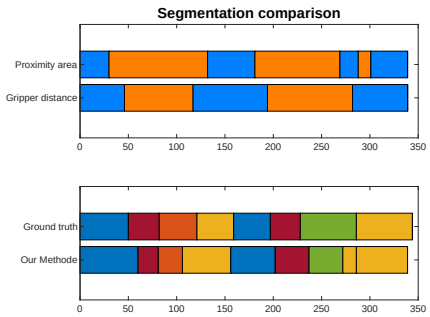
(c) Demonstration 3



(d) Demonstration 4

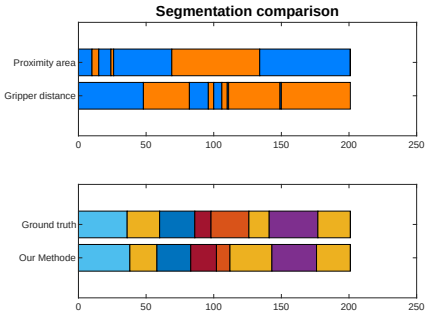


(e) Demonstration 5

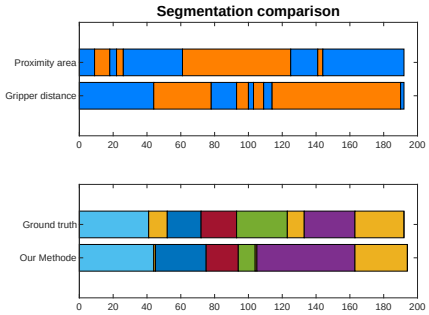


(f) Demonstration 6

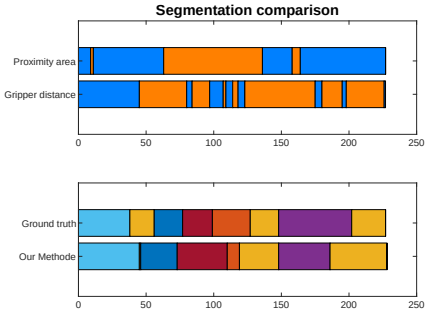
Figure 4.3: Segmentation result: Pick up, Place, Pick up, Stack



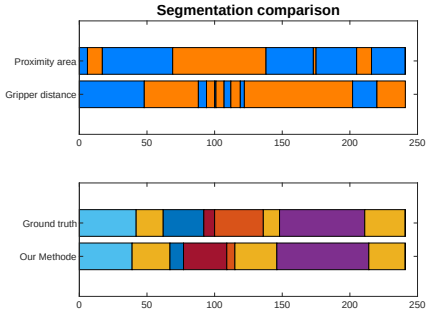
(a) Demonstration 1



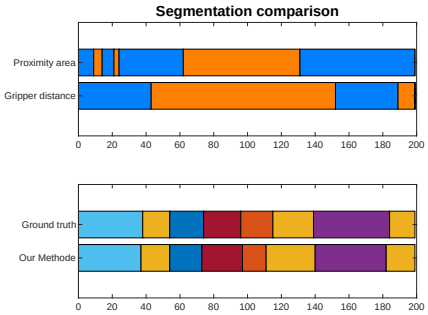
(b) Demonstration 2



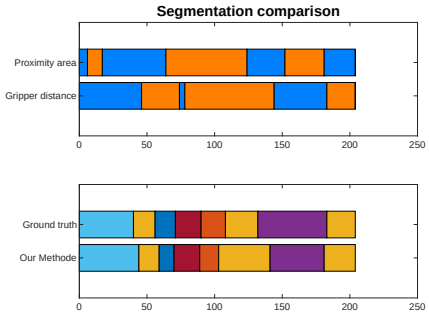
(c) Demonstration 3



(d) Demonstration 4

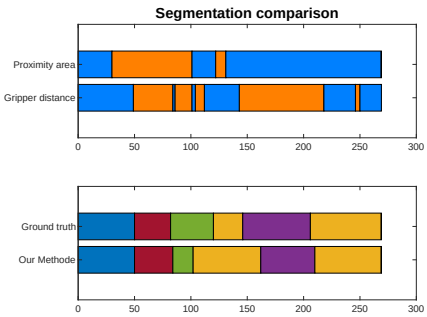


(e) Demonstration 5

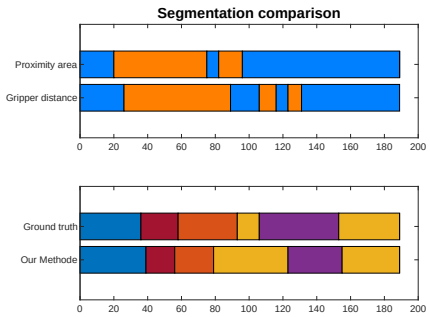


(f) Demonstration 6

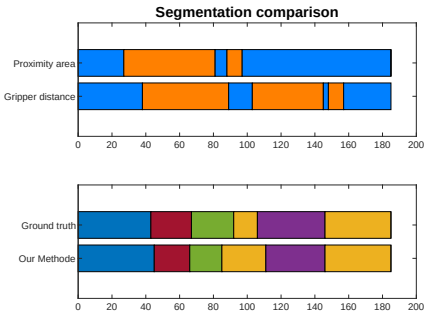
Figure 4.4: Segmentation result: Locate, Pick up, Stack, Push



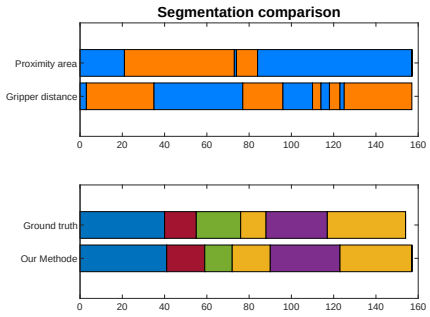
(a) Demonstration 1



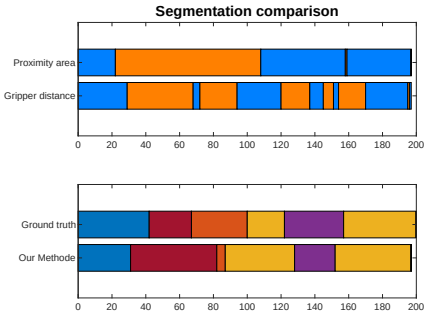
(b) Demonstration 2



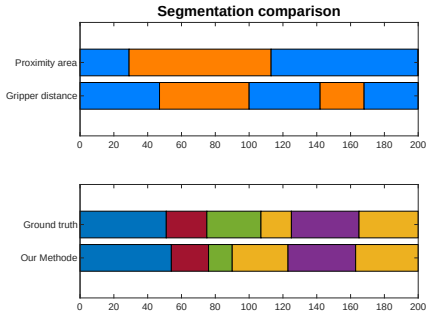
(c) Demonstration 3



(d) Demonstration 4

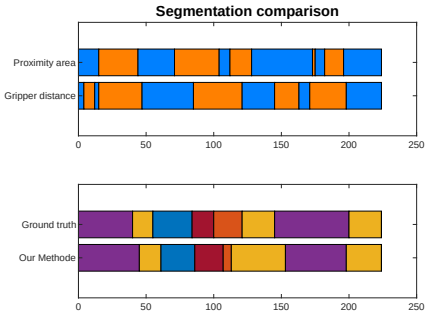


(e) Demonstration 5

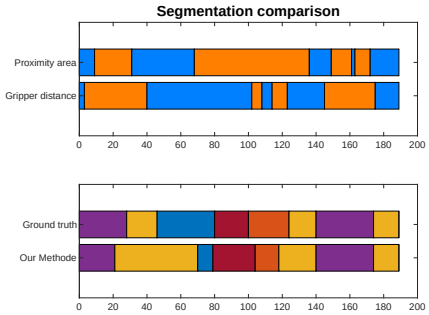


(f) Demonstration 6

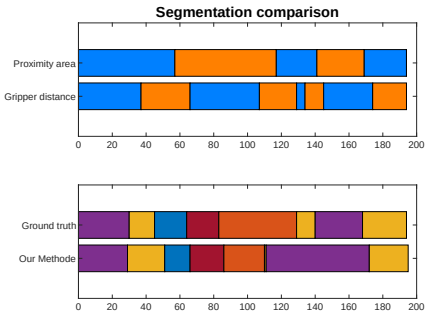
Figure 4.5: Segmentation result: Pick up, Stack, Push



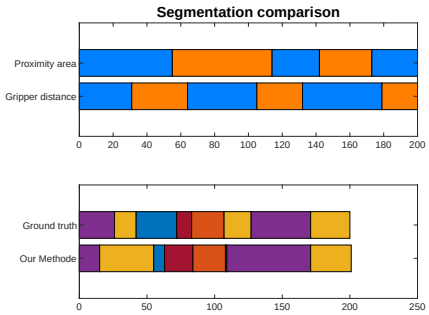
(a) Demonstration 1



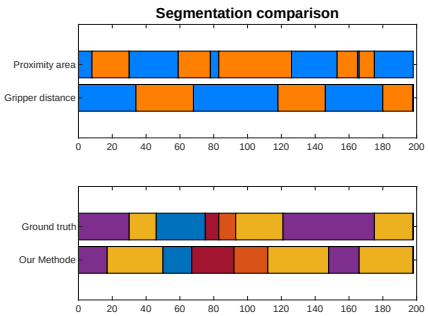
(b) Demonstration 2



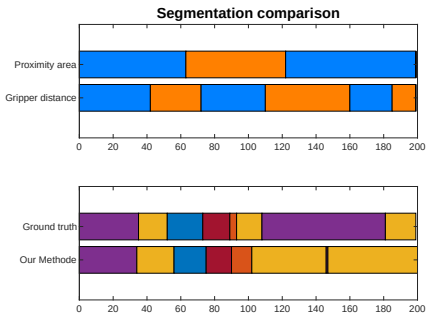
(c) Demonstration 3



(d) Demonstration 4



(e) Demonstration 5



(f) Demonstration 6

Figure 4.6: Segmentation result: Push, Pick, Place, Push

Chapter 5

Conclusion

5.1 Conclusion

In this paper we have presented a Learning by Demonstration algorithm, which uses a motion sensing device to record human demonstrations. A hand-pose estimation algorithm has been applied to determine the position of the demonstrator's finger tips and ankles. The position and orientation of the objects are tracked with AR tags. Further, the program has managed to define a robust mapping between the robot's gripper and the demonstrator's hand by creating a coordinate system around the demonstrator's hand and calculates the necessary transformation matrices. After calculating the relation between the demonstrator's hand position and orientation and the reference frame a robot in the simulator can be taught to reproduce the demonstration. Further, we propose a segmentation algorithm based on the Hidden Markov Model (HMM) to segment the human demonstration into semantic meaningful motion primitives, so-called skills. With the segmentation, the program can optimize the demonstration by automatically controlling the gripper and switching between different control modes. The efficacy of the segmentation algorithm has been tested with multiple demonstrations. At the end, with the help of DMP algorithms, the program exceeds the mere reproduction and offers a more generalized approach.

5.2 Limitations

Despite the benefits that our approach offers, the are still some major limitations in the proposed approach: First, the demonstration has to be conducted with great caution, generally a better recognition rate will be achieved under lower velocity. The reason is the templates were generated with low velocity, because the minimum number of data is 30 and therefore requires a demonstration to be at least 7 to 8 seconds.
Secondly, the segmentation procedure. Because the human motion segmentation is

based upon templates, which are defined and generated by the author, the segmentation is not generalized enough. If the program has detected unknown movements, it will try to find an existing skill to represent the observed movement and thus deliver meaningless segmentation.

5.3 Future Work

Though the proposed LfD policy provides a possibility to transfer task knowledge to a robot, there still exist many limitations, which require further research.

Demonstration: data recording The major limitation and the biggest challenge during the implementation was the data generation process, i.e. how to generate reliable data, that meets all the requirements, is the major issue. Depending on the lightning conditions, more than half of the recorded demonstrations are not usable, this has significantly impeded the data generation process. The problem is not only caused by the physical environment, but also because of the algorithm used. The hand-pose estimation algorithm [LL19] was trained without holding an object, the result is during the demonstration the demonstrator has to be extremely careful and constantly monitor the motion velocity and hand pose, otherwise the deformation of the hand will cause the recorded data to be unusable. This also hindered the implementation and the possibility of the program.

Template For each skill 9 demonstrations were used to generate a template HMM model. The reason for this relative small number of demonstration is the instability of the data recording. In order to generate 9 demonstrations which contain the characteristics of the skill, several dozens of demonstrations were made. However, a larger data set of demonstrations will certainly cover more variations of the skills and contributes to a better segmentation procedure.

Task parameterized GMM At the beginning of this thesis it was actually intended to use Task-parameterized Gaussian Mixture Models (TP-GMM) to make the learned demonstration more adaptive and generalized, however, because of the relative small work space DMP is also able to provide reliable and correct trajectory calculation and also the limited time range, TP-GMM has not been implemented. By applying the methods like TP-GMM, the learned demonstrations will certainly be more robust in complex environments and other challenges like obstacles avoidance could also be tackled.

List of Figures

Acronyms and Notations

HRC Human-Robot Collaboration

HRI Human-Robot Interaction

HRT Human-Robot Team

Bibliography

[ABRW14] Anne Angermann, Michael Beuschel, Martin Rau, and Ulrich Wohlfarth. *MATLAB-Simulink-Stateflow: Grundlagen, Toolboxen, Beispiele*. Walter de Gruyter, 2014.

[ACVB09] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[Cal16] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.

[CL17] Sylvain Calinon and Dongheui Lee. Learning control. 2017.

[Col15] Ian R. Cole. Modelling cpv, Jan 2015. URL: `https://hdl.handle.net/2134/18050`.

[CSFL19] Riccardo Caccavale, Matteo Saveriano, Alberto Finzi, and Dongheui Lee. Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction. *Autonomous Robots*, 43(6):1291–1307, 2019.

[DXW+16] Hao Deng, Zeyang Xia, Shaokui Weng, Yangzhou Gan, Peng Fang, and Jing Xiong. A motion sensing-based framework for robotic manipulation. *Robotics and biomimetics*, 3(1):23, 2016.

[DZML12] Guanglong Du, Ping Zhang, Jianhua Mai, and Zeling Li. Markerless kinect-based hand tracking for robot teleoperation. *International Journal of Advanced Robotic Systems*, 9(2):36, 2012.

[Fus] Matt Fussell. How to draw hands - the ultimate guide. [Online; accessed March 12, 2020]. URL: `https://thevirtualinstructor.com/how-to-draw-hands.html`.

[Gar] Damien Garcia. Retrieved March 12. URL: `smoothn(https://www.mathworks.com/matlabcentral/fileexchange/25634-smoothn),MATLABCentralFileExchange`.

[KSL94] Thomas P Krauss, Loren Shure, and John Little. *Signal processing toolbox for use with MATLAB®: user's guide*. The MathWorks, 1994.

[LC15] Hsien-I Lin and YP Chiang. Understanding human hand gestures for learning robot pick-and-place tasks. *International Journal of Advanced Robotic Systems*, 12(5):49, 2015.

[LK13] Jonathan Feng-Shun Lin and Dana Kulić. Online segmentation of human motion for automated rehabilitation exercise analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):168–180, 2013.

[LL19] Shile Li and Dongheui Lee. Point-to-pose voting based hand pose estimation using residual permutation equivariant layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11927–11936, 2019.

[NS] Scott Niekum and Isaac IY Saito. Ar track alvar ros package. *URL http://wiki. ros. org/ar_ track_alvar*, 3(4).

[PL18] Affan Pervez and Dongheui Lee. Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 11(1):61–78, 2018.

[PNA+16] Mikkel Rath Pedersen, Lazaros Nalpantidis, Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh, Volker Krüger, and Ole Madsen. Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, 37:282–291, 2016.

[QCG+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[Rab89] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RK04] Chotirat Ann Ratanamahatana and Eamonn Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 11–22. SIAM, 2004.

[RSF13] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

License