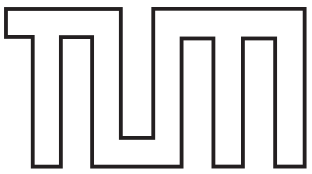


Technische Universität München

Chair of Sensor Based Robotic Systems and
Intelligent Assistance Systems

Prof. Dr. Albu-Schäffer

Bachelor Thesis

Enhancing Probabilistic Imitation Learning with Robotic Perception For Self-Organising Robot Workstation

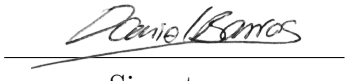
Author:	Daniel Barros
Matriculation Number:	03755117
Address:	Steinickeweg 7
	80798 Munich
Advisors:	Prof. Dr. Alin Albu-Schäffer
	M.Sc. Abhishek Padalkar
	M.Sc. Christoph Willibald
	Dr. João Silvério
Begin:	01.05.2024
End:	01.08.2024

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, August 19, 2024
Place, Date


Signature

München, August 19, 2024
Place, Date


Signature

Acknowledgements

I would like to give my warmest thanks to my supervisor M.Sc. Abhishek Padalkar for giving me the chance to explore robot learning and carry out this work at DLR (Deutsches Zentrum für Luft- und Raumfahrt, German Aerospace Center), your knowledge and enthusiasm for robotics is great inspiration and motivation to get things done. Warm thanks to M.Sc. Christoph Willibald for co-supervising this thesis and the excellent support especially during the last stretch of the project.

I want to thank M.Sc. Maximilian Mühlbauer for the patient and reliable help with using his implementation of the probabilistic imitation learning algorithms. A big thanks also goes to M.Sc. Wout Boerdijk for enabling the Computer Vision component of this thesis through energetic, to-the-point guidance on setting up the pipeline.

It was fantastic to always be able to ask technical questions to the group leader Dr. João Silvério and get thoughtful answers that enhanced my understanding and sparked other ideas, muito obrigado. Thank you to Dr. Freek Stulp for the personal guidance, the excellent, open department atmosphere and for giving me a wider perspective during the KRO Seminar.

I would like to acknowledge all RMC members for sharing their great research and insights and making me always feel like I am part of the team, special thanks to Dr. Thomas Eiband, Dr.Ing. Korbinian Nottensteiner, M.Sc. Markus Knauer, M.Sc. Stefan Schneyer, B.Sc. Timo Bachmann, M.Sc. Leonard Klüpfel and M.Sc. Dominik Winkelbauer. I extend my deepest gratitude to M.Sc. Florian Schmidt and Dipl.Ing. Oliver Eiberger, whose expertise and dedication make it possible to work with SARA effectively on a daily basis. I would also like to warmly thank B.Sc. Connor Herron for reviewing this thesis and all the other students who fostered a motivating and supportive atmosphere in the office: Clemente, Pirmin, Carsten, Aníbal, Hannah, Federico, Tim, Lugh, Janosch, Marco, Britt, Konrad, Valentina and Milena.

I want to thank Prof. Dr. Albu-Schäffer for the precise feedback and the opportunity to write this thesis at DLR.

Finally, I would like to express my profound gratitude to my parents Ana and João and my girlfriend Linda for their unwavering support and encouragement throughout the course of this thesis.

Abstract

The increasing adoption of robotic automation across various industries is driving extensive research into efficient and intuitive methods for transferring skills from humans to robots. Traditional robot programming techniques, along with the expertise they require, remain significant obstacles to scalability of robotic deployment and integration. Learning from human demonstrations has emerged as a promising approach to teach robots complex behaviors while making robot programming more accessible. For this approach to be effective, the demonstration process must be user-friendly, and the imitation learning algorithms must reliably generate the movements necessary to complete tasks in current and future manufacturing contexts. To address these needs, we develop a demonstration recording framework and combine it with state-of-the-art techniques from imitation learning and computer vision, forming an end-to-end software pipeline for teaching and deploying robotic skills. Our system is implemented on the Safe Autonomous Robotic Assistant (SARA) robot to record kinesthetic demonstrations and teach it to self-organize its workstation by picking up scattered objects. We utilize Kernelized Movement Primitives (KMP) to learn the movements and adapt them to different environment configurations by adding start and end poses as via points. A You Only Look Once (YOLO) vision model is trained to detect objects on the workstation, and the 3D pose is estimated from the 2D bounding box. This vision system serves two primary functions: first, it aids in automatically recording object-centric demonstrations, eliminating the need for manual object pose determination by the programmer; second, it ensures precise interaction with objects regardless of their pose on the workstation. During deployment, it helps generalize the learned task by providing the pose of the objects to be grasped. The developed system achieves reliable detections and pose estimations for a variety of object poses on the workstation, which the robot is able to clean up effectively and autonomously.

Contents

Chapter 1

Introduction

1.1 Bridging Imitation Learning and Computer Vision

Imitation learning is an established approach for transferring skills to robots by allowing them to learn tasks through observation and replication of human demonstrations [BCDS08]. Probabilistic imitation learning models also exhibit the ability to extrapolate to unseen situations [HRSC19]. This reduces the effort needed to program intricate, adaptable motions manually, and robots can therefore acquire complex behaviors that would otherwise require extensive coding and fine-tuning. It facilitates intuitive programming and makes advanced robotic capabilities accessible to a broader range of users.

In the context of intuitive programming, the importance of user-friendly and efficient interfaces for the demonstration process cannot be overstated. These interfaces enable users to provide demonstrations in a natural and straightforward manner, ensuring that the learning process is as seamless as possible. A well-designed demonstration interface can bridge the gap between human intention and robotic execution, allowing users to impart nuanced and sophisticated skills to robots with minimal effort [RPCB20].

Furthermore, imitation learning represents a good attempt at capturing the subtleties and intricacies of human motion. Humans naturally perform tasks with a level of fluidity and adaptability that is difficult to replicate through traditional programming methods. By observing and mimicking human actions, robots can learn to perform tasks in a way that is more aligned with human expectations and behaviors. This capability is crucial in applications where human-robot interaction is essential, such as in assistive robotics [LCGZ17], collaborative manufacturing [RCC+16], and service robotics [UASvdS04].

Computer Vision [MMMF24] can significantly enhance the capability of imitation learning models to extrapolate to unseen situations. By detecting objects and estimating their poses, vision systems provide critical environmental information that can be fed into imi-

tation learning algorithms. This integration allows the learned trajectories to be adapted dynamically based on the real-time location and orientation of objects. As a result, robots can respond more flexibly to variations in their environment [CZH+24], making the learned behaviors more robust and versatile.

There are three main types of learning from demonstration: learning from observation, i.e. seeing the human complete the task with a camera system; learning via teleoperation, i.e. an operator completes the task on the robot while recording data; and learning from kinesthetic demonstration, i.e. the human demonstrator guides the robot through the task with his hands. The latter is the focus of this thesis.

In this thesis, we will explore the application of vision-enhanced imitation learning in a self-organizing robotic workstation. We will delve into the mechanisms of how robots can learn from demonstrations, the challenges involved, and the solutions proposed to enhance both the robot's learning process and the user's teaching experience.

1.2 Objectives

The primary objective of this thesis project is to develop an end-to-end pipeline that integrates imitation learning and computer vision. This pipeline will then be used to teach the SARA (Safe Autonomous Robotic Assistant) [IOE+20] robot the skill of organising its own workstation by removing the grid clamps that clutter it. Grid clamps are small black objects that can be inserted in the holes of the metallic rails on the SARA workstation as depicted in Figure 1.1.

(a) Grid Clamp

(b) DLR-SARA robot and workstation

Figure 1.1: Setup for Self-Organizing Workstation Task

The specific deliverables set for this thesis project are:

- **A user-friendly demonstration framework**: Create an intuitive system for users to demonstrate tasks to the robot, facilitating easy and efficient programming. Deploy a concrete implementation for recording demonstration data on the SARA robot.
- **Robust object detection and pose estimation**: Set up and calibrate a camera system and train state-of-the-art computer vision models to ensure accurate and reliable detection and localization of grid clamps on the workstation.
- **Trajectory learning**: Set up an imitation learning pipeline which applies DLR implementations of Gaussian Mixture Model/Regression and Kernelized Movement Primitives to the data in order to retrieve and adapt trajectories.
- **End-to-end pipeline**: Develop a cohesive system that combines the vision pipeline with imitation learning algorithms to dynamically adapt to changes in the environment and improve the robustness of task execution.
- **Evaluate system performance**: Conduct testing and analysis to assess the effectiveness of the integrated system in autonomously cleaning up the workstation. Measure the accuracy of object detection and pose estimation, the reliability of trajectory execution, and the overall success rate of task completion.

These objectives guide the development and evaluation of the following system, ensuring that it meets the needs of both novice and expert users while leveraging the state of the art in robotic learning and perception.

1.3 System Design

The proposed system (Figure 1.2) is designed to facilitate the seamless integration of demonstration recording, imitation learning, and computer vision to enhance the capabilities of the SARA robot. The system design is organized into three main components.

Demonstration Recorder: The first component of the system focuses on recording demonstrations to provide the initial data required for imitation learning. The framework includes a graphical user interface (shown in Figure 1.3) that offers an intuitive and user-friendly programming experience, allowing users to record demonstrations with minimal effort. To ensure the accuracy and usability of the recorded data for imitation learning, the system preprocesses the trajectory data by aligning it in time and projecting it to a local frame (see Section 3.1).

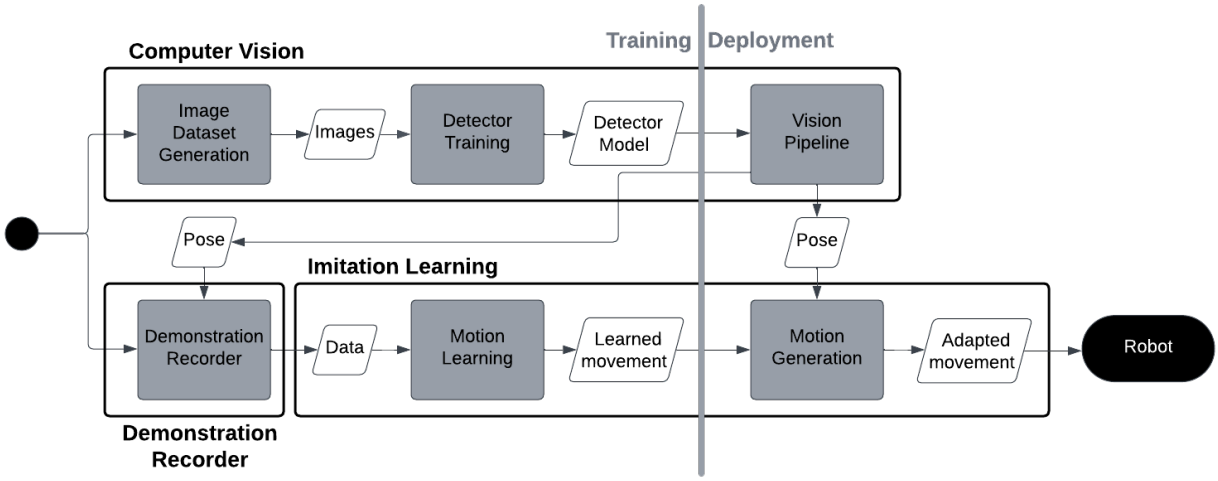


Figure 1.2: System overview with training and deployment phases.

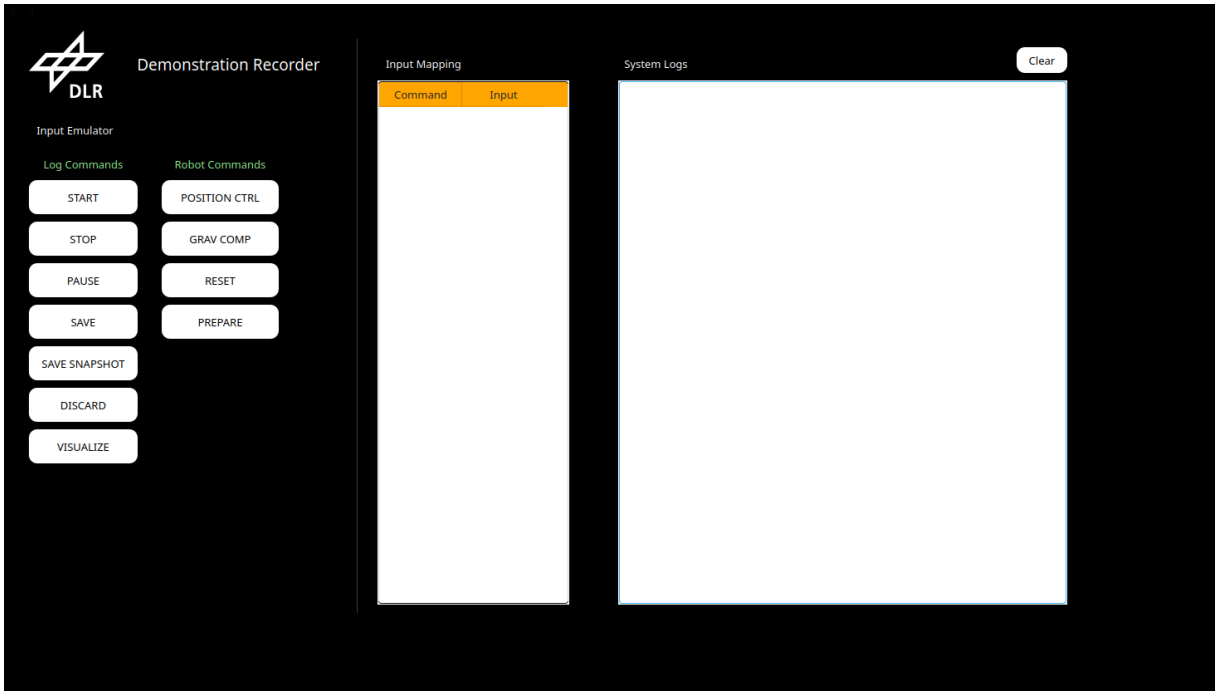


Figure 1.3: Graphical user interface for recording demonstrations implemented with PyQt5 including buttons to control the robot and the logger and text fields to display input mapping and system logs.

Imitation Learning: The second component involves using imitation learning algorithms to derive and refine the robot's trajectory from the recorded demonstrations. The system utilizes probabilistic methods, specifically Gaussian Mixture Model (GMM) and Gaussian Mixture Regression (GMR), to model the distribution of the data and retrieve

a probabilistic reference trajectory from the demonstration data (Motion Learning in Figure 1.2). Additionally, Kernelized Movement Primitives (KMP) is employed to generate the final local trajectory, providing flexibility and adaptability in the robot's movements (Motion Generation in Figure 1.2). This is discussed at length in Section 3.2.

Computer Vision: The third component integrates computer vision in two key areas: first, with the demonstration recorder to facilitate the projection of demonstration data into the local object frame; and second, with imitation learning to detect grid clamps on the SARA workstation and provide their poses to the learning algorithms. Capturing local, object-centric demonstrations is crucial for generalizing tasks to new object poses, as it encodes the interaction with the object independent of its position on the workstation. We explore this in Section 3.3.

The system's operation can be divided into two main phases as depicted in Figure 1.2: training and deployment.

During the training phase, the SARA robot records data from human demonstrations of the task using the demonstration recording framework. The collected trajectory data is then preprocessed to align in time and be transformed into the local object frame. The processed data is used to train the GMM and GMR algorithms to retrieve a reference trajectory, and KMP is applied to learn the final local trajectory. In parallel, an image dataset depicting the workstation scene is generated and used to train the YOLO detector for object recognition and pose estimation.

In the deployment phase, the system operates by first using the vision pipeline to capture the scene and identify the objects and their poses through the YOLO detector. Using the detected object pose as the reference frame, the system runs the learned trajectory on the SARA robot to perform tasks such as grasping grid clamps. There is also an option to include via points and query Kernelized Movements again to generate smoother trajectories, enhancing the flexibility and precision of the robot's actions when cleaning up its own workstation.

Chapter 2

Related Work

Imitation Learning: [ACVB09] frames imitation learning (IL) as a subset of Supervised Learning, where the agent learns from labeled training data composed of task demonstrations by a teacher. In this framework, the world consists of states and actions, with the learner accessing observed states through a probabilistic transition function, and selecting actions based on the observations via a policy. The efforts of the IL community primarily lie in reducing the effort and expertise needed to derive robust policies. Examples of high-expertise approaches are the traditional model-based policies, which largely depend on the derivation of an accurate world model and reinforcement learning, which requires precise design of a reward function. [BCDS08] discusses how early IL used graph-based encoding of demonstrations and symbolic reasoning to retrieve state-action-state transitions that achieve the subgoals of a task, which are learned from demonstrations. These techniques employed for example Hidden Markov Models [HSM96], which can also be used in the context statistical modeling of skills [TL96]. These address and leverage the naturally high variability in human demonstrations and are able to eliminate the need for interpolation techniques that arises when trajectories generated from HMM systems exhibit discontinuities. Several algorithms have been developed to generate smooth trajectories from demonstration data, such as Gaussian Mixture Model with Gaussian Mixture Regression (GMR) and Probabilistic Movement Primitives (ProMP). These are probabilistic methods that are able to retrieve trajectories to complete tasks, however their extrapolation to unseen situations are limited. Adaption with via points with GMM/GMR requires costly recomputing of parameters. ProMP can be used to introduce via points to the trajectory, however it relies on basis functions, which adds computational complexity as more parameters need to be estimated. Kernelized Movement Primitives is a fairly novel method, which addresses these limitations.

Kernelized Movement Primitives (KMP): Firstly derived in [HRSC19], KMP is a non-parametric method that does not depend on explicit basis functions. It also models trajectories with Gaussian components, but it removes basis functions through a kernel treat-

ment of the solutions for mean and covariance of each estimated Gaussian. This addresses the well-known curse of dimensionality in machine learning [VF05] by enabling applications with high-dimensional inputs and outputs. KMP uses an information-theoretic approach to minimize distance between probability distributions. It succeeds at adaption with trajectories with via points and extrapolation to unseen situations via an extension where trajectories are represented locally in the object frame. It has been used successfully for a reaching task with force-based adaptations, a 3rd-hand soldering task [HRSC19], painting task [HADSC20] and for movements on an upper limb rehabilitation robot [LAL+24]. These properties are also leveraged in this thesis project to accommodate varying object poses on the SARA workstation by adapting trajectories which are represented relative to the object frame.

Riemannian Manifolds: To handle the complexity of orientation data and other non-Euclidean features, Riemannian manifolds have been introduced into IL frameworks [ZHS+17]. Unlike Euclidean spaces, Riemannian manifolds provide a geometric structure that accommodates curved spaces, enabling more accurate representation and manipulation of orientation data. An example o such a representation in robotics is quaternions, which reside in a unit sphere, see [HADSC20]. The use of Riemannian manifolds allows for operations like the exponential and logarithmic maps, which facilitate the transformation between the manifold and its tangent space, making computations on these spaces more tractable. Techniques such as Gaussian conditioning and Gaussian product on manifolds have been developed to extend traditional probabilistic methods to these curved spaces, ensuring that the learned models accurately reflect the underlying geometry of the data, see [ZHS+17].

Computer Vision in Imitation Learning: Many different interfaces can be used to record demonstration data as well as information about the environment. As mentioned in [BCDS08], traditionally data was recorded while a human operator completed the task via teleoperation. Vision can also be used to make robots systems observe and replicate human gestures [KI95] and to acquire environment information and constraints such as poses of objects. In this thesis we use vision for the latter and kinesthetic teaching to record the demonstration data as in [CGB07]. We use YOLOv7 [WBL23] which is the state-of-the-art in object detection. The "You only look once" model introduced by [RDGF16] frames object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Competing variants, such as YOLOX [GLW+21] and PP-YOLOE [XWL+22], also offer high performance but typically lag behind YOLOv7 in terms of a balance between speed and accuracy.

This thesis proposes a comprehensive system that integrates imitation learning techniques and advanced computer vision to develop an end-to-end learning pipeline for robotic applications. By leveraging Kernelized Movement Primitives (KMP) for adaptive trajectory generation, Riemannian manifolds for accurately representing and manipulating non-Euclidean orientation data, and the state-of-the-art YOLOv7 model for precise object detection, the

implemented system aims to equip the SARA robot with the capability to autonomously learn and perform the self-organizing workstation task. This holistic approach ensures that the robot can not only learn from human demonstrations but also adapt to varying task conditions and environments, thus showcasing the potential of combining these cutting-edge methodologies to enhance robotic learning.

Chapter 3

Self-Organizing Robot Workstation

This chapter presents a deeper dive into the design and implementation of the subsystems put forward in the Introduction. We discuss how the system was applied as an end-to-end pipeline to teach the SARA robot how to clean up its own workstation. Algorithm 1 provides an overview of the entire process:

Algorithm 1 *Vision-enhanced Local Kernelized Movement Primitives with Via Points*

1: Initialization
- Define the kernel function and model parameters.
- Determine the local object frames with vision.

2: Learning from local demonstrations
- Collect demonstrations in the robot base frame.
- Project demonstrations into local frames.
- Perform dynamic time warping.
- Extract local reference database using GMM and GMR.

3: Deployment on the robot
- Determine the local object frame with vision.
- Project start and end points into local frame.
- Update local reference database.
- Predict the trajectory in local frame using KMP.
- Compute trajectory in base frame.
- Run movement on robot.

3.1 Demonstration Recorder

3.1.1 Robot Demonstration Recorder Framework:

The demonstration recording system is composed of several abstract subsystems. The functionality of each of them is explained in this section along with notes on the concrete implementation for the SARA robot as depicted in Figure 3.1.

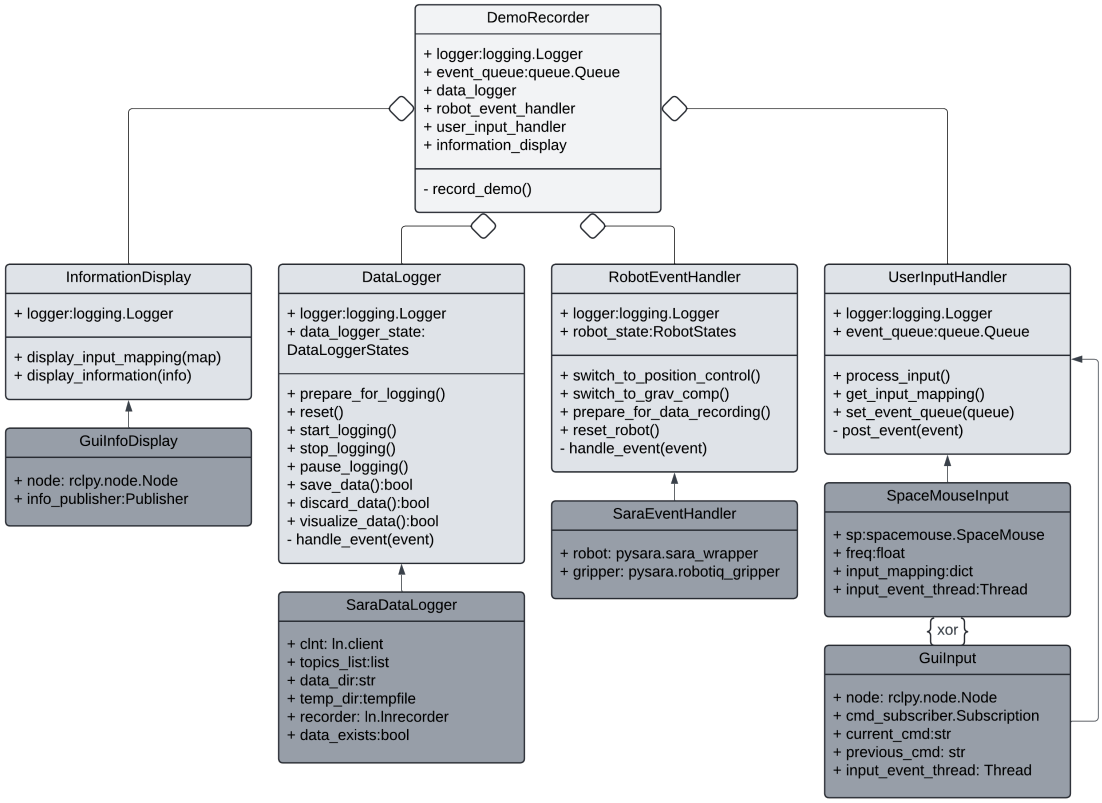


Figure 3.1: UML Class Diagram of the Demonstration Recorder Framework including Abstract Classes and Implementation for SARA

Data Logger: This subsystem is responsible for the recording of data during demonstrations. It includes methods for preparing, starting, pausing, and resetting recordings, along with saving, discarding, and visualizing data. The core of this subsystem is a Finite State Machine (FSM) that manages the states and transitions of the recorder based on user inputs. For the SARA robot, the Data Logger is implemented using the lnrecorder, which is part of Links and Nodes (LN). LN is a middleware designed to create and manage flexible distributed real-time systems, specifically for embedded robotic systems, and is comparable to the widely used robotics middleware ROS (Robot Operating System). The LN recorder is implemented as an LN client that is instantiated with a regular expression pattern describing a set of topic or service names and the frequencies at which their data

should be recorded. When messages are sent on any of these topics, the content is stored as a time series. For this thesis project, the recorded topic publishes robot pose data, which includes 3D Cartesian position and a unit quaternion as the orientation representation. During a demonstration, the user performs tasks with the robot and, upon requesting to save, the data is stored in the chosen folder. The plots in Figure 3.2 show the cartesian position recorded during demonstrations of the self-organising workstation task.

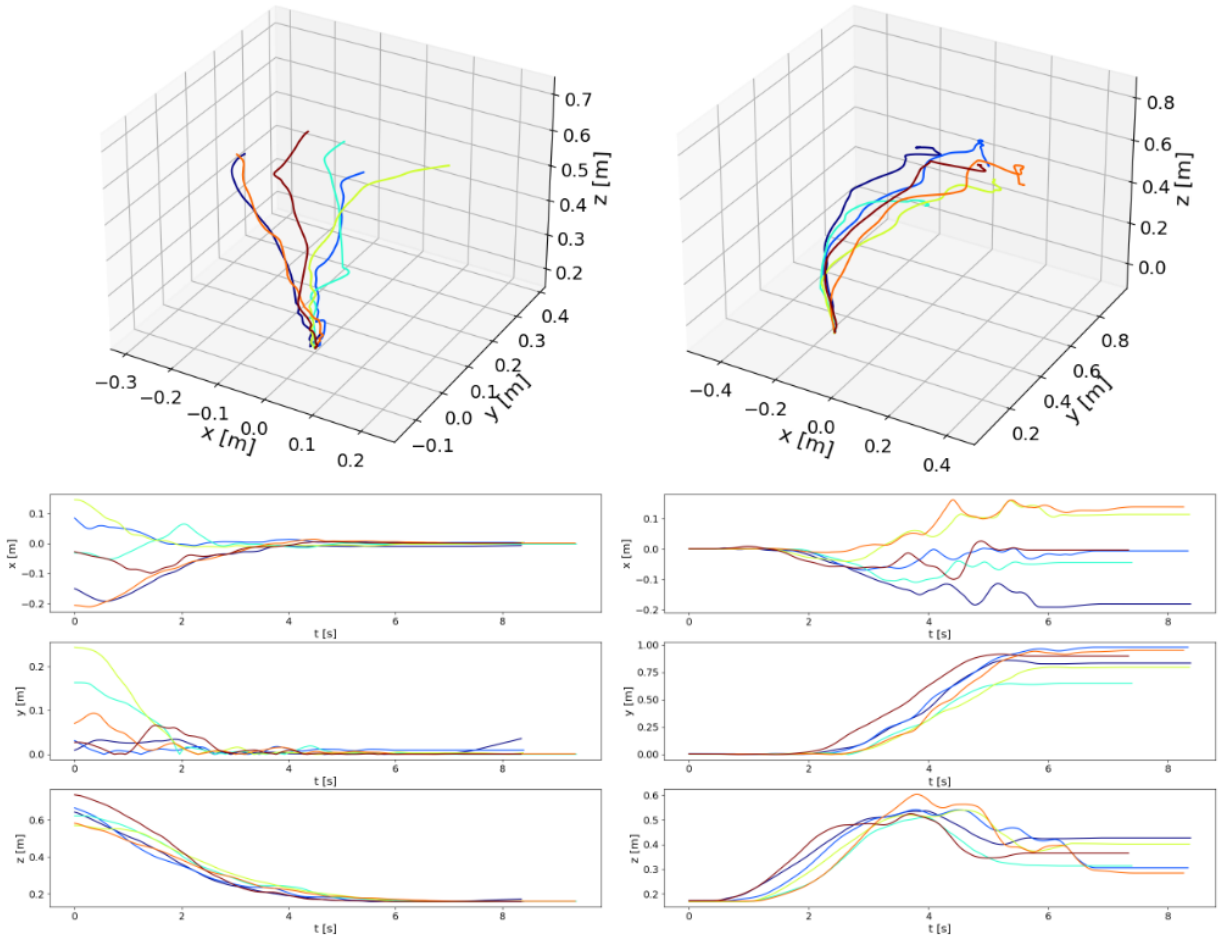


Figure 3.2: 3D and time plots of 6 demonstrations of the pick (left) and place (right) subtasks of the self-organising robot workstation

Robot Event Handler: This module also utilizes a FSM to handle events, in this case related to robot control actions. It manages tasks such as preparing for data recording, resetting the robot, switching the robot to position control or gravity compensation mode. The latter is used for the kinesthetic demonstrations as shown in Figure 3.3, as the robot is in a compliant state that enables a demonstrator to move it through the workstation. The concrete implementation involves calling the functions from SARA's Python API.

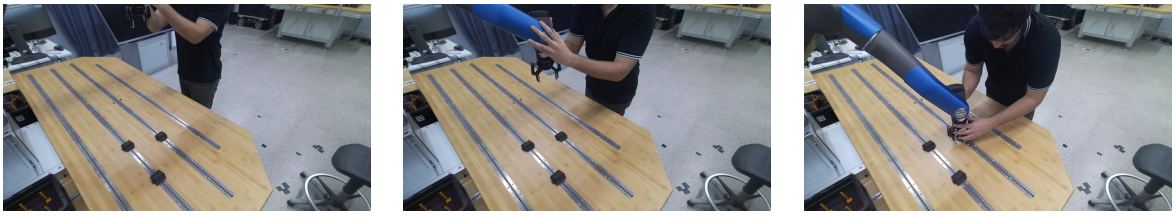


Figure 3.3: Recording kinesthetic demonstrations

User Input Handler: This subsystem manages the interaction between the human demonstrator and the system. It defines the mapping of user inputs to system commands and sends events to be processed by the FSMs of the Data Logger or the Robot Event Handler. In the concrete implementation, a software queue is used to handle events in a thread-safe manner. Two classes were implemented to support inputs from buttons on a Graphical User Interface (GUI) and a 3D Spacemouse, ensuring a more efficient, versatile and user-friendly interaction experience.

Information Display: This module provides methods to display system information to the human demonstrator. In the implementation, it focuses on logging actions of the system within the GUI *System Logs* box, see Figure 1.3. The interprocess communication between this system and the GUI process is achieved using ROS2, which ensures a robust, real-time, and scalable framework for seamless communication by utilizing its data-centric, publish-subscribe architecture. This setup ensures that system actions and events are logged and displayed to the user in a clear and timely fashion.

Demo Recorder: Acting as the orchestrator of the system, the Demo Recorder instantiates and connects all the other subsystems. It sequentially pulls elements from the User Input Handler's event queue and delegates the events to the appropriate subsystems, either the Data Logger or the Robot Event Handler.

This framework provides a robust and accessible demonstration recording process by:

- allowing teams to implement the system for various robotic setups with minimal effort, thanks to its modular and flexible design
- offering a user-friendly way to control the SARA robot
- giving useful feedback to the human demonstrator during demonstrations

3.1.2 Preprocessing Demonstration Data:

During the demonstrations, we collect time series of robot poses, which are essentially 8-dimensional arrays comprising the timestamp, 3D position, and a unit quaternion. Thus, these data points are defined on the manifold $\mathbb{R} \times \mathbb{R}^3 \times \mathrm{SO}(3)$. The recorded data is loaded into a software container developed as part of this thesis, which complies with the general DLR framework for storing data across different robotic systems.

Local Projection: The raw data is initially saved in the tool center point (TCP) frame as it is recorded. However, to obtain a local demonstration, one viewed from the perspective of the object, the data is transformed into the object coordinate frame. The origin of the object coordinate frame is determined by the vision system during the demonstrations. For this thesis, only static poses for each demonstration are implemented, however it can be easily be expanded to store a time series of the object's pose. This automated acquisition of the object frame's origin eliminates the need for manual programming by the demonstrator, streamlining the process and reducing potential sources of error. Object-centric trajectories offer several advantages in the context of imitation learning, including enhanced grasping precision due to low covariance in the model close to the object, improved consistency in task execution, and increased robustness to changes in the robot's initial position or orientation.

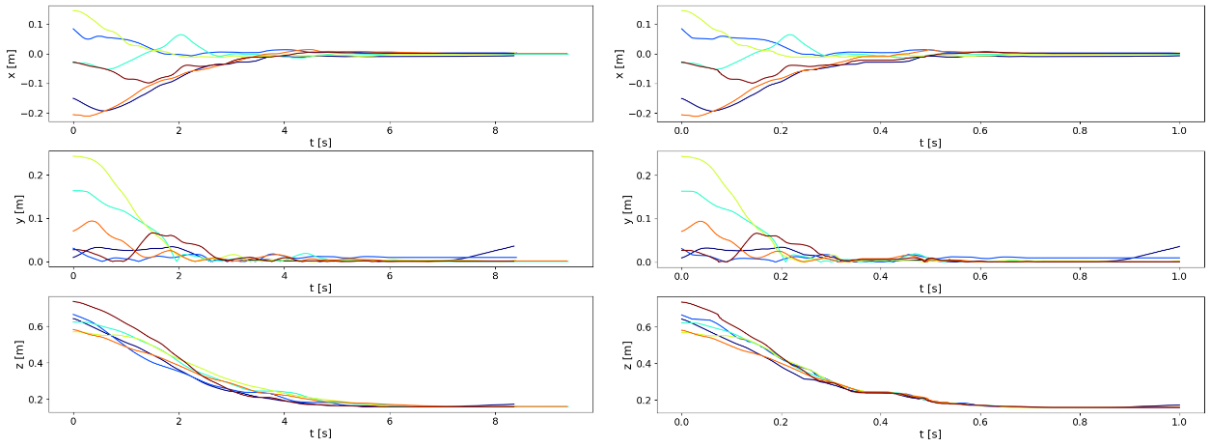


Figure 3.4: Dynamic Time Warping effect of aligning trajectories in time: before (left) and after (right)

Dynamic Time Warping: Dynamic Time Warping (DTW) [Mül07] is a technique used to align time series data by warping the time dimension to achieve the best match between sequences. Traditional DTW cannot be directly applied to our data because it relies on Euclidean distance, which is not suitable for data residing on a manifold. Instead, we use an implementation of DTW that defines the data and distance function to be compliant with manifold properties. This implementation uses a distance function based on geodesic dis-

tances, specifically computing the logarithmic map on the manifold to measure differences between points, ensuring the alignment process respects the geometric properties of the pose data. This approach provides an accurate temporal alignment of the demonstrations, as can be seen in Figure 3.4, while maintaining the integrity of the manifold structure.

By the end of the preprocessing phase, we have a well-structured, temporally aligned dataset transformed into the relevant coordinate frames. Next, we will delve into how this preprocessed data is utilized to train imitation learning models, enabling the robot not only to replicate the demonstrated tasks, but also adapt to new start and end points.

3.2 Imitation Learning Pipeline

Let $\{\{\vec{t}_{n,h}, \vec{\xi}_{n,h}\}_{n=1}^{N}\}_{h=1}^{H}$ denote the set of demonstration data. $\vec{t}_{n,h} \in \mathbb{R}$ is the time input and $\vec{\xi}_{n,h} \in (\mathbb{R}^3 \times \mathrm{SO}(3))$ the pose output. H represents the number of demonstrations and N the trajectory length.

3.2.1 Retrieving Reference Trajectory

The process of retrieving a reference trajectory from the demonstration data involves fitting a Gaussian Mixture Model (GMM) and then performing Gaussian Mixture Regression (GMR). These techniques allow us to model and generalize the demonstrated tasks, creating a probabilistic reference trajectory that accounts for variations and uncertainties in the data.

Gaussian Mixture Model (GMM) Fitting: The first step involves fitting a Gaussian Mixture Model to the $\mathbb{R} \times \mathbb{R}^3 \times \mathrm{SO}(3)$ manifold data. The GMM models the data as a mixture of multiple Gaussian distributions, each defined by a mean and covariance:

$$\begin{bmatrix} \vec{t} \\ \vec{\xi} \end{bmatrix} \sim \sum_{c=1}^{C} \pi_c \mathcal{N}(\vec{\mu}_c, \Sigma_c). \tag{3.1}$$

π_c, $\vec{\mu}_c$ and Σ_c represent the prior probability, mean and covariance of the c-th Gaussian component, respectively. $C \in \mathbb{N}$ denotes the number of Gaussian components, where $C = 10$ is chosen empirically as a value that ensures adequate coverage of the demonstration dataset as depicted in Figure 3.5a. This is an estimation of the joint probability distribution $\mathcal{P}(\vec{t}, \vec{\xi})$ of the demonstration data, which captures its underlying structure and variability.

The GMM fitting process begins with an initial clustering, typically using the k-means algorithm, to provide a starting point for the mixture model. The k-means algorithm clusters the data into k groups, the initial Gaussian parameters (means and covariances) are estimated from these clusters. The GMM is then iteratively refined using the Expectation-Maximization (EM) algorithm. During the Expectation step, the algorithm calculates the likelihood of each data point belonging to each Gaussian component. In the Maximization step, the Gaussian parameters are updated to maximize this likelihood. This process continues until convergence, defined by the change in log-likelihood between iterations falling below a threshold ϵ. The parameters were chosen as $\epsilon = 0.0001$ and $max\_iterations = 3000$.

Gaussian Mixture Regression (GMR): Following the fitting of the Gaussian Mixture Model (GMM) we employ GMR, which leverages the joint probability distribution produced by the GMM and conditions it on the time variable. Specifically, GMR calculates the conditional probability distribution $\hat{\vec{\xi}}_n | \vec{t}_n \sim \mathcal{N}(\hat{\vec{\mu}}_n, \hat{\Sigma}_n)$, yielding the probabilistic

reference trajectory $\{\hat{\vec{\xi}}_n\}_{n=1}^N$ in Figure 3.5b. As put forward in [HRSC19], we can interpret the result also as a *reference database* $\mathbf{D} = \{\vec{s}_n, \hat{\vec{\mu}}_n, \hat{\Sigma}_n\}_{n=1}^N$. It provides not only the expected pose for each time step but also the associated uncertainty represented by Σ. In other words, it accounts for the natural variability in human demonstrations and provides a flexible template for the robot to follow.

3.2.2 Learning Local KMP Trajectory

After retrieving the reference trajectory using Gaussian Mixture Regression (GMR), we refine this trajectory using Kernelized Movement Primitives (KMP). KMP is a powerful tool for learning and generating smooth, flexible trajectories based on kernel methods, which provide several advantages over traditional approaches as discussed in Chapter 1. The main advantage over simply deploying the GMR trajectory directly is the ability to add via points in order to adapt to new task requirements as carried out in [HRSC19].

We query KMP with a linearly spaced time vector $\vec{t}^* = \mathit{linspace}(0,\ 1,\ 100)$ in order to get a predicted trajectory that is similar to training in terms of time:

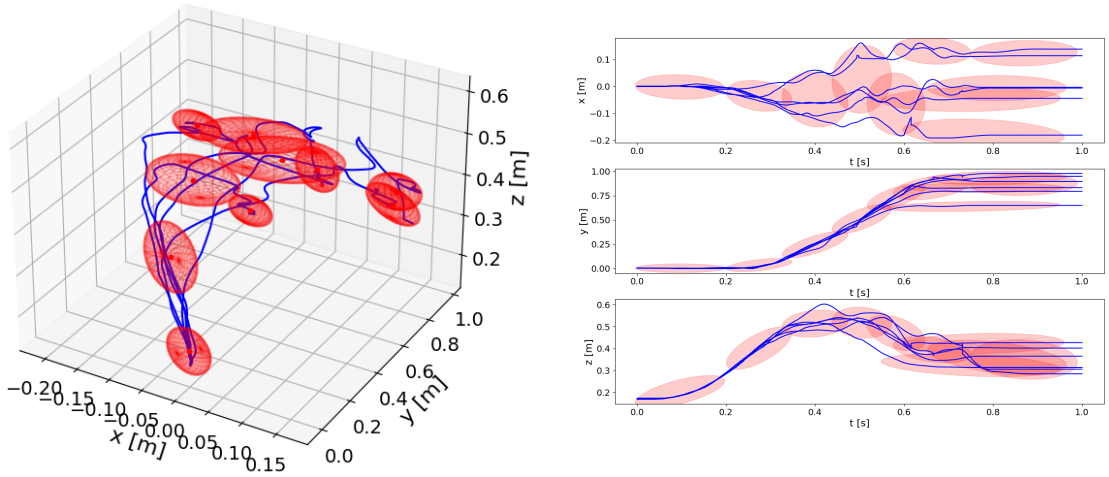
$$\mathbb{E}(\vec{\xi}(t^*)) = \ \vec{k}^*(K + \lambda\Sigma^{-1})\vec{\mu}. \tag{3.2}$$

This prediction is calculated with the help of the kernel matrix K which captures the similarity between data points. We use the Gaussian kernel

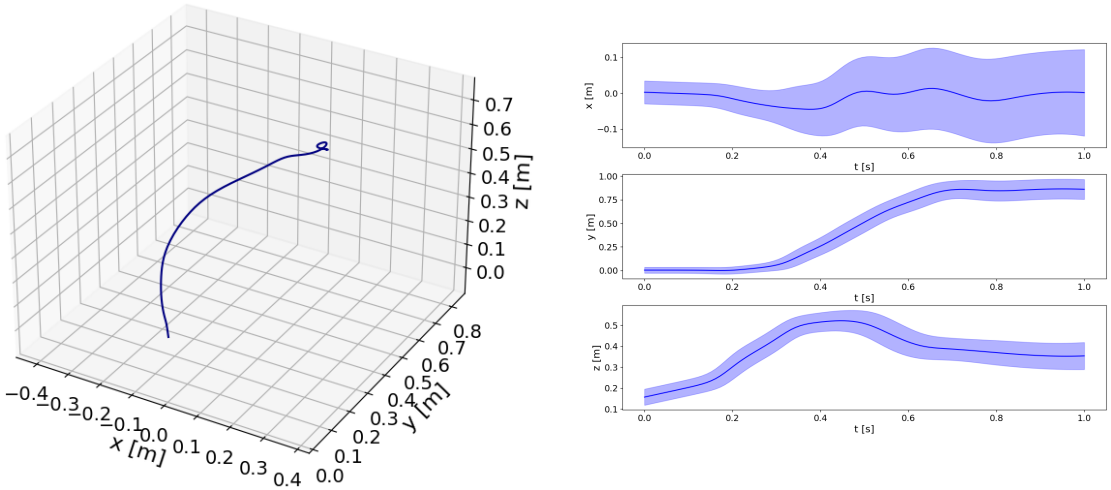
$$k(t_i, t_j) = \exp(-\ell(t_i - t_j)^2) \tag{3.3}$$

with length scale hyperparameter $\ell > 0$ for this computation. Please refer to [HRSC19] for the derivation. The result is depicted in Figure 3.5c.

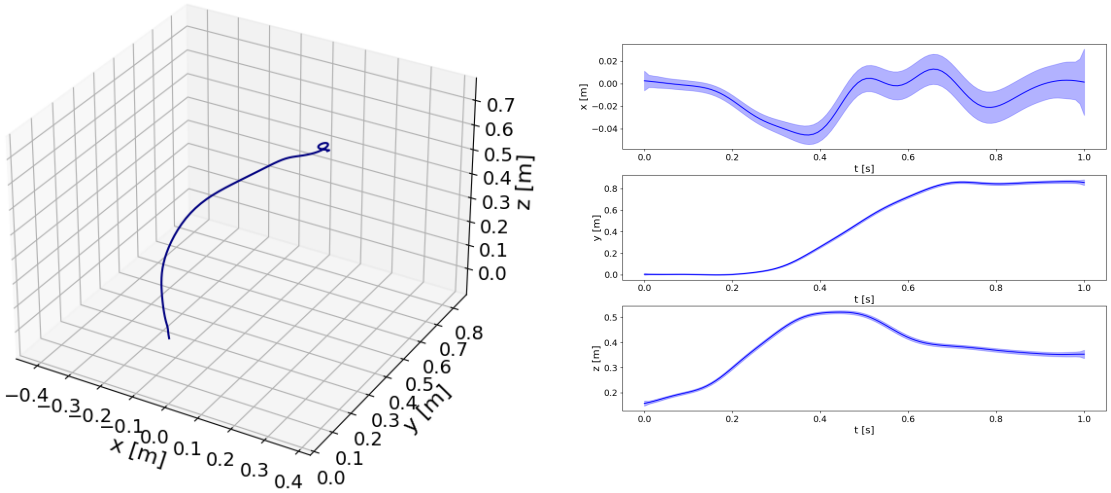
The final output of the KMP process is a series of Gaussian distributions, each representing the predicted pose (mean and covariance) for a specific time step. These distributions provide a refined probabilistic description of the local trajectory, capturing again both the expected poses and the associated uncertainties.



(a) Modeling demonstrations with GMM



(b) Probabilistic Reference Trajectory with GMR



(c) Probabilistic Trajectory with KMP

Figure 3.5: Running imitation learning algorithms on the demonstration data.

3.2.3 Deploying Learned Trajectory

There are two main variants for deploying the learned trajectory: directly running the learned trajectory and using the via point feature of KMP to adapt the trajectory in real-time.

KMP Trajectory: In the first variant, the learned trajectory runs on the robot as is. This approach achieves successful completion of the workstation cleanup task. However, it has two downsides:

- The robot performs the same trajectory relative to the object each time, regardless of the initial pose of the robot or the bin pose.
- Certain object poses may lead to unreachable configurations for the robot. This occurs when the object's position or orientation results in the robot needing to adopt poses that are outside its range of motion or cause collisions with the environment.

KMP Trajectory with Via Point: To address these limitations, we can use the via point feature of KMP. This approach allows us to add the current robot pose $\hat{\vec{\xi}}_s \sim \mathcal{N}(\hat{\vec{\mu}}'_s, \hat{\Sigma}'_s)$ (start) and the bin pose $\hat{\vec{\xi}}_e \sim \mathcal{N}(\hat{\vec{\mu}}'_e, \hat{\Sigma}'_e)$ (end) as via points. For both points, we set the covariance

$$\hat{\Sigma}' = \mathrm{diag}(\sigma) \tag{3.4}$$

where σ is a small value, e.g., $\sigma = 0.001$, ensuring that the KMP optimization respects the via points and therefore adapts the trajectory to different task instances. To implement this, we modulate the reference database $\mathbf{D}$, which becomes the desired database $\mathbf{D}'$

$$\mathbf{D}' = \left\{ \begin{array}{l} \left(\vec{t}_s, \hat{\vec{\mu}}'_s, \hat{\Sigma}'_s\right), \\ \left(\vec{t}_2, \hat{\vec{\mu}}_2, \hat{\Sigma}_2\right), \\ \vdots \\ \left(\vec{t}_{N-1}, \hat{\vec{\mu}}_{N-1}, \hat{\Sigma}_{N-1}\right), \\ \left(\vec{t}_e, \hat{\vec{\mu}}'_e, \hat{\Sigma}'_e\right) \end{array} \right\}, \tag{3.5}$$

from which we can generate a new and adapted motion with KMP. This effectively anchors the trajectory to the current robot pose, ensuring a smooth transition towards the object instead of moving to the default start of the vanilla KMP trajectory first. To achieve this, it proved necessary to use a larger length scale ℓ in the kernel function (Equation 3.3). The larger length scale makes the kernel decay slowly, which adds more weight to distant points in the prediction, ensuring that the influence of the start and end points extends more deeply into the trajectory. This results in a smoother transition from the start point through the trajectory points and from there to the end point, making it more adaptable to variations in the robot's start and end configurations while still maintaining accuracy

near the target object. This reduces the risk of encountering unreachable configurations. By incorporating these techniques, we ensure that the deployed trajectory is both robust and flexible, capable of handling a variety of object poses along with different start and end poses within the robot's workspace.

Figure 3.6 depicts the imitation learning component of Algorithm 1. After employing GMM/GMR/KMP on the demonstration data to retrieve a local probabilistic trajectory, this is saved in a file. In skill deployment, the trajectory is loaded, enhanced with via points and transformed into the robot base frame so the robot can perform the movement.

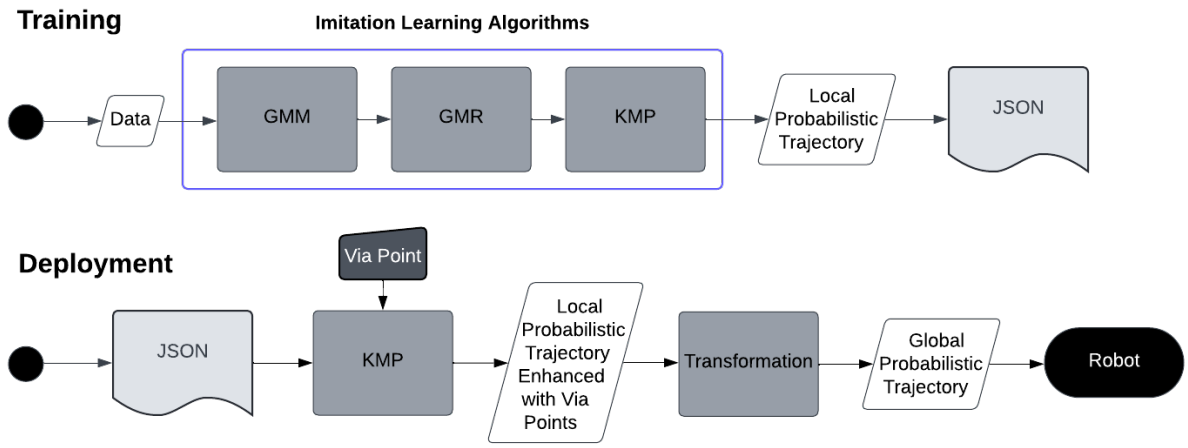


Figure 3.6: Imitation Learning Pipeline during Training and Deployment Stages

3.3 Computer Vision Pipeline

3.3.1 Camera Setup and Calibration

Setup: The Azure Kinect camera was selected for the computer vision component of this project due to its advanced depth sensing capabilities and comprehensive SDK support, which facilitate precise and efficient object detection and pose estimation. As shown in Figure 3.7, the camera is positioned on the left side of the station at an angle of approximately 30 degrees. This setup allows us to have a consistent and unobstructed view of the workstation, independent of the current configuration of the robot or the object location on the workstation. Additionally, it facilitates the integration of this external camera view with the tool center point (TCP) camera used in other applications, enhancing the overall system's flexibility and robustness.

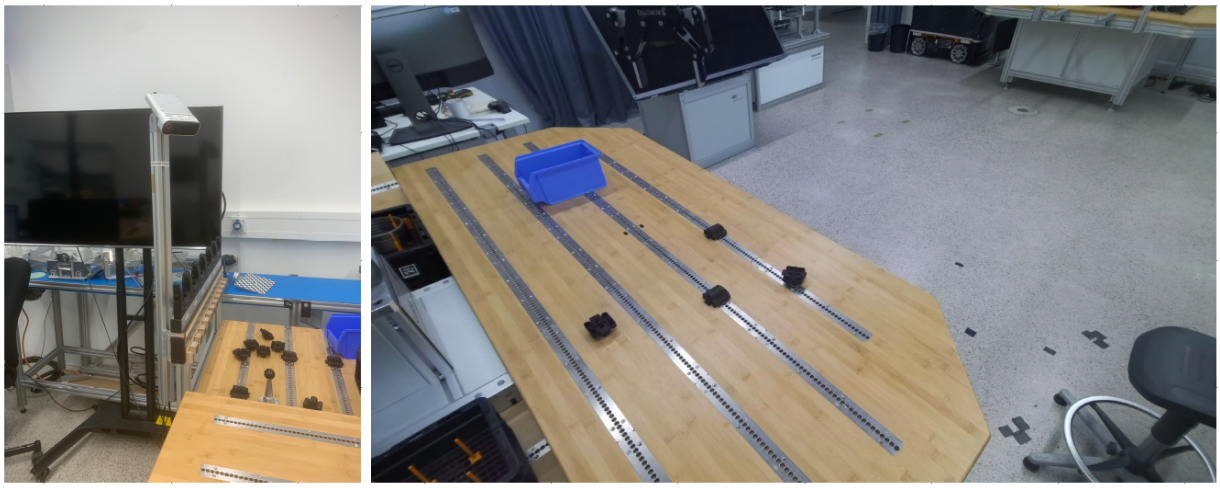
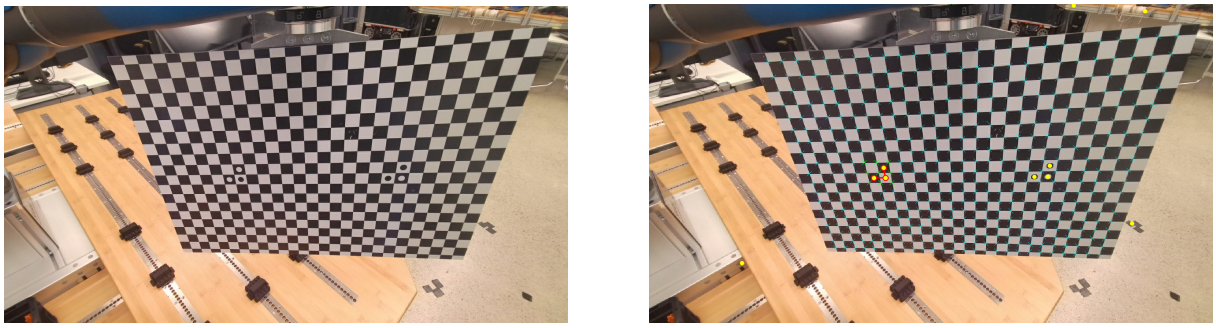


Figure 3.7: Camera setup and view of the workstation

Calibration: The calibration of the camera involves determining both intrinsic and extrinsic parameters to accurately map the camera's view to the robot's coordinate system. The camera is calibrated by obtaining a series of TCP poses in the robot base frame and capturing corresponding images while holding a chessboard calibration pattern as the end effector (Figure 3.8). The chessboard pattern is used to determine the intrinsic parameters, which include the focal length and optical center of the camera, and the extrinsic parameters, which describe the position and orientation of the camera relative to the robot base frame. Internal DLR calibration tools were employed for this process. Ensuring a variety of poses in which the chessboard is clearly visible in the camera is crucial for accurate calibration. This required careful planning to avoid transitions that might cause the chessboard to collide with the workstation table. The successful calibration provides us with the intrinsic matrix K and the homogeneous transformation matrix ${}^{SARA}T_{\text{CAM}}$ from the camera coordinate system to the SARA robot base coordinate system.



(a) Chessboard calibration pattern

(b) Corner detection for calibration

Figure 3.8: Camera Calibration

3.3.2 Image Dataset Generation

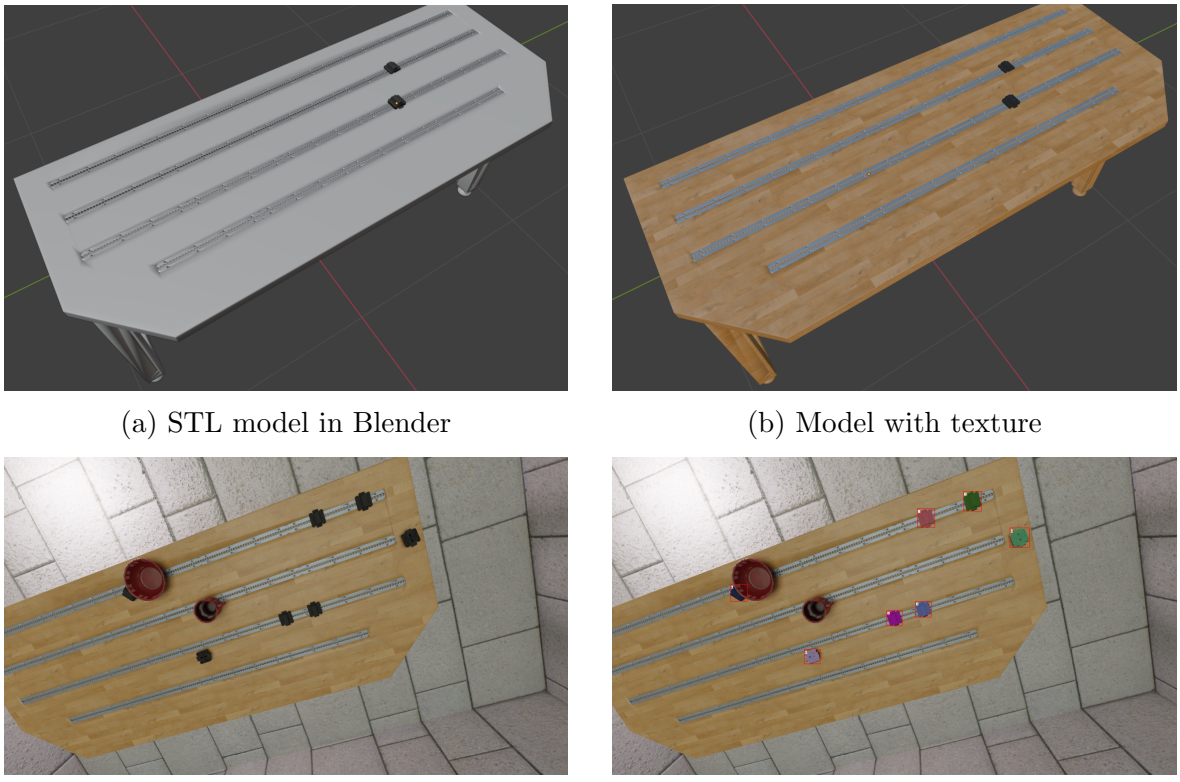
The starting point for generating the image dataset are the .stl models of the grid clamp object and the SARA workstation (Figure 3.10a). The first step involved visualizing the task in Blender, a powerful open-source software used for 3D modeling, rendering, and animation. Blender allows for the creation of highly detailed and realistic scenes. In Blender, texture was added to the models by including the materials depicted in Figure 3.9, which match the wood and grid clamp appearance, making the scene representation more realistic (Figure 3.10b).

To generate the image dataset, the BlenderProc framework was used, which facilitates the procedural generation of 3D scenes and datasets. Using BlenderProc, multiple scenes of grid clamps placed on the SARA workstation are created (Figure 3.10c). The total number of grid clamps and their placement, i.e. on or off the rails, is varied across a total of 1600 rendered scenes. For each scene, 25 images are captured, resulting in a dataset of 40 000 images. The rendering process is carried out on the GPU cluster at DLR, which provided the necessary computational resources. The server used features 2 x 32-core AMD CPUs (2.9Ghz, 1 T/C), 1024GB RAM, 10 GigE, and 8x NVIDIA GeForce RTX 3090 (Ampere) 24GB GPUs.

BlenderProc's Write Utility is used to automatically annotate the images with bounding boxes around the target objects: the grid clamps (Figure 3.10d).Additionally, unannotated distractor objects are included in the scenes to simulate a more realistic and challenging environment. The camera angle and radius are varied for each image relative to the point of interest, calculated as the mean pose of the grid clamps, to ensure most of the on-scene grid clamps are visible in each image. This variety in camera perspectives helps improve the robustness and generalizability of the to-be-trained detector model.

(a) Table Wood Texture (b) Rail Metal Texture

Figure 3.9: SARA workstation textures



(a) STL model in Blender (b) Model with texture

(c) Self-Organizing workstation scene (d) Annotated grid clamps

Figure 3.10: Image dataset generation for detector training

3.3.3 Vision Model Training

The generated grid clamp workstation images are used to train the You Only Look Once (YOLO) model [WBL23], a state-of-the-art object detector. YOLO works by dividing the input image into a grid and simultaneously predicting bounding boxes and class probabilities for each grid cell. This approach allows for real-time object detection with high accuracy, as YOLO processes the entire image in a single forward pass of the network. Please refer to [RDGF16] for a complete discussion.

The YOLO model learns to predict bounding boxes around the grid clamps along with the confidence scores for each detection. Training the YOLO model with the comprehensive and varied dataset ensures that the model can reliably detect grid clamps in different configurations and lighting conditions, making it a robust tool for the autonomous operation of the robot in its workstation.

Once the YOLO model is trained, it can be used to detect grid clamps in new images by outputting the bounding boxes and corresponding confidence scores. From these bounding boxes, the pose of the object can be estimated. Two different approaches were tested:

The first involves calculating the position of the grid clamp based on the detected bounding box coordinates and the depth information. Given a bounding box $\begin{bmatrix} x^{\text{rel}}_{\text{max}} & y^{\text{rel}}_{\text{max}} & x^{\text{rel}}_{\text{min}} & y^{\text{rel}}_{\text{min}} \end{bmatrix}$ with corners defined relative to the image, the corresponding coordinates in the image plane are

$$x_{\text{max}} = x^{\text{rel}}_{\text{max}} \times \text{image\_width} \tag{3.6}$$
$$y_{\text{max}} = y^{\text{rel}}_{\text{max}} \times \text{image\_height} \tag{3.7}$$
$$x_{\text{min}} = x^{\text{rel}}_{\text{min}} \times \text{image\_width} \tag{3.8}$$
$$y_{\text{min}} = y^{\text{rel}}_{\text{min}} \times \text{image\_height} \tag{3.9}$$

and we use the center of the bounding box in the image plane

$$x_{\text{c}} = \left\lfloor \frac{x_{\text{max}} + x_{\text{min}}}{2} \right\rfloor \tag{3.10}$$
$$y_{\text{c}} = \left\lfloor \frac{y_{\text{max}} + y_{\text{min}}}{2} \right\rfloor \tag{3.11}$$

for the position estimation. Using the Kinect's depth map we get the depth at the center point

$$z_c = \text{depth\_map}[y_{\text{c}}, x_{\text{c}}]. \tag{3.12}$$

The 2D homogeneous coordinates of the center point are

$$\vec{p}_h = \begin{bmatrix} x_{\text{c}} \\ y_{\text{c}} \\ 1 \end{bmatrix} \tag{3.13}$$

and we can map them to the 3D camera frame using the intrinsic camera matrix K:

$$\vec{p}_c = \left(K^{-1}\vec{p}_h\right) z_c. \tag{3.14}$$

Finally we project the object's 3D location in the camera frame to the SARA base frame with the camera to SARA base transformation matrix ${}^{SARA}T_{\text{CAM}}$:

$$\vec{p}_w = {}^{SARA}T_{\text{CAM}}\,\vec{p}_c. \tag{3.15}$$

The second option consists of training a dense pose estimator as in [UDS+23], which also estimates the orientation in addition to the position. Due to the unavailability of the estimator, this option could not be utilized.

3.3.4 Vision System Integration

Here, we provide an overview of each subsystem and their roles in the vision pipeline. This integration is implemented using the Portable Computer Vision Pipeline (PCVP) framework utilized at DLR. This framework consists of a manager which manages the lifecycle of modules, which communicate via specific interfaces. Figure 3.11 shows the implemented PCVP modules and the transferred data through the pipeline.

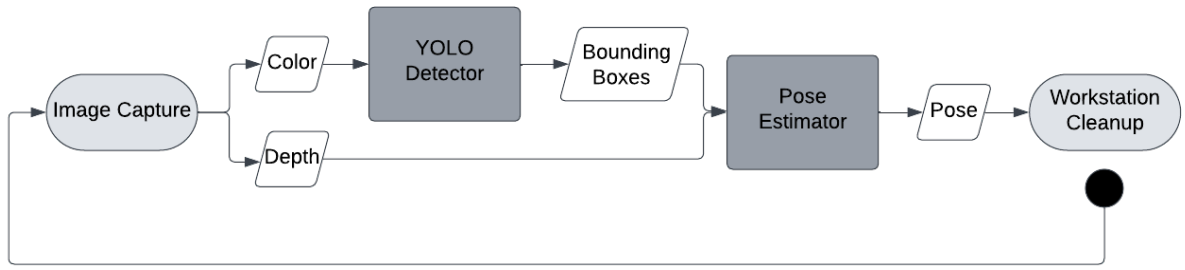


Figure 3.11: Computer Vision Pipeline consisting of PCVP Modules

Image Capture: The Image Capture module is responsible for obtaining images from the Azure Kinect camera. It captures both color and depth images.

YOLO Detector: The YOLO Detector module processes the color image captured by the Image Capture module. Using the trained YOLO model, this module detects grid clamps in the image and outputs bounding boxes along with confidence scores for each detected clamp. These bounding boxes indicate the location of the grid clamps within the image, and the confidence scores represent the likelihood that the detections are accurate.

Pose Estimator: The Pose Estimator module takes the bounding boxes produced by the YOLO Detector and the depth image from the Image Capture module to estimate the pose of each detected grid clamp. As shown in the previous section, this involves calculating the 3D position and orientation of the grid clamps in the robot base frame. The estimation is based on the geometric properties derived from the depth data and the bounding box coordinates, providing the pose of each detected clamp.

Workstation Cleanup: The Workstation Cleanup module receives the object poses from the Pose Estimator and uses them to execute the KMP trajectory on the robot. This involves running a learned trajectory to grasp the grid clamps based on their estimated poses. The KMP trajectory ensures that the robot moves smoothly and accurately to the target objects, facilitating efficient and effective cleanup of the workstation.

By integrating these subsystems, the vision system provides a comprehensive solution for detecting and estimating the pose of objects on the robotic workstation. The use of the PCVP framework ensures that the system is modular, scalable and also adaptable to other applications.

Chapter 4

Results

In this chapter, we delve into the empirical outcomes of our experiments, highlighting the effectiveness of the vision model and the adaptability of the Kernelized Movement Primitives (KMP).

4.1 Self-Organising Robot Workstation

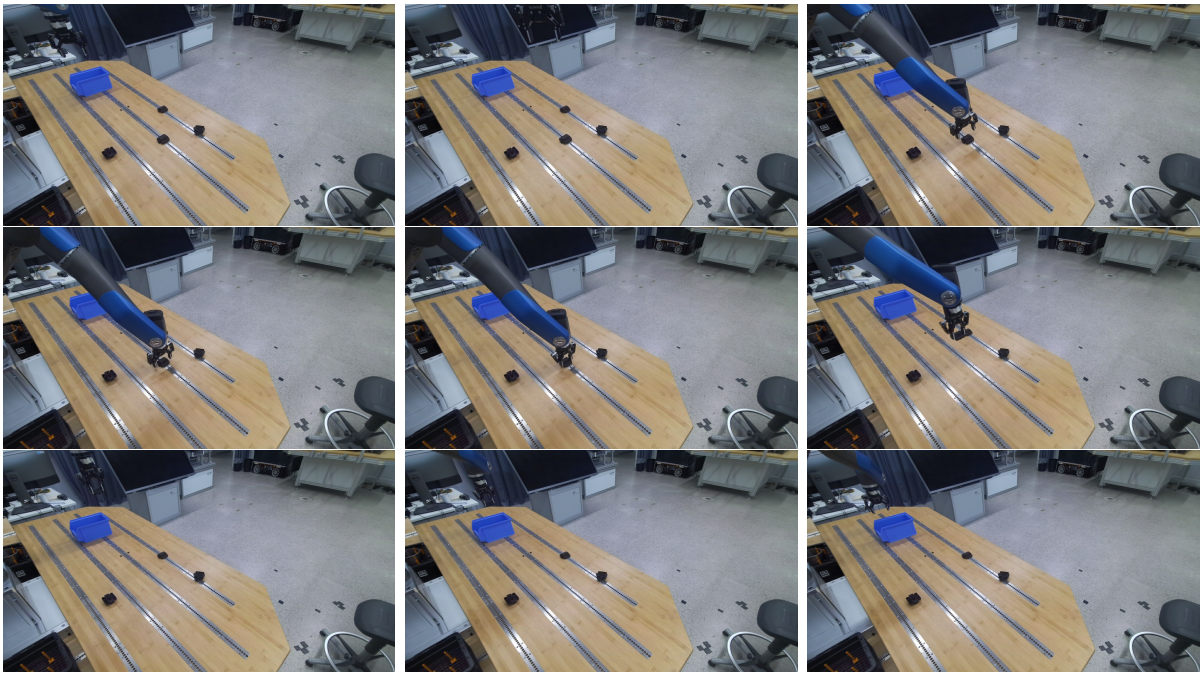


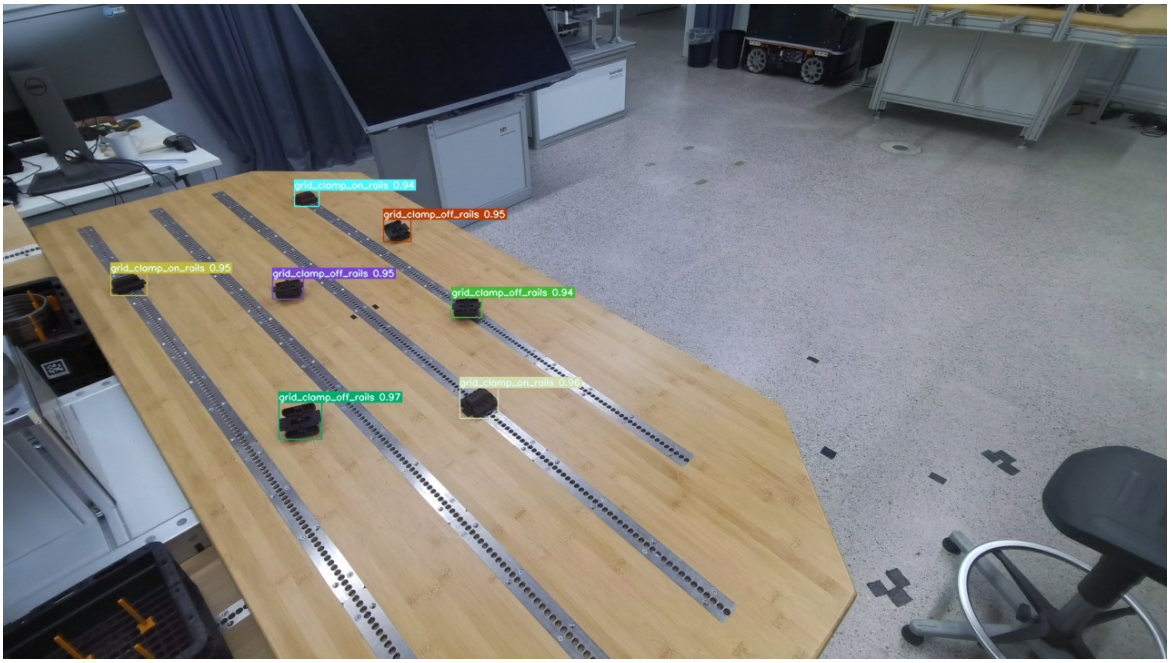
Figure 4.1: Self-Organising Robot Workstation Task

The skill of self-organising its workstation was learned and successfully deployed on the SARA robot (Figure 4.1). Utilizing the vision system, the position of the grid clamps is identified. With this information, SARA executes a smooth KMP trajectory, adjusting the path to start at the current position, to pick up the clamps. After grasping them, SARA follows another KMP trajectory, this time adapted to reach the position of the bin and place the clamps there. In the following sections, we will discuss the performance of both the vision pipeline and the imitation learning algorithms as well as necessary modifications that enabled this result.

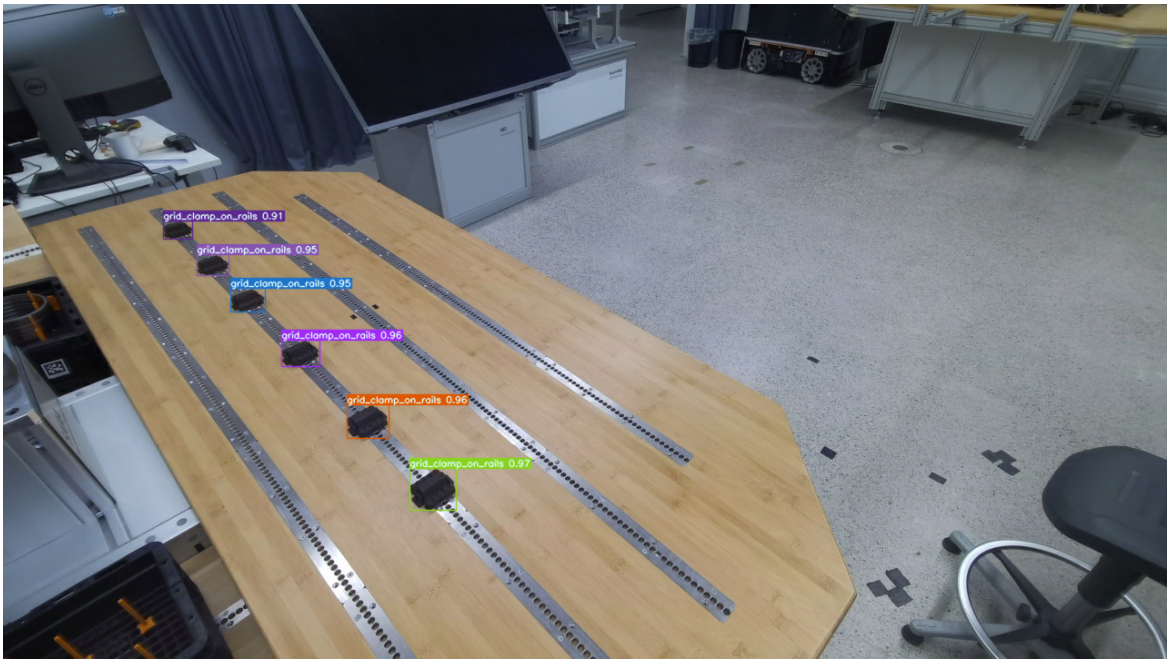
4.2 Vision Model Performance

We are able to make the following qualitative and quantitative observations about the quality of training results from the YOLO model and about the pose estimation:

- YOLO successfully detects grid clamps on the entire workstation and classifies them as being on or off the rails, as depicted in Figure 4.2a.
- The confidence level decreases with increasing distance from the camera, staying within the range 0.91-0.97 for grid clamps on the workstation table, as shown in Figure 4.2b.
- The confidence level exhibits very low variablity:
  - across runs
  - for different orientations in the same position
- The position estimation with Equation 3.14 operates with constant $x_{error} \approx 7$mm $y_{error} \approx 3$mm, which is enough to reliably grasp grid clamps. z can be set through model knowledge of the table surface height.
- YOLO fails to detect grid clamps placed on surfaces with significantly different textures from the table, for example grid clamps which are on top of other objects like a blue bin. This limitation is due to the training data consisting exclusively of images with grid clamps on the table surface. To enable this, training with a more diverse dataset would be needed to enhance the model's robustness to different backgrounds and textures.



(a) Reliable detection and classification



(b) Decreasing confidence with distance

Figure 4.2: YOLO grid clamp detection and classification

4.3 KMP Performance

4.3.1 Vanilla KMP

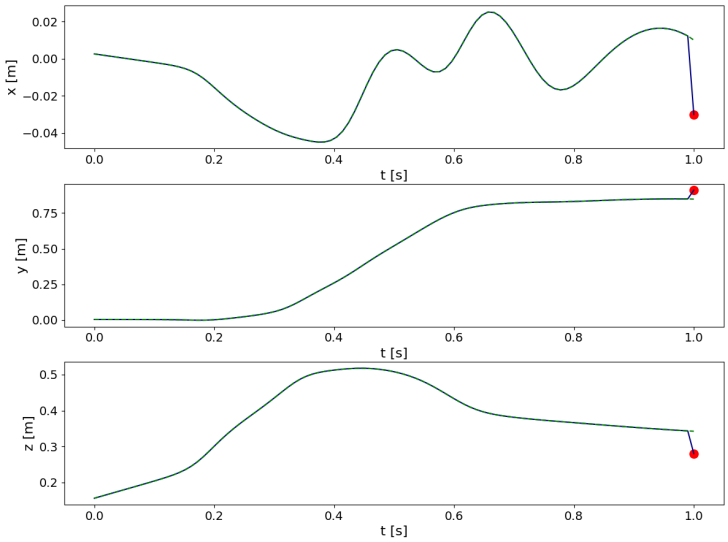
In the self-organising robot workstation use case, the robot receives the pose of the object from the vision system and performs a KMP trajectory to grasp the object and place it inside a bin. The vanilla KMP is able to generate a trajectory that reliably accomplishes this task under most conditions. However, for object poses located at the edges of the workstation, parts of the trajectory may lie outside the robot's configuration space, leading to singularities and failed grasps. This indicates that while vanilla KMP is effective in general, allbeit limited to one single learned movement, it struggles with extreme poses that require the robot to operate at the limits of its reach.

4.3.2 KMP with Via Points

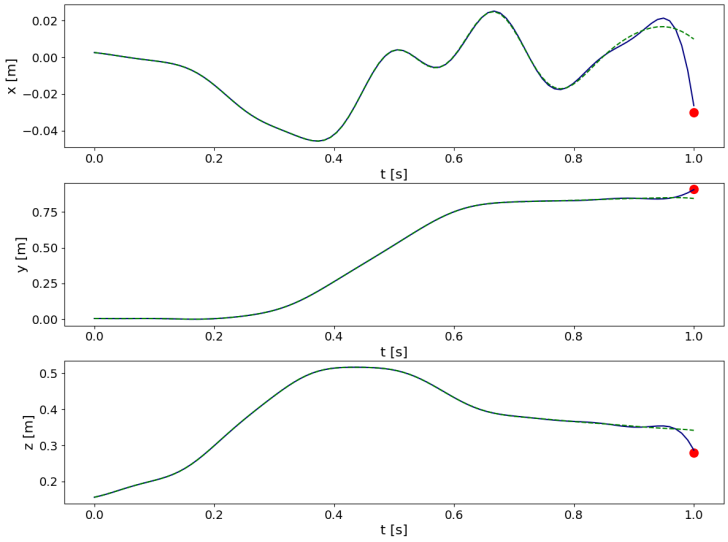
To address the limitations of the vanilla KMP, we introduce via points to the trajectory as described in [HRSC19]. By incorporating situation-specific start and end poses with low covariances, the trajectory modulation effectively reduces the risk of the robot defaulting to positions beyond its range of motion. This strategy improves both the naturalness and feasibility of the trajectory. Using via points, the trajectory adapts well to various start and end poses, even those at the edges of the workstation. This adaptation significantly reduces the occurrence of singularities and makes the robot's movements more reliable.

Following the discussion in Section 3.2.3, the choice of length scale ℓ for the kernel in Equation 3.3 is paramount to achieve smooth trajectory adaptation (Figure 4.3b). If it is too small, the adaptation towards it will happen too late and the change will be abrupt (Figure 4.3a). If it is chosen too large, then KMP might fail to produce a trajectory that actually hits that via point, even with the low covariance (Figure 4.3c). Additionally, the system performs well when the bin is located to the right of the grid clamps, as the demonstrations were conducted in this configuration. However, when the bin is placed on the left, the robot struggles to generate a suitable trajectory. This issue arises because the training data was biased towards right-side bin placements, indicating a need for more balanced training scenarios to improve generalization.

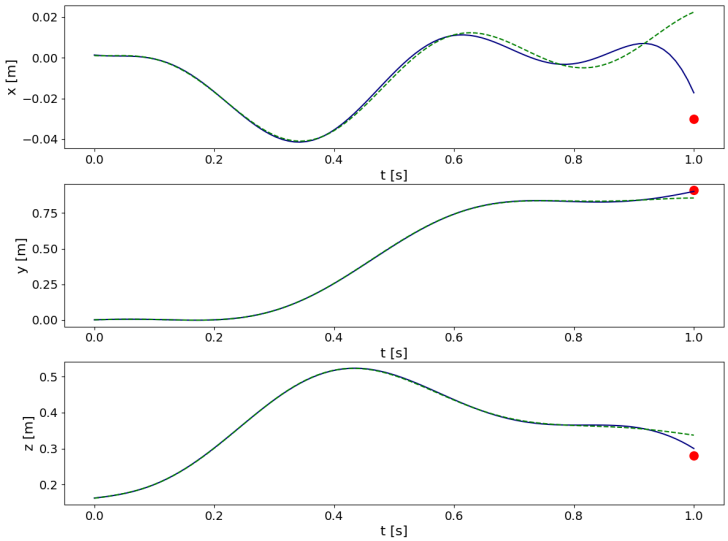
Overall, the vision model and the KMP implementation demonstrate strong performance in their respective tasks, leading to a reliable demo of the self-organising workstation. The vision model achieves high detection confidence and accurate pose estimation, while the KMP effectively generates trajectories for object grasping. The introduction of via points in KMP notably improves the adaptability of the robot's movements.



(a) KMP makes an abrupt adaptation to the end point: $\ell = 0.01$



(b) KMP trajectory is smooth towards the end point: $\ell = 0.1$



(c) KMP smoothness increases, but the end point is not reached: $\ell = 0.3$

Figure 4.3: Adapting the KMP trajectory to a new end point in placing task.

Chapter 5

Future Work

The results presented in this thesis enhance imitation learning approaches with computer vision for teaching robots a wide range of manipulation tasks in general and self-organising its workstation in particular. In this chapter we explore several avenues for future research and development which can further improve and expand upon the current system.

Uncertainty-influenced Impedance Control: There is unused potential in the deployment of KMP trajectories, since the current system does not make use of the uncertainty measure attached to each trajectory point as in [SHAD+19]. This can be used to set appropriate impedance control parameters during task execution that increase safety and reliability. For example, when the uncertainty for a given query point is high due to large variability in the demonstrations, then the stiffness coefficient can be set low so that the system is compliant, e.g. in case of collision. The stiffness can then be increased when the robot moves closer to the object, where the uncertainty is low. This enables higher precision when grasping the object.

Sensor Fusion and Visual Servoing: Another possible improvement is to fuse the Azure Kinect camera with the TCP camera on SARA for visual servoing. Initially, the Azure camera could provide the first detections and pose estimations of the grid clamps. Then, the TCP camera, positioned closer to the object, could be used for more precise estimations as the robot approaches the object. The assumption is that increasing proximity to the object will yield more accurate pose estimations, thereby enhancing the robot's ability to interact with the objects more effectively.

Reinforcement Learning: RL [KBP13] could be used to optimize the force interaction when grasping the grid clamps as demonstrated in [PQR+23]. Another potential extension is to use movement primitives as the action space in Reinforcement Learning (RL) problems. By leveraging the smooth and adaptable trajectories generated by KMP, an RL agent could explore the trajectory distributions and learn to optimize complex tasks

in dynamic environments. This approach could lead to more efficient learning processes and better task performance, as the movement primitives provide a structured and flexible action space for the agent to explore.

Constrained Imitation Learning: The current workstation cleanup problem could be elevated to include variable box sizes. This could be defined as a constrained imitation learning problem as in [HC20] and be solved with Linearly Constrained KMP.

Detect More Objects: Another area to look into is to expand the pool of detectable objects, so that the robot is able to clean its workstation regardless of what objects currently clutter it. Recent research into grasping unknown objects could be leveraged for this.

Natural Language Interfaces with LLMs and VLMs: Leveraging large language models (LLMs) and vision-language models (VLMs) as the first layer of the human-machine interface can significantly enhance the interaction between users and robots. By integrating LLMs, users can issue complex commands in natural language, such as instructing the robot to clean only specific objects like grid clamps on or off rails, or other designated items. VLMs enable the robot to understand and respond to these commands by analyzing visual input from its environment. This setup allows users to query the robot about its surroundings and receive detailed, context-aware responses in natural language. For instance, a user could ask, "Which objects are currently on the workstation?" and the robot would be able to identify and describe the objects it detects. Additionally, during the demonstration process, users can inform the robot about the specific object involved in the upcoming demonstration. This enhancement would provide both a more intuitive and accessible interface for task specification and environment awareness.

Expand the Framework: There is always room to add support for more modalities of user input and visualization when recording demonstrations. Incorporating diverse input methods such as voice commands, gesture recognition, and haptic feedback can make the system more intuitive and accessible to a broader range of users, including those with limited technical expertise or a disability. By enabling voice commands, operators can easily instruct the robot during the demonstration, making the process more seamless and efficient. Adding gesture recognition as an option can allow users to guide the robot's movements more naturally, mimicking the way humans teach each other. Additionally, haptic feedback can provide real-time sensory feedback, helping users to fine-tune the robot's actions with greater precision. Visualization tools are equally crucial in this expansion. Advanced visualization techniques, such as augmented reality (AR) overlays, can provide users with immediate visual feedback on the robot's understanding of the task. AR can highlight key points of interest, show the robot's predicted path, and visualize sensor data in a user-friendly manner. This not only aids in debugging and refining the demonstrations

but also enhances the overall user experience by making the process more interactive and engaging.

Furthermore, the framework should facilitate easy testing and comparison of different algorithms on the recorded pose and vision data. Implementing a more modular architecture in the learning subsystem, where different algorithms can be plugged in and tested without extensive reconfiguration would significantly streamline the experimentation process. Users could quickly switch between algorithms, evaluate their performance, and select the most suitable one for their specific application. This flexibility is crucial for adapting the system to various tasks and environments, ultimately broadening the scope of its applicability and fostering innovation.

Accessibility Study: Investigating the application of the developed system in various domains and industries could reveal new opportunities and challenges. It would be particularly interesting to study how the acceptance of robotic systems evolves across different professions when operators are empowered to program the robots themselves using the framework proposed in this thesis. The hypothesis is that this elevated interaction, where individuals can pass on a mastered skill to a robot intuitively through a user-friendly demonstration process, could significantly enhance acceptance. By enabling workers to train robots in this hands-on manner, the technology becomes more accessible and less intimidating. This approach could not only democratize the use of robotics but also foster a deeper understanding between humans and machines, as operators see their expertise directly reflected in the robot's actions. Consequently, this method of skill transfer could lead to higher adoption rates, increased efficiency, and greater innovation across various fields, as workers leverage their unique skills to taylor robotic performance to their work and environment.

Chapter 6

Conclusion

This thesis project presents the successful development and demonstration of an integrated system that combines probabilistic imitation learning with computer vision, applied to teach the DLR-SARA robot to autonomously organize its workstation. The system's key components include a demonstration recorder framework, robust vision models for object detection and pose estimation, and a trajectory learning pipeline that uses Kernelized Movement Primitives (KMP). The evaluation showed that the system accurately detects and grasps objects on the workstation, even in a varying environment, demonstrating the robustness and flexibility of the approach.

In accordance with the objectives set in Section 1.2, the main accomplishments include the creation of an intuitive demonstration recording process with a Graphical User Interface implemented with PyQt5; the setup and calibration of the Azure Kinect camera on the SARA workstation; the training of a YOLO model for reliable object detection; the implementation of 3D pose estimation from 2D bounding boxes; the implementation of a KMP-based learning pipeline that adapts to different task requirements through via point insertion; the combination of these sytems into a cohesive end-to-end pipeline; and the application of this pipeline to record demonstrations, learn the self-organising workstation task and evaluate the system's performance. These advancements make it easier for users to program robots through demonstration, significantly reducing the need for manual coding and allowing robots to acquire complex behaviors efficiently.

The benefits of this approach are manifold. It enhances the robot's ability to generalize from demonstrations to new situations, making the system adaptable and resilient. Additionally, the integration of computer vision allows for real-time environmental awareness, enabling dynamic adjustments to learned behaviors. This combination of techniques paves the way for more intuitive and accessible robotic systems, broadening their application and accessibility in various fields such as collaborative manufacturing, service robotics, and assistive technologies.

The completion of this thesis offers exciting opportunities for future work which could

explore the integration of advanced sensor fusion techniques, reinforcement learning for optimizing force interactions, and expanding the system's capabilities to detect and manipulate a wider variety of objects. Moreover, improvements could be made by incorporating additional user input modalities and visualization tools that further enhance the system's usability and make advanced robotic programming accessible to a broader audience.

List of Figures

Bibliography

[ACVB09] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 05 2009.

[BCDS08] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Survey: Robot Programming by Demonstration*, page 1371â1394. Springrer, 2008.

[CGB07] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[CZH+24] Chang Che, Haotian Zheng, Zengyi Huang, Wei Jiang, and Bo Liu. Intelligent robotic control system based on computer vision technology. *arXiv preprint arXiv:2404.01116*, 2024.

[GLW+21] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[HADSC20] Yanlong Huang, Fares J Abu-Dakka, João Silvério, and Darwin G Caldwell. Toward orientation learning and adaptation in cartesian space. *IEEE Transactions on Robotics*, 37(1):82–98, 2020.

[HC20] Yanlong Huang and Darwin G Caldwell. A linearly constrained nonparametric framework for imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4400–4406. IEEE, 2020.

[HRSC19] Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G Caldwell. Kernelized movement primitives. *The International Journal of Robotics Research*, 38(7):833–852, 2019.

[HSM96] Geir E Hovland, Pavan Sikka, and Brenan J McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *Proceedings of IEEE international conference on robotics and automation*, volume 3, pages 2706–2711. Ieee, 1996.

[IOE+20] Maged Iskandar, Christian Ott, Oliver Eiberger, Manuel Keppler, Alin Albu-Schäffer, and Alexander Dietrich. Joint-level control of the dlr lightweight

robot sara. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8903–8910. IEEE, 2020.

[KBP13] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[KI95] Sing Bing Kang and Katsushi Ikeuchi. A robot system that observes and replicates grasping tasks. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1093–1099. IEEE, 1995.

[LAL+24] Zemin Liu, Qingsong Ai, Haojie Liu, Wei Meng, and Quan Liu. Human-like trajectory planning based on postural synergistic kernelized movement primitives for robot-assisted rehabilitation. *IEEE Transactions on Human-Machine Systems*, 2024.

[LCGZ17] Clemente Lauretti, Francesca Cordella, Eugenio Guglielmelli, and Loredana Zollo. Learning by demonstration for planning activities of daily living in rehabilitation and assistive robotics. *IEEE Robotics and Automation Letters*, 2(3):1375–1382, 2017.

[MMMF24] Nikoleta Manakitsa, George S Maraslidis, Lazaros Moysis, and George F Fragulis. A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision. *Technologies*, 12(2):15, 2024.

[Mül07] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[PQR+23] Abhishek Padalkar, Gabriel Quere, Antonin Raffin, et al. A guided reinforcement learning approach using shared control templates for learning manipulation skills in the real world. PREPRINT (Version 1) available at Research Square, August 2023.

[RCC+16] Leonel Rozo, Sylvain Calinon, Darwin G Caldwell, Pablo Jimenez, and Carme Torras. Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527, 2016.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[RPCB20] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3(1):297–330, 2020.

[SHAD+19] João Silverio, Yanlong Huang, Fares J. Abu-Dakka, Leonel Rozo, and Darwin G. Caldwell. Uncertainty-aware imitation learning using kernelized move-

ment primitives. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, November 2019.

[TL96] SK Tso and KP Liu. Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. In *Proceedings of the IEEE International Conference on Industrial Technology (ICIT'96)*, pages 294–298. IEEE, 1996.

[UASvdS04] Holger Urbanek, A Albu-Schaffer, and Patrick van der Smagt. Learning from demonstration: repetitive movements for autonomous service robotics. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 4, pages 3495–3500. IEEE, 2004.

[UDS+23] Maximilian Ulmer, Maximilian Durner, Martin Sundermeyer, Manuel Stoiber, and Rudolph Triebel. 6d object pose estimation from approximate 3d models for orbital robotics, 2023.

[VF05] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pages 758–770. Springer, 2005.

[WBL23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[XWL+22] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. Pp-yoloe: An evolved version of yolo. *arXiv preprint arXiv:2203.16250*, 2022.

[ZHS+17] Martijn JA Zeestraten, Ioannis Havoutis, João Silvério, Sylvain Calinon, and Darwin G Caldwell. An approach for imitation learning on riemannian manifolds. *IEEE Robotics and Automation Letters*, 2(3):1240–1247, 2017.