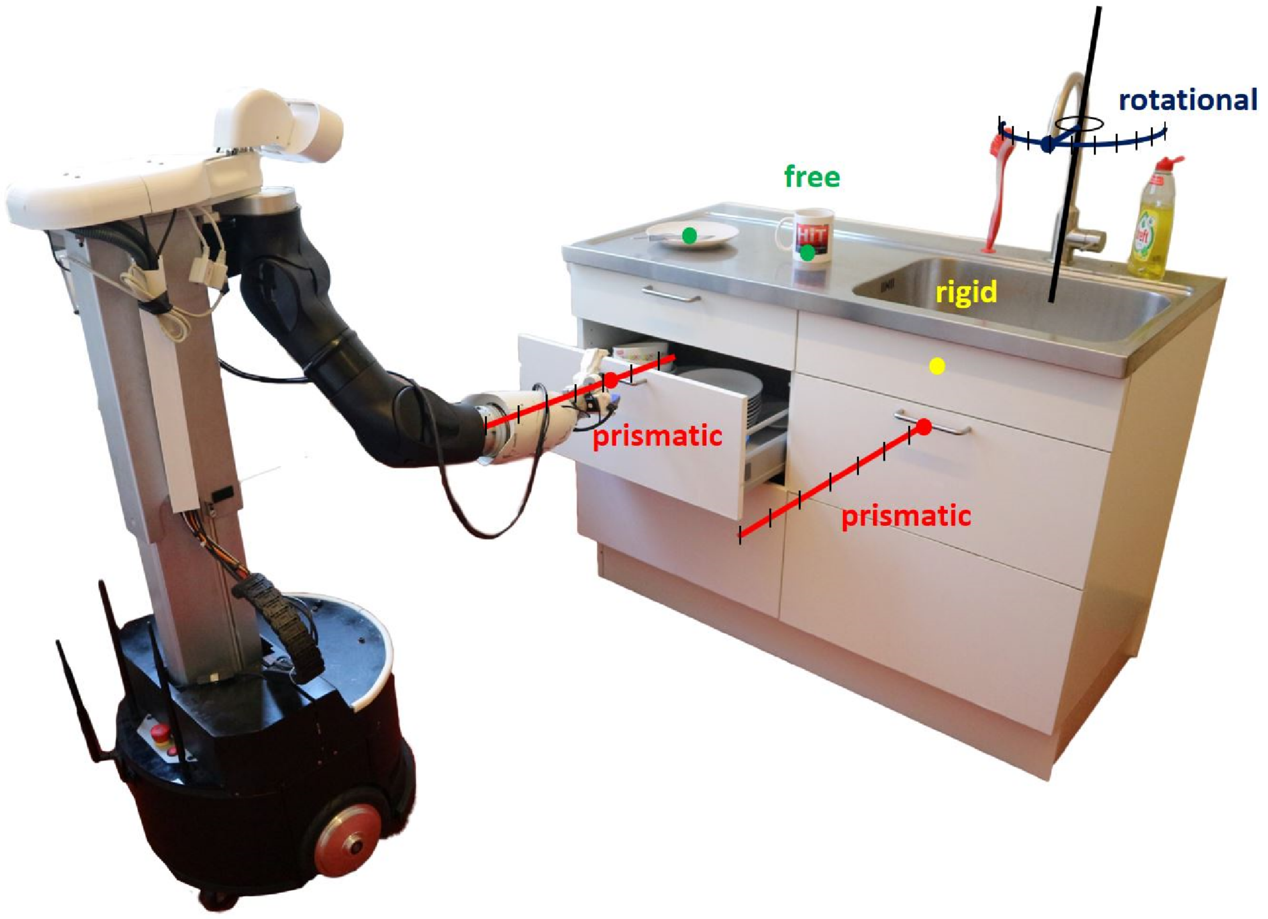
Learning kinematic models using a single tele-demonstration.

Mart Beeftink

Master of Science Thesis



Delft University of Technology

Cognitive Robotics

Learning kinematic models using a single tele-demonstration.

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Mart Beeftink

December 7, 2018

Supervisor(s):	Dr.ir. J. Kober	Department of Cognitive Robotics (CoR), TU Delft
	Dr.ir. E.A. Arkenbout	Heemskerk Innovative Technology B.V., the Netherlands
Reader(s):	Prof.dr.ir. J. Hellendoorn	Department of Cognitive Robotics (CoR), TU Delft
	Dr.ing. S. Grammatico	Department of Delft Center for Systems and Control (DCSC), TU Delft
	Dr.ir. C.E. Celemin Paez	Department of Cognitive Robotics (CoR), TU Delft

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Heemskerk Innovative Technology. Their cooperation is hereby gratefully acknowledged.

Abstract

To successfully perform manipulation tasks in an unknown environment, a robot must be able to learn the kinematic constraints of the objects within this environment. Over the years, many studies have investigated the possibility to learn the kinematic models of articulated objects using a Learning from Demonstration (LfD) [1] approach. In the majority of these studies the assumption is made that the robot solely manipulates articulated objects. In reality, however, robots also manipulate free-space objects that generally do not encounter any constraints. As a result, a human has to manually confirm which of the observed demonstrations concern articulated objects and which concern free-space objects. Furthermore, the majority of these studies do not evaluate the quality of the kinematic models prior to learning them. As a consequence, incorrect or uncertain models can be learned by the robot, which could lead to task failure or even dangerous behavior.

In this report, the novel Kinematic Model Learner (KML) framework is introduced, which aims to solve both of these problems using a multi-modal approach. In doing so, special attention is given to the understandability of the created framework, and its ability to adjust to different robot applications.

The KML framework consist of two separate frameworks called KML$_{traj}$ and KML$_{force}$. After the demonstration is given, the KML$_{force}$ framework first uses the force data to determine whether the manipulated object is free-space or constrained. If the object is recognized to be free-space, it will be classified as such after which the corresponding kinematic model is learned. If the object is classified as constrained, the KML$_{traj}$ framework uses the observed trajectory data to classify and learn the kinematic models of the constrained objects. In order to prevent the robot from learning incorrect or uncertain models, a probabilistic classifier is used which only learns a kinematic model if the corresponding confidence level is above a certain learning threshold.

The designed frameworks are experimentally validated by performing a total of 27 demonstrations on the care robot Marco using tele-operation. From these manipulations the trajectory and force data were used as inputs to validate each framework separately. Additionally, the KML$_{traj}$ framework is also evaluated using the Cody dataset, which contains the trajectories of 35 different manipulation tasks.

It has been concluded that the KML framework can robustly recognize and learn the kinematic models of the free-space and articulated objects. Moreover, a robustness analysis showed that the KML framework is more robust than the current state-of-the-art `articulation` package [2]. Additionally, the KML framework is able to asses the quality of the learned models and can prevent the robot from learning incorrect or uncertain models. Finally, the framework can be easily adjusted to different robot applications as the effects of the tuning parameters are easy to understand and can be determined by assessing the robot applications or by performing simple experiments.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Foremost, I would like to express my sincere gratitude towards my supervisors Jens Kober and Ewout Arkenbout for their enthusiasm, understanding, motivation and patience that helped me during my graduation. The many 'progress meetings' we had helped me enormously to keep focused and work towards the goal of finishing my thesis. Throughout my entire graduation period, I was kept on being surprised by the personal engagement, the enthusiasm and the willingness to schedule, yet another, meeting with me, that both of you showed.

Besides my supervisors, I would also like to thank my colleagues of Heemskerk Innovative Technology (HiT) for all the fun, educative and exciting moments that have made my stay very enjoyable. First of all, I would like to thank Cock Heemskerk for all his encouragement and endless positivity, and for giving me the opportunity to graduate at HiT and work with a real world robot application. Furthermore, I would like to thank my fellow graduates Karel and Jelle for our collaborations on a technical, but also personal level. Also a special thanks to my all-time favorite robot operator Dimitri for manually controlling all of the robot manipulation experiments.

Last but definitely not least, I would like to thank my friends, family and girlfriend for their support and encouragement along the way. Without them, none of this would have been possible.

Thank you all!

Delft, University of Technology
December 7, 2018

Mart Beeftink

Chapter 1

Introduction

1-1 Motivation

As a result of the increasing quality of public health, health care and nutritions, most countries in the world have rising life expectancy and an ageing population [4]. For this reason, it will be hard to preserve the quality of care, and thus people's well-being. Consequently, one of the European Union's grand challenges is to improve the life-long health and well-being of all, while maintaining economically sustainable care systems [5]. In order to solve this challenge the employment of new technologies is essential.

A promising new technology is the implementation of (semi-)autonomous care robots. Care robots are able to assist elderly and disabled people in their Activities of Daily Living (ADL), as illustrated in Figure 1-1. This aid can empower elderly to remain active and independent. Doing so will contribute to increasing, and lengthening the duration of, their physical, social, and mental well-being [5].

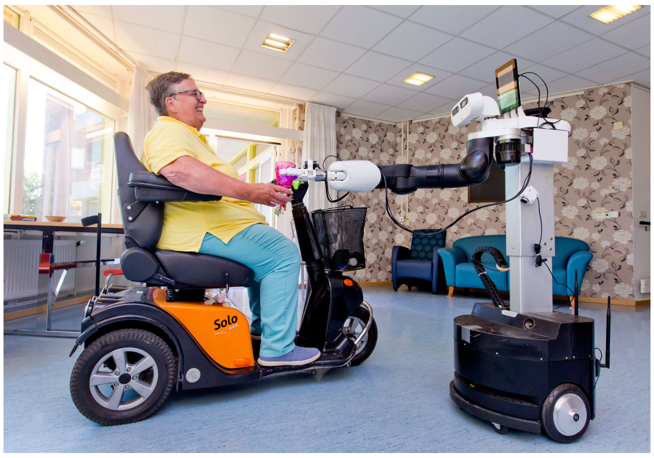


Figure 1-1: Care robot Marco fetches a cup. Courtesy of Heemskerk Innovative Technology (HiT).

To fulfill typical ADL tasks, care robots have to be able to manipulate a large variety of day-to-day objects [6]. In order to successfully manipulate an object autonomously, it is essential for the robot to first understand the spatial movements that are possible for that specific object. Some objects are partially constrained, such as doors, cupboards and drawers, while others can be moved freely through space or are immovable. Many of the objects found within a typical household settings tend to exhibit common kinematic structures, which makes it possible to learn these structures. Based on the resulting kinematic model and joint configurations, the robot is able to select and adapt actions, recognize their successful completion and detect failure [7].

This Master's thesis will focus on the problem of recognizing and learning the kinematic models of day-to-day objects. As an illustrating example, consider Figure 1-2 where a mobile manipulation robot interacts with various day-to-day objects in a kitchen environment, and learns their kinematic model.

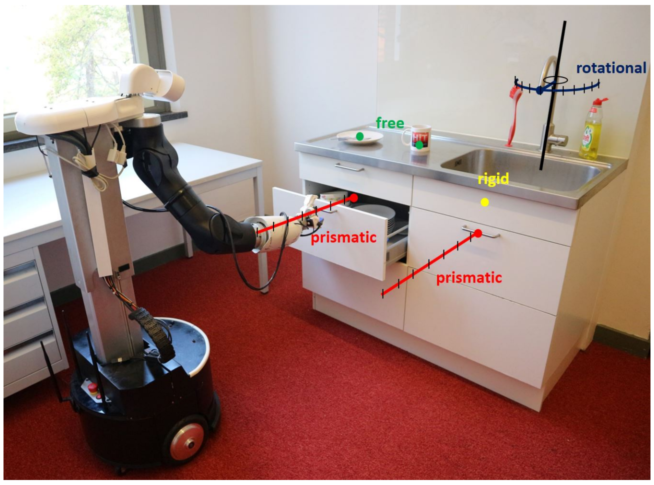


Figure 1-2: Care robot Marco interacts with various day-to-day objects in a kitchen environment, and learns their kinematic model.

1-2 Prior Work: Learning kinematic models

Over the last two decades, researchers have developed a number of robotic systems to open cabinets, doors and drawers [8, 9, 10]. A downside of these robotic system is that the robots are limited in their ability to adapt to previously unseen objects. For this reason, other work in service robotics focuses on using new observations, acquired by the robot, to estimate the kinematic parameters of previously unseen doors, drawers and rigid bodies [3, 11, 12, 13, 14]. This research domain is especially interesting as it enables the robot to learn new kinematic models during employment.

In this chapter a short summary will be given on the prior work of learning kinematic models. First, Section 1-2-1 gives a more formal definition of the five most common types of kinematic models of day-to-day objects. Subsequently, Section 1-2-2 explains how the kinematic models can be learned by making the robot interact with the object of interest. Finally, Section 1-2-3 describes the current state-of-the-art `articulation` package, which is able to determine the kinematic models of constrained objects using solely trajectory data.

1-2-1 Kinematic model types

When inspecting the set of objects that are typically manipulated by a service robot, it becomes evident that the movement constraints that act on these objects can be categorized into five different classes: rigid, prismatic, rotational, free-space, or complex. The different object classes are illustrated in Figure 1-3. The properties of the first four classes are very generic, thereby making it possible to describe them using four generic kinematic models. Sturm et al. [11] described these models using a parameter vector $\alpha \in \mathbb{R}^k$ that contains k parameters. All of the object classes have a different structure and are defined using a different number of parameters.

First of all, rigid objects (e.g., push-buttons or heavy furniture) are fully constrained and cannot be moved by the robot. The model that describes these objects is called $\mathcal{M}^{rigid}$ and has $k = 6$ parameters specifying the fixed position and orientation of the object.
Secondly, prismatic objects (e.g., drawers, sliding doors or blind windows) are constrained by prismatic joints causing the object to move in a straight line. The model that describes these objects is called $\mathcal{M}^{pris}$ and has 2 more parameters than the rigid model that describes the direction of the prismatic axis.[1]
Thirdly, rotational objects (e.g., doors, windows or dishwashers) can only rotate around the axis of rotation. The model that describes these objects is called $\mathcal{M}^{rot}$ and has 3 more parameters than the rigid model. The additional parameters describe the direction of the rotational axis (2 parameters) and its radius (1 parameter).
Fourthly, free-space objects (e.g., mugs, water bottles or apples), can be moved in any direction and therefore do not have any kinematic constraints. As a consequence, this kinematic model has 0 parameters.
Finally, the last object class contains constrained objects that have different mechanical linkages, such as a garage doors, or objects that consist of multiple constraints that can be manipulated simultaneously, such as two-bar office lamps. As the objects within this class have a more complex structure than the other classes, the model that describes these objects is called $\mathcal{M}^{complex}$. Complex objects can, for example, be modeled by splitting them into a combination of simpler models, or by fitting a Gaussian Process with varying number of parameters [11].

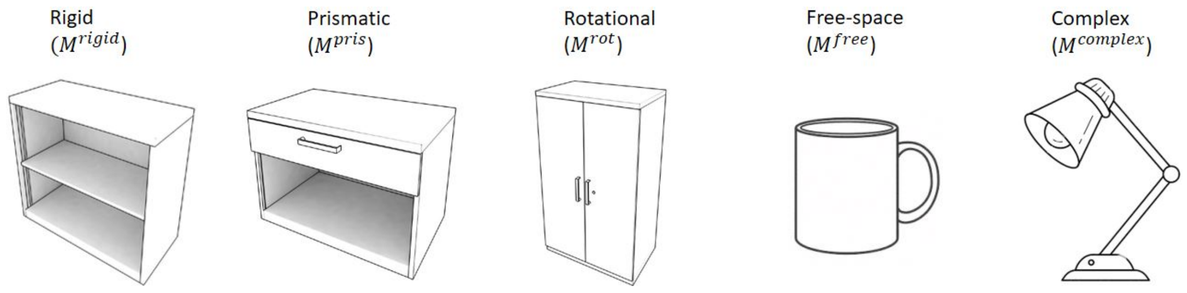


Figure 1-3: In a domestic environment there are 5 different classes of objects that are relevant for a robot manipulator application. All of the object classes have a different structure and are defined using a different number of parameters.

[1]Sturm et al. does not explain why only 2 additional parameters are required, but it is assumed that the direction of the prismatic axis is defined by 2 rotations around a predetermined axis.

This report focuses on recognizing free-space, prismatic and rotational objects and learning their kinematic models: $\mathcal{M}^{free}$, $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$. This thesis does not focus on learning models of the rigid or complex objects, as these objects are only very rarely manipulated by a service robot. In spite of this, in Section 2-3, it is shown that it is possible to recognize complex objects by means of elimination. For a more detailed description of the kinematic constraints or the classification and model learning of the $\mathcal{M}^{rigid}$ and $\mathcal{M}^{complex}$ models, the interested reader is referred to the work of Sturm et al. [11].

1-2-2 Interactive Perception

Kinematic models and other task-relevant information can be determined by carefully inspecting the interaction between the robot and the object. The field of research that studies the correlation between robot actions and changes in the sensor signals is called 'interactive perception'.

Traditionally, most interactive perception approaches have been based on one single sensor modality, mostly using vision [12, 14, 15, 16]. These uni-modal approaches fail when the used sensor stream does not contain the relevant information. According to Martín-Martín and Brock [3], these failures can be caused by changes in the environment (e.g., lights go off), in the characteristic of the interaction (e.g., self occlusions or grasp/contact loss), or in the perceptual task. Some researchers alleviate these failures by integrating multiple sensor modalities [12, 3].

Hausman et al. [12] introduced a vision based approach that combines the visual observations (obtained using fiducial markers), with the outcome of the robot's actions. More specifically, the outcome of the robot's manipulation actions are used to actively reduce the uncertainty of the candidate kinematic models and their parameters. This is done by fusing two different sensor modalities. The first sensor modality performs vision-based object tracking, similarly to Sturm et al. [11]. The second sensor modality determines whether an action is either successful (the expected motion is observed) or unsuccessful (the action cannot be completed because the joint friction is too high). The framework recursively selects those actions that are expected to be most informative and therefore quickly converges to the correct kinematic model.

Martín-Martín and Brock [3] created a multi-modal framework that combines force, trajectory and vision data in order to learn the kinematic and dynamic models of articulated objects. The created framework, as illustrated in Figure 1-4, is able to integrate and balance between different modalities according to their uncertainty.

The goal of most of these interactive perception approaches is to make a robot learn autonomously by making it interact with the unknown objects in its environment. The downside to this approach is, however, that the current state-of-the-art robot applications are cognitively incapable of dealing with the unpredictable nature of the objects within a domestic environment. As a consequence, the robot is unable to answer essential questions, such as: 'Which objects can/should I interact with?', 'How much force should I apply to move the object?' or 'How should I manipulate the object such that the task is successful?'.
Section 1-2-3 explains how Sturm et al. [11] circumvent this problem by using initial demonstrations (e.g., provided by a human tele-operator) as a starting point for learning kinematic

models. This simplifies the learning problem, since the robot can assume that the task is successfully completed during the demonstration, making it unnecessary to answer the questions posed earlier.

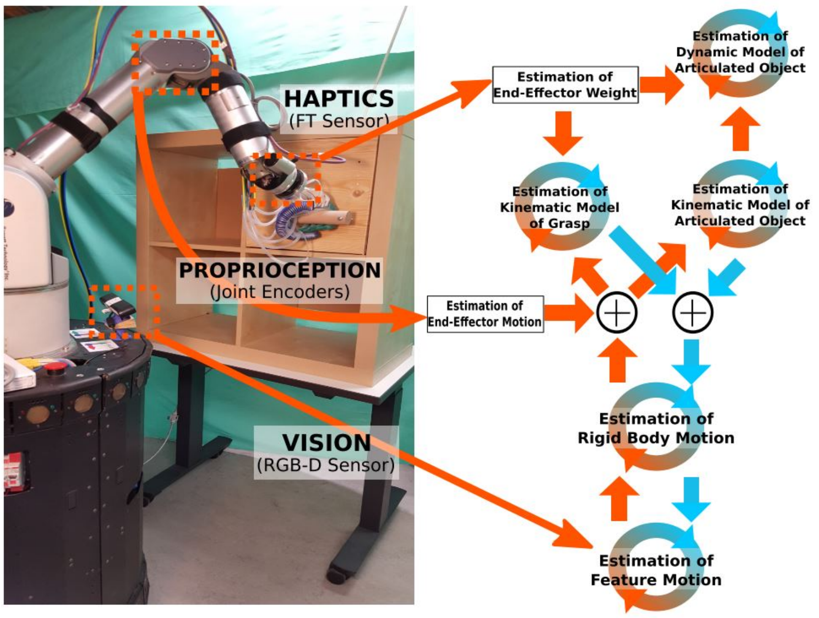

HAPTICS
(FT Sensor)
PROPRIOCEPTION
(Joint Encoders)
VISION
(RGB-D Sensor)
Estimation of End-Effector Weight
Estimation of Dynamic Model of Articulated Object
Estimation of Kinematic Model of Grasp
Estimation of Kinematic Model of Articulated Object
Estimation of End-Effector Motion
Estimation of Rigid Body Motion
Estimation of Feature Motion

Figure 1-4: The multi-modal setup used by Martín-Martín and Brock [3] to analyze the interaction between the robot and the object.

1-2-3 Using trajectory data

Sturm performed most of the pioneering work on learning kinematic models [2, 11, 17, 18, 19]. Based on this work, the state-of-the-art `articulation` package was created, which is able to determine the kinematic models of rigid, prismatic, rotational and complex objects using solely the trajectory data that was recorded by the robot during demonstrations. This package has been validated using multiple robot applications performing numerous manipulation tasks. After a demonstration is given to the robot, the best fitting kinematic model is found and learned in two steps.

First of all, the best fitting model for each object class is obtained by estimating the parameter vector $\alpha \in \mathbb{R}^k$ that maximizes that data likelihood:

$$\hat{\alpha} = \arg\max_{\alpha} p(\mathbf{x}|\mathcal{M}, \alpha), \tag{1-1}$$

where $p(\mathbf{x}|\mathcal{M}, \alpha)$ is the probability that the end-effector trajectory $\mathbf{x}$ would be observed, given the predicted model $\mathcal{M}$ and its corresponding parameter vector α. This probability is based on the distance between the observed trajectory and the predicted model, and can be tuned using the tuning parameter σ_z.[2]

[2]For a more detailed description of how the probability function $p(\mathbf{x}|\mathcal{M}, \alpha)$ is calculated, the reader is referred to [11].

Secondly, all of the candidate kinematic models are evaluated, after which the best fitting kinematic model is selected and learned by the robot. It is, however, difficult to select the correct model, as the models with a larger number of parameters have a fitting advantage over models with a smaller number of parameters. For example, a rotational model will often fit better than the prismatic model, as the former has one more parameter to fit. To compensate for the different number of tunable parameters, Sturm decided to use the Bayesian information criteria (BIC) [20], which is defined as:

$$BIC(\mathcal{M}) = -2 \log p(\mathbf{x}|\mathcal{M}, \hat{\alpha}) + k \log N, \tag{1-2}$$

where $\hat{\alpha}$ is the maximum-likelihood parameter vector, k is the number of parameters and N is the number of samples of the observed trajectory. The model selection is then reduced to selecting the model that has the lowest BIC score:

$$\hat{\mathcal{M}} = \arg\min_{\mathcal{M}} BIC(\mathcal{M}). \tag{1-3}$$

Finally, Sturm et al. also demonstrated how the learned kinematic models can be used by a robot to perform the task autonomously, using a Cartesian Equilibrium Point (CEP) controller [21]. For details on the equilibrium point control, and how it can be used to coordinate the motion of a mobile robot, the reader is referred to [22, 23].

1-3 Problem Description

This section describes the three main problems of the state-of-the-art approaches, as described in Section 1-2.

First of all, previous research has shown that the kinematic models of drawers and doors can be learned by the robot using vision, force or trajectory data. The majority of these studies assumed that the robot only manipulates articulated objects. In real world scenarios, however, service robots often operate free-space objects, for example when performing pick-and-place tasks. As a consequence, a human has to inform the robot whether it is dealing with an articulated or free-space object. Moreover, it is impossible for the trajectory-based approaches to reliably recognize free-space objects. This is because the free-space objects can be moved in any direction and can therefore easily describe the same trajectory as any other articulated object.

Secondly, it is often assumed that the kinematic models that are learned give a good representation of the manipulated object constraints. Consequently, most of the state-of-the-art frameworks do not take the quality of the learned models into account. In reality, however, robots often use different tools and tactics, such as hooks [22] or caging grasps [24] to manipulate objects. In many of these scenarios a rigid grasp cannot be guaranteed. As a result, the kinematic models have a chance of being of low quality as the observed trajectory can be a poor representation of the real constraints of the object. These low quality models generally cannot be used for autonomous manipulation, as this can result in unsafe movements and/or damage to the robot. Therefore, it is absolutely essential that a robot is able to evaluate the quality of the learned models.

Finally, another common problem for state-of-the-art frameworks is that they cannot be easily adjusted to serve other robot applications. As increasingly complex algorithms are used, the interpretation of the tuning parameters and the consequences of changing them can become intractable for robot engineers.

1-4 Goals

In order to solve the first problem as described in Section 1-3, the following main goal is defined:

> **Main goal:**
> Conceive a framework capable of recognizing free-space, prismatic and rotational objects based on a given (tele-)demonstration, and learn their kinematic models.

This goal translates into devising a framework that uses trajectory and force data of a given robot demonstration as input, and gives the learned kinematic model as output. As the framework should be applicable to other robot manipulators, it is also required to solve the second and third problem statement defined in Section 1-3. These problem statements result in the following two sub-goals:

> **Sub-goal 1:**
> The conceived framework should be able to assess the quality of the learned kinematic models.

> **Sub-goal 2:**
> The conceived framework should be understandable and easily adjustable to different robot manipulators.

1-5 Approach

In order to accomplish these goals, different approaches are used as described in this section.

> **Approach for main goal:**
> In order to recognize free-space, prismatic and rotational objects and learn their kinematic models, the novel Kinematic Model Learner (KML) framework is created (Chapter 2), which consists of two separate frameworks called KML$_{traj}$ and KML$_{force}$. The KML$_{traj}$ framework (Section 2-1) uses the trajectory data of the demonstration to find the best fitting prismatic and rotational models. However, as explained in Section 1-3, trajectory-based kinematic model learners are unable to reliably recognize free-space movements. To solve this problem, the KML$_{force}$ framework (Section 2-2) first uses the force data of the demonstrated manipulations to recognize free-space objects. Finally, both framework are combined into the KML framework (Section 2-3).

Approach for sub-goal 1:
To evaluate the quality of the learned kinematic models, a probabilistic classification method is created (Section 2-1-4) which determines the confidence levels of each of the learned models. The confidence levels are based on the difference between the candidate model and the observed trajectory. For safety reasons, only the kinematic models should be learned which have a high confidence level.

Approach for sub-goal 2:
In order to create an understandable framework that can be easily adjusted to other robot manipulators, comprehensible tuning parameters have to be used. These parameters can be either derived by assessing the robot goals and the quality of the robot sensors, or they can be directly observed through simple experiments. Furthermore, the kinematic models can be visually inspected and the confidence levels intuitively display the quality of the models.

Chapter 3 describes the experimental set-up and the 27 manipulation tasks that are used for the validation of the KML framework. Finally, based on these manipulations the results of the framework are discussed in Chapter 4.

Chapter 2

Kinematic Model Learner (KML) Framework

The current state-of-the-art `articulation` package [11] uses trajectory data to learn kinematic models, as described in Section 1-2-3. It was explained that there are three limiting factors of this framework. First of all, the `articulation` package cannot recognize free-space movements. Secondly, prismatic objects that move in a straight line can easily be recognized as rotational object with a very large radius. Finally, the package does not evaluate the quality of the learned models and the likelihood that the classification is correct. This chapter introduces the novel KML framework, which aims to solve all of these problems.

The novel KML framework is a combination of two separate frameworks that complement each other: KML$_{traj}$ and KML$_{force}$. The schematic overview of the KML framework is shown in Figure 2-1. This figure also serves as a roadmap for this chapter, in which each section is devoted to one part of the framework.

First, Section 2-1 describes how the KML$_{traj}$ framework uses the trajectory data of a manipulation to learn the best fitting prismatic and rotational model. Furthermore, it is explained how the KML$_{traj}$ framework evaluates the quality of the learned models and expresses this quality in a confidence level. If both the prismatic and rotational models fit poorly, the object can be classified as free-space/complex. Subsequently, Section 2-2 explains how the KML$_{force}$ framework uses force data to reliably recognize free-space movements. If the object is recognized to be free-space, it will be classified as such after which the corresponding kinematic model is learned. When the object is not recognized as free-space, it will be classified as a constrained object, which can either be prismatic, rotational or complex. Finally, in Section 2-3 the results of both frameworks are combined into the final KML framework. This framework is able to recognize and learn the kinematic models of free-space, prismatic and rotational objects. Furthermore, the framework is able to recognize complex objects by means of elimination. Throughout this chapter, it is also explained how the framework can be adapted to any robot manipulator application.

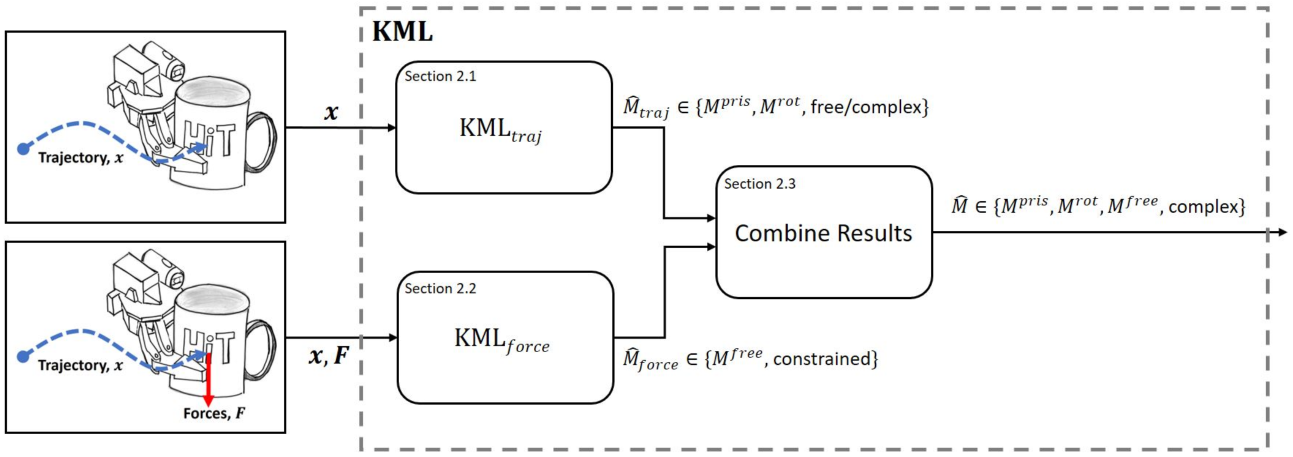

KML
Trajectory, x
x
Trajectory, x
Forces, F
x, F
Section 2.1
KML$_{traj}$
$\hat{M}_{traj} \in \{M^{pris}, M^{rot}$, free/complex$\}$
Section 2.2
KML$_{force}$
$\hat{M}_{force} \in \{M^{free}$, constrained$\}$
Section 2.3
Combine Results
$\hat{M} \in \{M^{pris}, M^{rot}, M^{free}$, complex$\}$

Figure 2-1: A schematic overview of the KML framework. This also serves as a roadmap for this chapter, in which each section is devoted to one part of the framework.

2-1 Part 1: Using trajectory data - KML$_{traj}$

This section describes the KML$_{traj}$ framework. The KML$_{traj}$ framework uses the trajectory data of a manipulation to learn the best fitting prismatic ($\mathcal{M}^{pris}$) and rotational model ($\mathcal{M}^{rot}$). If both the prismatic and rotational models fit poorly, the manipulation is classified as free-space/complex.

First, Section 2-1-1 describes the general principles of a least-squares approach. After this, Sections 2-1-2 and 2-1-3 explain how the best fitting prismatic and rotational models can be found. Finally, Section 2-1-4 explains how the quality of the fitted models can be evaluated using three probabilistic functions that are combined into a confidence level. In Chapter 4 the KML$_{traj}$ framework is validated using the experiments described in Chapter 3.

The key contributions of the KML$_{traj}$ framework:

- A novel approach is used to determine the best fitting rotational model $\mathcal{M}^{rot}$. Instead of penalizing the number of variables, as is done in `articulation` package, the KML$_{traj}$ framework penalizes small rotation angles (Section 2-1-3). This approach prevents the optimization function from finding $\mathcal{M}^{rot}$ with an undesirable large radius when dealing with a prismatic object.
- A confidence level (Section 2-1-4) is determined using the average distance between the observed trajectory and the predicted models. The robot can use this confidence level to exclusively learn models that are likely to be correct. Furthermore, the confidence level gives robot operators a better understanding of the quality of the learned models.

2-1-1 Model fitting using a least-squared method

Given the observed end-effector trajectory of the robot manipulation, the best fitting models ($\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$) can be determined using a least-squares approach [25]. In a least-squares approach, the goal is to find a solution which minimizes the squared residual error.

For robot manipulators, the error can be defined as the distance between an observed end-effector pose x_i and its nearest point on the predicted model $\hat{x}_i$. The nearest point can be calculated by projecting the observed pose onto the predicted model: $\hat{x}_i = \text{proj}_{\hat{\mathcal{M}}}(x_i)$. The least-squares solution then becomes:

$$\arg\min_{\hat{\mathcal{M}}} = \sum_{i=1}^{N} |\text{proj}_{\hat{\mathcal{M}}}(x_i) - x_i|^2. \tag{2-1}$$

In the KML$_{traj}$ framework, the least-squares approach is used to find the best fitting prismatic model $\mathcal{M}^{pris}$. However, this ordinary least-squares approach is not reliable when attempting to find the best fitting rotational model $\mathcal{M}^{rot}$. The unreliability is caused by the fact that a straight line can be approximated very well by a rotational model with an unrealistically large radius and thus a small rotation angle. Simply stated, a straight line may be very well approximated with $\mathcal{M}^{rot}$ given a sufficiently large radius and small rotation angle. As a result, it thus becomes hard to distinguish whether prismatic movements (e.g., opening a drawer) should be classified as $\mathcal{M}^{pris}$ or $\mathcal{M}^{rot}$. To cancel out this limitation, the KML$_{traj}$ framework amends this least-squares approach to determine the best fitting rotational models. In this approach, it is assumed that, when dealing with rotational objects, the robot is very likely to rotate this object more than a certain angle θ around its axis of rotation. It could, for example, be assumed that the robot will never open a cabinet only 20 or 30 degrees, as the contents of this cabinet cannot be reached by the robot when only partially open. In essence, the KML$_{traj}$ framework adds an additional cost function to the least-squares optimization problem which penalizes low rotation angles that are unlikely to be demonstrated. The final optimization function will be explained in more detail in Section 2-1-3.

2-1-2 Fitting prismatic models ($\mathcal{M}^{pris}$)

Given the observed end-effector trajectory of the robot manipulation, the best fitting prismatic model $\mathcal{M}^{pris}$ is determined using a least-squares approach. The least-squares problem, as described in Equation (2-2), finds a straight line in 3D which minimizes the squared residual error. Because the solution is a straight line, the problem can be specified as a linear least-squares problem. As a consequence, instead of using an optimization approach, the solution can be calculated algebraically. First, a point on the prismatic axis is found by taking the mean of the trajectory. Subsequently, the direction of this axis can be calculated as the first right-singular vector using Singular Value Decomposition (SVD) [26]. As an illustration, Figure 2-2 shows a simple 2D example of a prismatic movement and the fitted prismatic model.

$$\arg\min_{\mathcal{M}^{pris}} = \sum_{i=1}^{N} |\text{proj}_{\mathcal{M}^{pris}}(x_i) - x_i|^2. \tag{2-2}$$

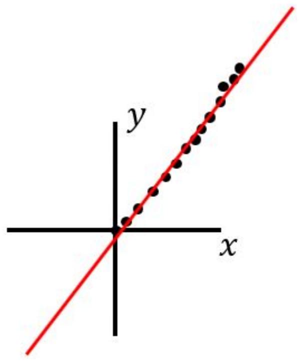


Figure 2-2: A simple 2D example of a prismatic movement. The best fitting prismatic model is represented by a 2D line (red), which minimizes the squared residual error.

2-1-3 Fitting rotational models ($\mathcal{M}^{rot}$)

Given the observed end-effector trajectory of the robot manipulation, the best fitting rotational model $\mathcal{M}^{rot}$ is determined in two steps.

> **Step 1:**
> Find a plane $\mathcal{P}$ that best fits the observed trajectory $\mathbf{x}$, and project $\mathbf{x}$ onto $\mathcal{P}$. This reduces the dimension of the optimization problem from $\mathbb{R}^3 \to \mathbb{R}^2$.

The best fitting plane $\mathcal{P}$ is found by solving the following least-squared problem:

$$\arg\min_{\mathcal{P}} = \sum_{i=1}^{N} |z_i - x_i|^2, \tag{2-3}$$

where z_i is the projection of x_i onto the plane, $z_i = \text{proj}_{\mathcal{P}}(x_i)$. The resulting plane $\mathcal{P}$ and projected trajectory $\mathbf{z}$ are used as inputs for step 2.

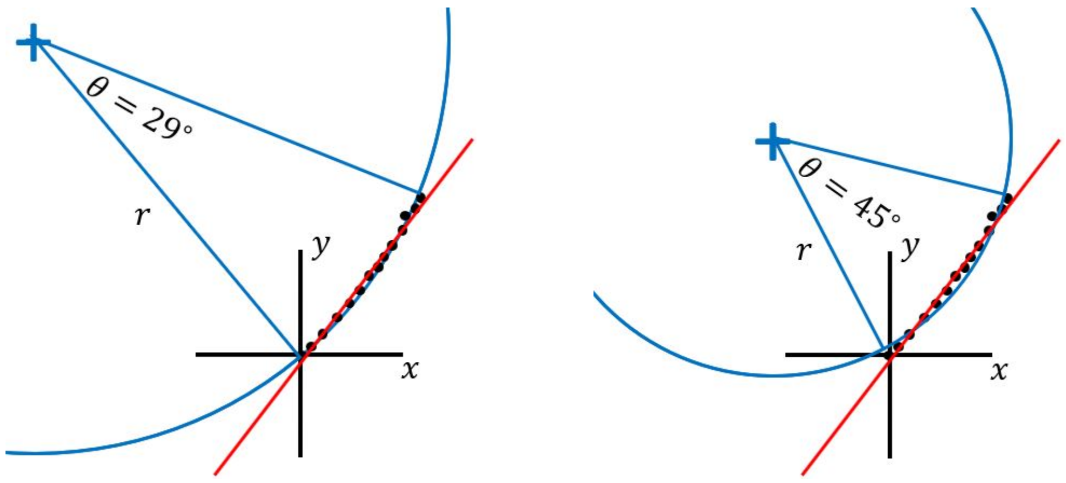
> **Step 2:**
> Find the best fitting rotational model by finding the circle on plane $\mathcal{P}$ which best fits the projected trajectory data $\mathbf{z}$.

The best fitting rotational model $\mathcal{M}^{rot}$ is obtained by finding a circle on plane $\mathcal{P}$ that best fits the projected trajectory $\mathbf{z}$. As described in Section 2-1-1, an ordinary least-squares approach should not be used to fit rotational models, as a prismatic movement can be approximated very accurately by a circle with an unrealistically large radius and concomitant small rotation angle. To prevent this from happening, an additional cost function $J(\theta)$ is added to the optimization problem. This cost function penalizes rotational models with small rotation angles, while leaving models with large rotation angles untouched. The final optimization function that finds the rotational models then becomes:

$$\arg\min_{\mathcal{M}^{rot}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |\text{proj}_{\mathcal{M}^{rot}}(z_i) - z_i|^2} + \lambda J(\theta), \tag{2-4}$$

where λ is a weighting factor that determines the overall influence of $J(\theta)$ during the optimization. The effect of the additional cost function is illustrated in Figure 2-3 using the same 2D example as in Figure 2-2. The next subsection provides a more detailed description of the used cost function and how it can be tuned to fit different robot application.

The optimization function, as formulated in Equation (2-4), has multiple local optima. Consequently, in order to find the global minimum, the used optimization algorithm should be carefully chosen.



(a) A least squares solution.

(b) The proposed optimization function with additional cost function $J(\theta)$.

Figure 2-3: A simple 2D example of a prismatic movement. In **(a)** the least-squares solution fits a rotational model (blue) with a large radius and small rotation angle. Both $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ fit equally well. In **(b)** the cost function $J(\theta)$ pushes the optimization function to search for rotation models with a larger rotation angle until θ_h is reached. Using the novel optimization function, it can be concluded that $\mathcal{M}^{pris}$ fits better.

Cost function $J(\theta)$

The cost function $J(\theta)$ is based on two assumptions about the robot's manipulations. On the one hand the robot is unlikely to rotate an object less than θ_l degrees, and on the other hand the robot is likely to rotate an object more than θ_h degrees. Based on these assumptions, the cost function is described as follows:

$$J(\theta) = \begin{cases} 1, & \text{if } \theta \leq \theta_l \\ q \arctan(-\theta + \frac{\theta_l + \theta_h}{2})\frac{1}{\pi} + 0.5 & \text{if } \theta_l < \theta < \theta_h \\ 0, & \text{if } \theta \geq \theta_h \end{cases} \tag{2-5}$$

where q is a scaling factor that prevents a gap between the three regions of the cost function. In the scenario that $\theta \leq \theta_l$ the cost is maximum, whereas for $\theta \geq \theta_h$ there is no additional cost. As a consequence, the cost function pushes the optimization solver to search for models where $\theta \geq \theta_h$.

The θ_l and θ_h values are tunable parameters that can easily be adjusted to the expected rotation angles within a given environment. As an example, in many robot applications, the robot hardly ever rotates an object less than $\theta_l = 25°$. Furthermore, for a task, such as opening a door, to succeed a rotation angle larger than $\theta_h = 45°$ is often required. The resulting cost function is shown in Figure 2-4. Of course it is also possible to use other cost functions (e.g., affine functions).

After the cost function has the desired shape, λ can be varied to either increase or decrease the influence of the cost function $J(\theta)$. In general, when increasing λ, the rotation angle of the found $\mathcal{M}^{rot}$ will increase until θ_h is reached, whereas lowering λ decreases the overall influence of the cost function. $\lambda = 0$ results in the original least-squares solution and should therefore not be used within the context of fitting a rotational model.

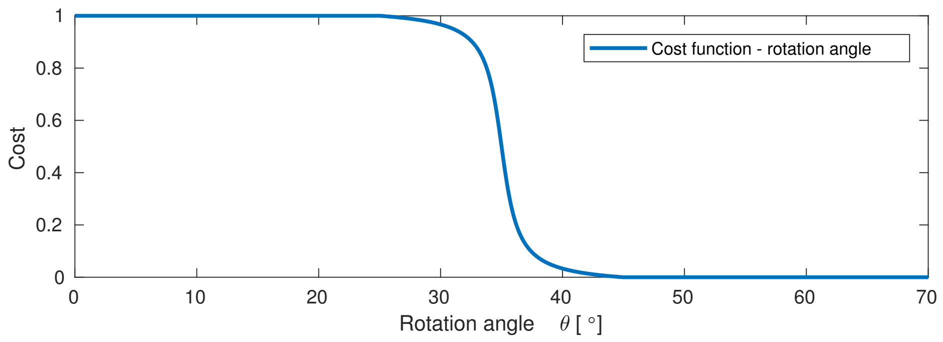


Figure 2-4: Cost function $J(\theta)$ that penalizes low rotation angles. The parameters in this example are $\theta_l = 25°$ and $\theta_h = 45°$.

2-1-4 Classification approach

Using the approaches explained in previous sections, the best fitting prismatic model $\mathcal{M}^{pris}$ and rotational model $\mathcal{M}^{rot}$ can be obtained. As explained in Section 1-3, it is very important that the robot exclusively learns the kinematic models that are likely to be correct. Therefore, a probabilistic classification method is developed that determines the confidence levels of each of the candidate models. This confidence level is determined using predefined probability functions that describe the likelihood that a candidate model is correct based on its Mean Absolute Error (MAE). The MAE, as defined in Equation (2-6), is used rather than the commonly used Root-Mean-Square Error (RMSE), as it is a more intuitive measurement and therefore simplifies the tuning of parameters [27].

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\text{proj}_{\hat{\mathcal{M}}}(x_i) - x_i| \tag{2-6}$$

The confidence level is based on a set of fuzzy rules describing four scenarios, as summarized in Table 2-1. In scenario 1, both $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ fit poorly, as their MAEs are high. By means of elimination, it can be concluded that the manipulated object is likely to be free-space/complex.[1] In scenarios 2 and 3, one model fits well (low MAE) while the other model does not fit well (high MAE). In these scenarios, the model with the lowest MAE is the most

[1]It is not possible to distinguish between the free-space and complex objects using only trajectory data. Therefore the object will be classified as free-space/complex. In Section 2-2 it is described how the KML$_{force}$ framework uses force data to make this distinction.

likely to be correct. In the final scenario both models do not fit perfectly (moderate MAE) but could be correct as the found error is not that high. In this scenario the robot should not classify the object or learn any model as it is uncertain which one is correct.

Table 2-1: Fuzzy rules that are used to determine the likelihood that a manipulated objects belongs to an object class.

Scenario	MAE^{pris}	MAE^{rot}	Most likely
1	High	High	$\mathcal{M}^{free/complex}$
2	Low	High	$\mathcal{M}^{pris}$
3	High	Low	$\mathcal{M}^{rot}$
4	Moderate	Moderate	$\mathcal{M}^{pris}$ or $\mathcal{M}^{rot}$

Probability functions

A probability function is used to describe the likelihood that an object belongs to a certain class. The probability function is expressed using a cumulative distribution function (CDF). The CDF can be written as $F_{\mu,\sigma}(\text{MAE})$ and is defined by $F_{\mu,\sigma}(\text{MAE}) = \Pr(X \leq \text{MAE})$, where X is a random variable. Thus, for a continues random variable the CDF becomes:

$$F_{\mu,\sigma}(\text{MAE}) = \int_{-\infty}^{\text{MAE}} f_{\mu,\sigma}(t)dt, \tag{2-7}$$

where $f_{\mu,\sigma}$ is the probability density function of a normal distribution $\mathcal{N}(\mu, \sigma^2)$.

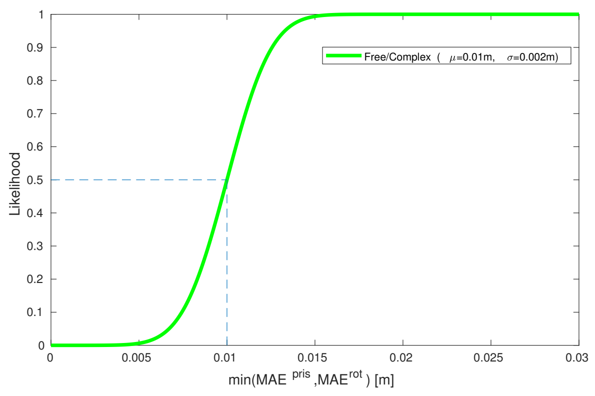
As shown in Table 2-1, the only scenario where the manipulated object should be classified as a free-space or complex movement, is when both models fit poorly. Therefore, the likelihood that the object is free-space or complex is calculated using the lowest MAE of both models:

$$p(\mathcal{M}^{free/complex}) = F_{\mu_{fc},\sigma_{fc}}\Big(\min(\text{MAE}^{pris}, \text{MAE}^{rot})\Big) \tag{2-8}$$

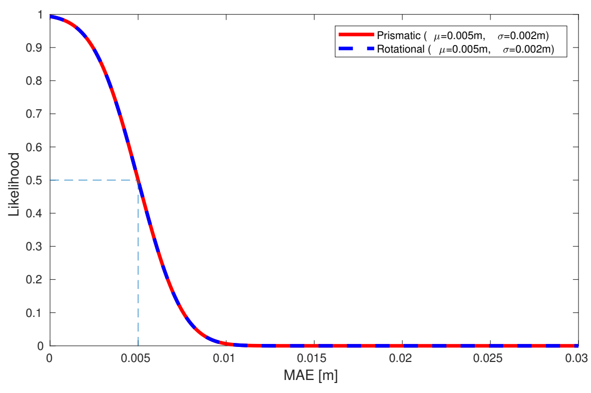
The likelihood that the object is either prismatic or rotational is calculated as:

$$\begin{aligned} p(\mathcal{M}^{pris}) &= 1 - F_{\mu_{pr},\sigma_{pr}}(\text{MAE}^{pris}) \\ p(\mathcal{M}^{rot}) &= 1 - F_{\mu_{pr},\sigma_{pr}}(\text{MAE}^{rot}) \end{aligned} \tag{2-9}$$

The shape of the probability functions can be easily tuned using the parameters μ and σ. The parameter μ describes the 50% likelihood point, whereas σ determines the steepness of the curve. Users can adjust these parameters to fit their robot application. For example, more accurate robot manipulators are advised to decrease μ and σ, as lower MAEs are expected during the manipulations. For our robot application the values $\mu_{fc} = 0.01\,\text{m}$, $\sigma_{fc} = 0.002\,\text{m}$, $\mu_{pr} = 0.005\,\text{m}$, and $\sigma_{pr} = 0.002\,\text{m}$ are chosen. The resulting probability functions are shown in Figure 2-5.



(a) Probability function $p(\mathcal{M}^{free/complex})$.



(b) Probability functions $p(\mathcal{M}^{pris})$ and $p(\mathcal{M}^{rot})$.

Figure 2-5: The probability functions that describe the likelihood that an object belongs to a certain object class. **(a)** shows the probability function for the free/complex class. **(b)** shows the probability functions for the prismatic and rotational class. The functions overlap as they are chosen to be identical.

Confidence levels (C)

A confidence level is used to describe the confidence that a manipulated object belongs to either the prismatic, rotational or free/complex object classes. The confidence levels of all candidate models should sum to 100 %. The confidence levels are calculated in two steps, using the previously designed probability functions. First of all, the confidence level of $\mathcal{M}^{free/complex}$ is calculated using:

$$C^{free/complex} = p(\mathcal{M}^{free/complex}|\mathbf{x}) \cdot 100\% \tag{2-10}$$

Secondly, the residual confidence level is allocated to the prismatic and rotational models using their relative likelihood given by:

$$\begin{aligned} C^{pris} &= \left(1 - p(\mathcal{M}^{free/complex}|\mathbf{x})\right) \frac{p(\mathcal{M}^{pris}|\mathbf{x})}{p(\mathcal{M}^{pris}|\mathbf{x}) + p(\mathcal{M}^{rot}|\mathbf{x})} \cdot 100\% \\ C^{rot} &= \left(1 - p(\mathcal{M}^{free/complex}|\mathbf{x})\right) \frac{p(\mathcal{M}^{rot}|\mathbf{x})}{p(\mathcal{M}^{pris}|\mathbf{x}) + p(\mathcal{M}^{rot}|\mathbf{x})} \cdot 100\% \end{aligned} \tag{2-11}$$

Additionally, an adjustable learning threshold is added, which allows robot engineers to decide how well the predicted model should fit before the robot learns a kinematic model. An object is only classified if the confidence level is above this learning threshold. If the object is classified as prismatic or rotational, the corresponding kinematic model is attached to the object and learned on the robot. If the KML$_{traj}$ framework classifies the object to be free-space/complex, the object class is still uncertain and no model can be learned. If the confidence level of all models are below the learning threshold, no model is learned. This classification approach is illustrated in Figure 2-6.

In a nutshell, the confidence level gives an indication about the quality of the prismatic and rotational models by assessing whether the difference between MAE^{pris} and MAE^{rot} is significant enough. Moreover, if both MAEs are high, it can be concluded that the manipulated object either belongs to the free-space or complex object class. The learning threshold makes sure that no incorrect or uncertain[2] models are learned.

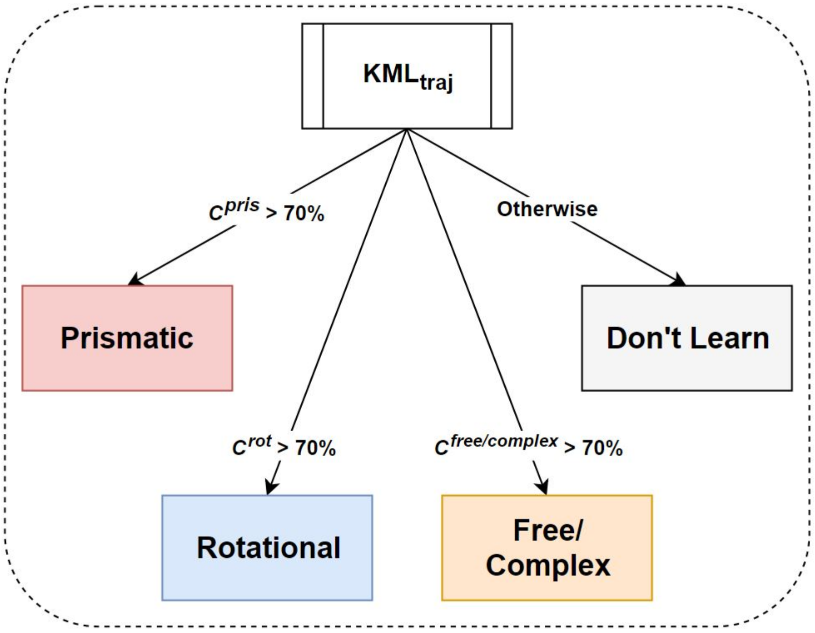


Figure 2-6: The KML$_{traj}$ classification scheme. The KML$_{traj}$ framework uses a probabilistic classifier, which only classifies an object if the confidence level is above a certain learning threshold (70% in this figure). If the object is classified as prismatic or rotational, the corresponding kinematic model is attached to the object and learned on the robot. If the KML$_{traj}$ framework classifies the object to be free-space/complex the object class is still uncertain and no model can be learned.

2-2 Part 2: Using force data - KML$_{force}$

As described in the previous section, the trajectory data cannot be used reliably to recognize free-space objects. This is mainly because free-space objects can be moved in any direction and therefore often move in a straight or circular fashion. To deal with this uncertainty, the KML$_{force}$ framework is designed such that it can recognize free-space movements using the available force data measured by the force/torque (f/t) sensor.

As the goal of this framework is to recognize free-space objects, it is only interesting to know the external forces on the f/t sensor caused by the manipulated object. Furthermore, in order to make these external forces independent of the end-effector configuration, the input forces used in the KML$_{force}$ framework are with respect to the world frame Ψ_W, as illustrated in Figure 3-1.

In Section 2-2-1 the raw force data is pre-processed such that only external forces caused by the objects are left. Subsequently, Section 2-2-2 describes the two characteristic properties of objects that move through free-space and determines the inputs for the classifier. Finally, Section 2-2-3 uses the characteristic free-space properties to create the decision boundaries of

[2]The correctness of a model is uncertain if there are multiple high-likelihood candidate models.

the binary classifier. This binary classifier is able to distinguish free-space movements from constrained movements using the force data that was recorded during a manipulation. In Chapter 4 the KML$_{force}$ framework is validated using the experiments described in Chapter 3.

Main contributions of the KML$_{force}$ framework:

- The KML$_{force}$ framework can distinguish free-space from constrained movements using the external forces that are caused by the manipulated object.
- Manipulations which only partially move through free-space can also be recognized by the framework as the manipulations are first split into smaller segments.

2-2-1 Pre-processing the force data

The forces of robotic manipulators are generally measured with a f/t sensor which is placed at the end of the wrist, right before the end-effector. Since the sensor is positioned at the wrist of the robot, the sensor measures not only the external forces that are acting on the end-effector, but also the forces that are caused by the weight and the motion of the end-effector. In order to obtain solely the external forces that are caused by the manipulated object, the forces caused by the end-effector should be removed computationally.

In addition, f/t sensors are occasionally disturbed by drift in their measurements. As a result of this drift, the sensor values change over time. A common approach to minimize the effect of drift is to compute the offset between the measurements and the true state of the sensor [28]. This can be done by measuring the forces of a stationary empty gripper and removing the gripper's own weight. In order to effectively minimize the effect of drift, the offset should be removed regularly.

In many robot applications, the noise of the f/t sensor is filtered using, for example, a low-pass filter. In Section 4-2-3 it is explained that both filtered and unfiltered data can be used as inputs for the KML$_{force}$ framework, as long as the classification boundaries are adjusted accordingly.

2-2-2 Inputs for the binary classifier

The external forces are caused by the inertial, centrifugal, Coriolis and gravitational forces of the object. However, as the velocity and acceleration of the robot's end-effector is generally low, it can be assumed that the inertial, centrifugal and Coriolis forces are close to zero. Furthermore, it is assumed that during a free-space movement, the object has no contact with the environment. With these assumptions two characteristic properties of free-space movements can be formulated.

> **Property 1:**
> During free-space movements the external forces remain approximately constant.

As a consequence, the expected standard deviation of the measured forces in each direction is close to zero:

$$\sigma_x \approx 0, \qquad \sigma_y \approx 0, \qquad \sigma_z \approx 0. \tag{2-12}$$

Property 2:
The external forces caused by the object almost entirely consists of gravitational forces.

Therefore, it is expected that the average forces measured by the sensor during a free-space movement are:

$$\mu_x \approx 0, \qquad \mu_y \approx 0, \qquad \mu_z \approx -9.81 \cdot \mathrm{m}_{obj}, \tag{2-13}$$

where m_{obj} is the mass of the object that is moved through free-space.

When manipulating free-space objects, there are often parts of the movement where the object is in contact with the environment and thus only partially moves through free-space. For example, when picking up or placing down the objects during a pick-and-place task. The properties described in Equations (2-12) and (2-13), however, assume that the object has no contact with the environment. In order to still recognize partially free-space movements, the manipulation is segmented into overlapping windows of d_{seg} cm (e.g., if $d_{seg} = 5\,\mathrm{cm}$ the segments are $0\text{-}5\,\mathrm{cm}, 1\text{-}6\,\mathrm{cm}, 2\text{-}7\,\mathrm{cm}$, etc.). The chosen segmentation length d_{seg} should be smaller than the expected length of the free-space movement, such that at least one segment contains the forces that are solely caused by the free-space movement.

For each of these segments the average forces and standard deviation are calculated and given as input to the binary classifier:

$$\text{Input per segment} = [\mu_x, \mu_y, \mu_z, \sigma_x, \sigma_y, \sigma_z]. \tag{2-14}$$

2-2-3 Classification approach

The previous section described the expected average and standard deviation of the forces with respect to the world frame. In reality however, the forces measured by the f/t sensor deviate from the expected values due to sensor imperfections (e.g., noise or drift) and robot behavior (e.g., abrupt movements or workspace limitations). To deal with these uncertainties, the decision boundaries of the binary classifier describe a region around the expected μ and σ values, as illustrated in Figure 2-7.
In order to create this region, the decision boundaries are chosen such that the mean and standard deviation of the forces in x and y direction should lay within an ellipse centered at zero and with diameters μ_{max} and σ_{max}. Furthermore, the mean and standard deviation of the forces in z direction should be less than $\mu_{z,max}$ and σ_{max}, respectively. Therefore, the manipulated object is classified as a free-space object if the following conditions hold for at least one segment:

$$\begin{gathered}\frac{\mu_x^2}{\mu_{max}^2} + \frac{\sigma_x^2}{\sigma_{max}^2} \leq 1, \\ \frac{\mu_y^2}{\mu_{max}^2} + \frac{\sigma_y^2}{\sigma_{max}^2} \leq 1, \\ \mu_z \leq \mu_{z,max}, \quad \sigma_z \leq \sigma_{max},\end{gathered} \tag{2-15}$$

where μ_{max}, σ_{max} and $\mu_{z,max}$ are three tuning parameters that describe the shape of the decision boundaries.

The tuning parameters μ_{max} and σ_{max} are based on the expected uncertainties and anomalies of the f/t sensor. Tuning parameter μ_{max} is mostly dependent on the drift of the sensor, whereas σ_{max} is mostly dependent on the noise on the sensor. For our robot application $\mu_{max} = 1\,\text{N}$ and $\sigma_{max} = 0.3\,\text{N}$ are chosen. The tuning parameter μ_{max} is based on the observed sensor drift and the minimal weight of the objects that ought to be detected, whereas σ_{max} is chosen to be three times[3] the standard deviation of the measured noise during a stationary experiment. Furthermore, the value $\mu_{z,max} = -0.5\,\text{N}$ is chosen such that objects of approximately 50 grams or more are detected while avoiding misclassification due to drift.

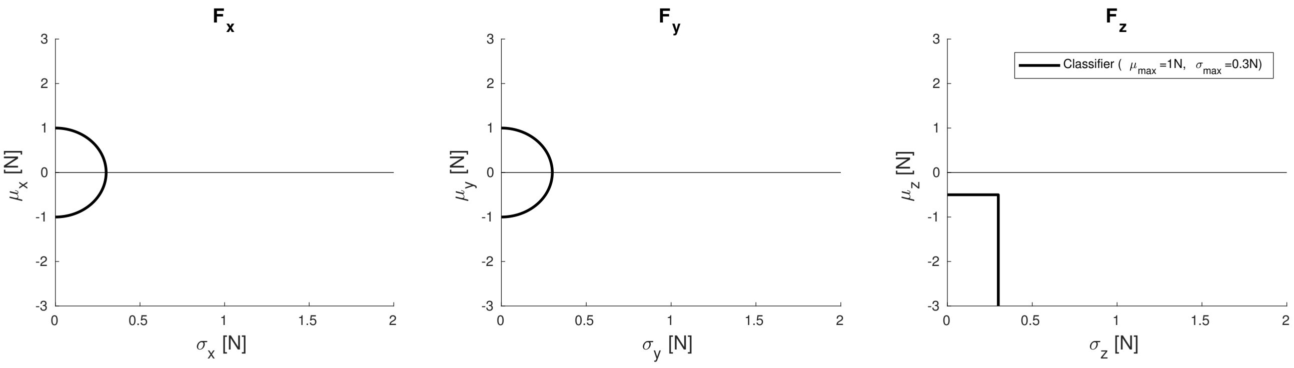


Figure 2-7: Visualization of the 6D decision boundaries using three 2D plots, each showing the mean (μ) and standard deviation (σ) of the forces in either the x, y or z direction. The input of the binary classifier is represented by one point in each subplot. In order to classify the segment as a free-space movement, each of the three points that represent one segment should lay within the classifier bounds (black lines).

2-3 Final KML framework

The results of the KML$_{traj}$ and KML$_{force}$ frameworks can be combined into the final KML framework. As both frameworks have their own strengths, the resulting multi-modal approach has more functionality and can be used on robot manipulators performing all kinds of manipulation tasks. The final framework can classify free-space, prismatic, rotational and complex objects. Furthermore, the framework is able to evaluate the quality of the learned prismatic and rotational models. The frameworks are combined in the following way, which is summarized in Figure 2-8.

After the manipulation task is executed, the KML$_{force}$ framework first determines whether an object is a free-space or constrained object using the binary classifier, as described in Section 2-2. If the object is recognized to be free-space, it will be classified as such after which the corresponding kinematic model is learned. If the object is classified as constrained, the KML$_{traj}$ framework uses the observed trajectory data to determine whether the object is prismatic, rotational or free/complex. This is done using a probabilistic classifier, as described in Section 2-1, which only classifies an object if the confidence level is above a certain learning threshold. If the object is classified as prismatic or rotational, the corresponding model is attached to the object and learned on the robot. If the KML$_{traj}$ framework classifies the

[3]The measured value is multiplied by three in order to make the classifier more robust.

object to be either free-space or complex, the object will be classified as complex as the KML$_{force}$ framework already concluded that the object is constrained. Because the complex objects can be recognized in this way, it is possible to also learn their kinematic models using, for example, Gaussian Processes, as is done by Sturm et al. [11]. This is, however, currently not part of the KML framework.

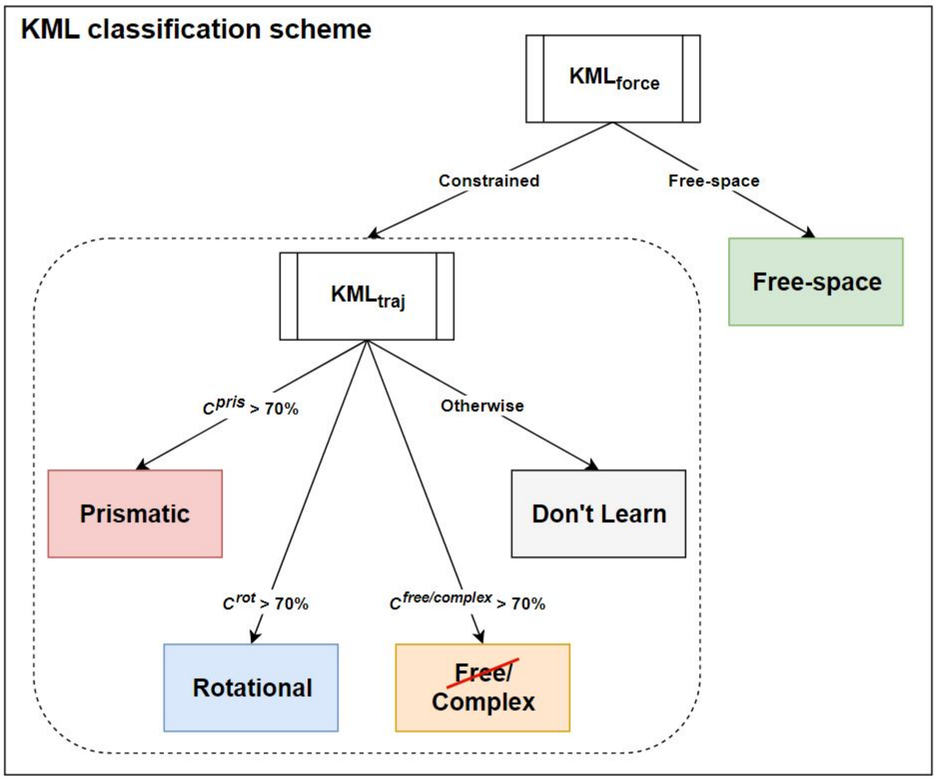


Figure 2-8: The KML classification scheme. The KML framework is a combination of the KML$_{traj}$ and KML$_{force}$ frameworks. The former is a binary classifier, whereas the latter is a probabilistic classifier with a learning threshold of 70%. If the KML$_{traj}$ framework classifies the object to be either free-space or complex, the object is classified as complex since the KML$_{force}$ framework already concluded that the object is constrained.

Chapter 3

Experimental setup

This chapter describes the setup of the experiments used for the validation of the Kinematic Model Learner (KML) framework. First, Section 3-1 describes the robot setup used to perform the experiments. Subsequently, Section 3-2 describes the performed manipulation tasks in more detail. In Chapter 4, the created frameworks are evaluated using the trajectory and force data recorded during these manipulations.

3-1 Robot Setup

The framework was tested on the semi-autonomous care robot 'Marco'. This robot is a prototype of the TIAGo platform. An impression of the TIAGo platform is given in Figure 3-1, in which the relevant parts and reference frames of the robot are depicted. Similar to the TIAGo platform, Marco is equipped with a differential drive base, a RGB-D camera on its head, a 1 Degree of Freedom (DOF) torso and a 7 DOF arm equipped with a 6 DOF ATI Mini45 force/torque (f/t) sensor on its wrist. Marco uses a slightly different 1 DOF parallel gripper and has an additional camera attached on the gripper to increase the depth perception of the operator.

During the experiments the operator has full control over the robot using tele-operation. This is done using a coactive interface [29] with a haptic device that lets the operator sense the forces in the robot's environment [30], as depicted in Figure 3-2. During the manipulations, the operator cannot directly see the robot.

Force/torque data

The force/torque sensor measures the contact forces and torques with a resolution of 0.125 N and 0.003 Nm, at a frequency of 100 Hz and is measured with respect to the sensor frame $\Psi_{f/t}$, as shown in Figure 3-1. As mentioned in Section 2-2-1, the measured f/t data should be pre-processed before it can be used.

Since we are only interested in the forces that are caused by contact with an object, a package is created that computationally eliminates all non-contact related forces. Furthermore, the noise of the sensor is filtered using a first-order Butterworth (low-pass) filter with a cutoff frequency of 4 Hz. Subsequently, it was found that the offset of the sensor changes a lot in between experiments (± 2 N) due to drift. Therefore, the sensor is recalibrated prior to each manipulation. Finally, the measured forces and torques are transformed with respect to the world frame Ψ_W, such that the z-axis is orthogonal to the floor. A more detailed description of these pre-processing steps is given in Appendix A.

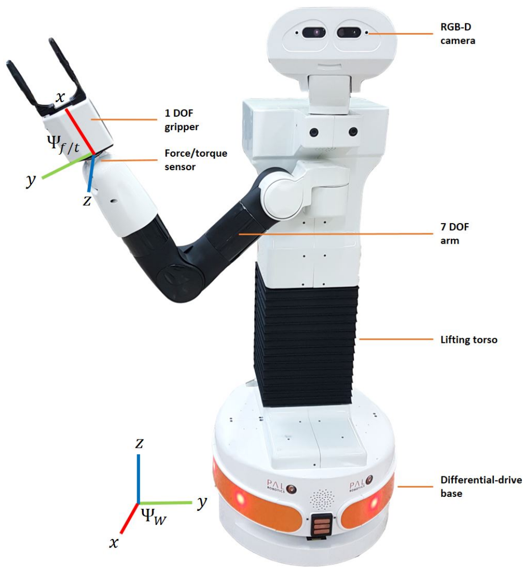


Figure 3-1: TIAGo robot platform with relevant world frame Ψ_W and force/torque sensor frame $\Psi_{f/t}$. Adapted from www.tiago.pal-robotics.com.

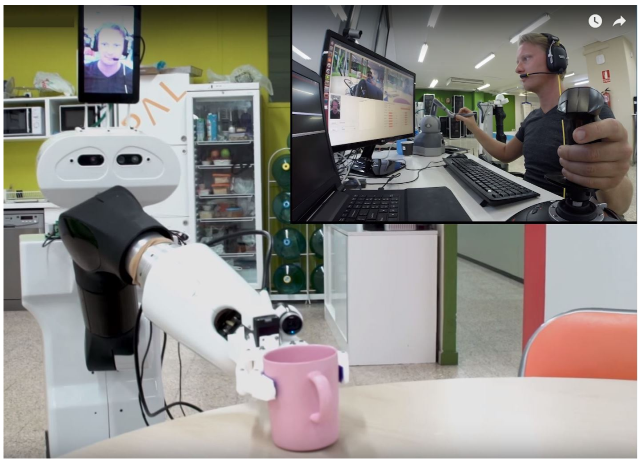


Figure 3-2: The operator controls the robot's movement using a coactive interface with a haptic device that lets the operator sense the forces in the robot's environment. Courtesy of Heemskerk Innovative Technology (HiT).

Magnetic tool

A magnetic tool, as shown in Figure 3-3, is designed to manipulate the prismatic and rotational objects. The magnetic tool is attached to the objects and can be grabbed by the robot, as shown in Figure 3-4. The tool is designed to prevent the robot from breaking the gripper and to simulate the scenario of a non-rigid grasp.

The magnet is attached to a flat surfaced bolt that is rigidly attached to the object. However, the magnet-bolt connection can only absorb (4.0 ± 0.5) N in the planar direction of the magnet and (40.0 ± 1.0) N in the orthogonal direction. The maximal torque around any planar axis is (0.38 ± 0.05) Nm, whereas almost no torque can be applied around the orthogonal axis. As a consequence, the magnet can rotate around the bolt if the forces or torques on the magnet reaches any of these limits. The maximum deviation from the object path is 7 cm, due to the size of the magnetic tool.

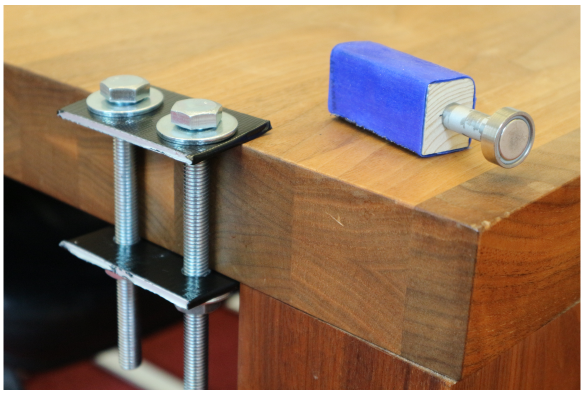


Figure 3-3: The magnetic tool consists of two bolts that can be attached to an object and a magnet that can be grabbed.

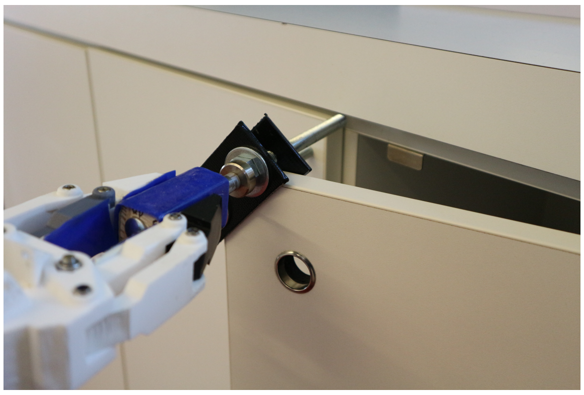


Figure 3-4: The magnetic tool can be grabbed by the robot.

3-2 Manipulation tasks

The performance of the KML framework is evaluated with 9 representative manipulation tasks consisting of 3 pick-and-place (PaP) tasks, opening 3 doors and opening 3 drawers. In the first PaP task the goal is to move a mug from A to B. In the second PaP task an obstacle is inserted, and in the third PaP task the mug starts on top of the obstacle and has to be placed on the table. An overview of the different manipulation tasks is given in Figure 3-5. Each of these different manipulation tasks are repeated 3 times, making up a total of 27 manipulations.

The robot starts at the position and configuration that the operator finds most suitable for the manipulation. As soon as the gripper is closed, the robot starts recording the forces of the f/t sensor and the end-effector poses. The recording is stopped after the gripper is opened again and the manipulation has succeeded. Only successful manipulations are used for classification. A task is considered to be successful if the magnetic tool remains connected, and if the drawers are opened more than 15 cm or if the rotation angle of the doors is at least 30 degrees.

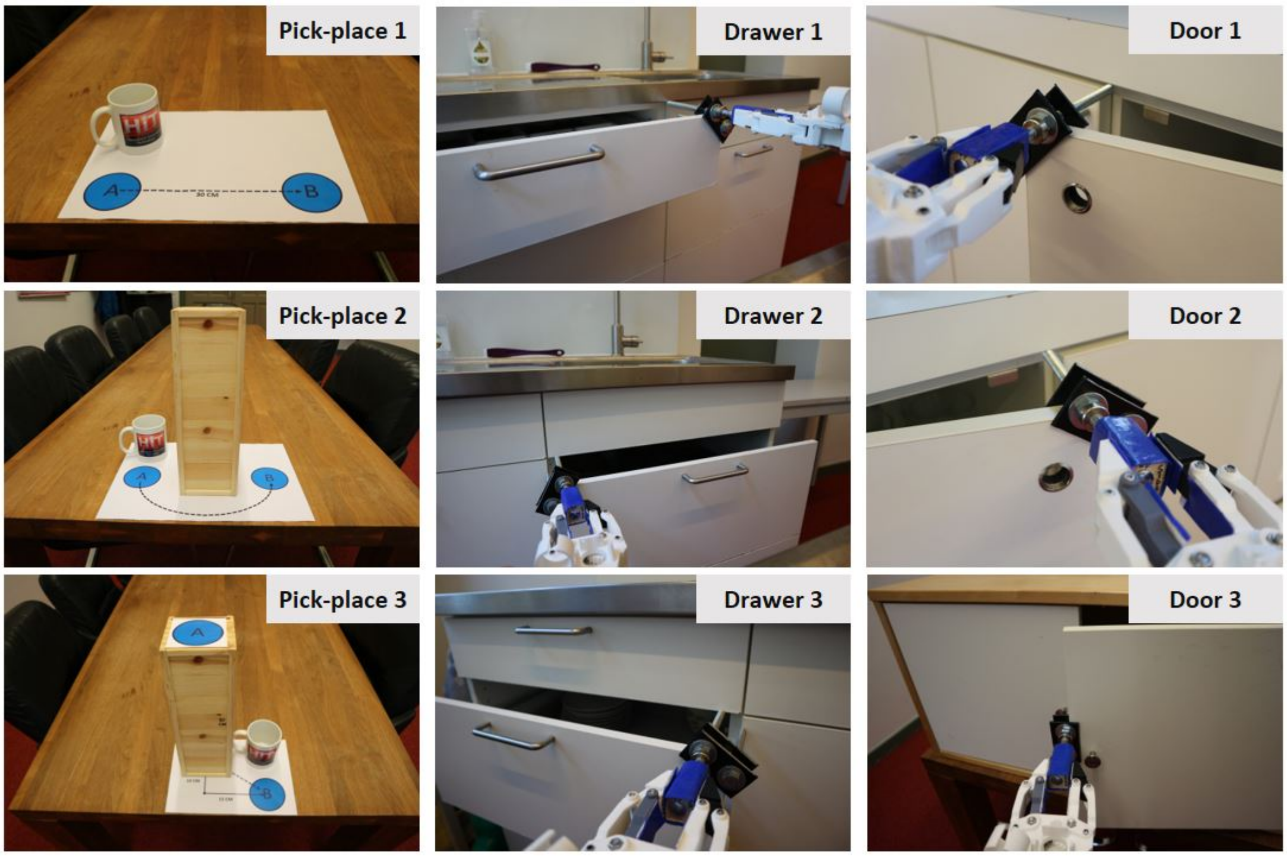


Figure 3-5: The operator performs 9 different manipulation tasks, consisting of 3 PaP tasks, opening 3 doors and opening 3 drawers. In the first PaP task the goal is to move a mug from A to B. In the second PaP task an obstacle in inserted, and in the third PaP task the mug starts on top of the obstacle and has to be placed on the table.

Chapter 4

Results & Discussion

This chapter discusses the results of the Kinematic Model Learner (KML) framework using the recorded trajectory and force data of the manipulation experiments, as described in Chapter 3. Furthermore, the KML$_{traj}$ framework will be compared with the current state-of-the-art `articulation` package created by Sturm et al. [11], and is additionally evaluated using the 'Cody' dataset.

The results of the KML$_{traj}$ and KML$_{force}$ frameworks are discussed in Sections 4-1 and 4-2, respectively. Finally, the results of the multi-modal KML framework are discussed in Section 4-3.

4-1 KML$_{traj}$

In this section the results of the KML$_{traj}$ framework are discussed. In Section 4-1-1, the found kinematic models are validated using visual inspection. After this, the confidence level of each candidate kinematic model is determined in Section 4-1-2. Based on these confidence levels and the predetermined learning threshold, the objects can be classified and the corresponding kinematic models can be learned by the robot. In Section 4-1-3, the robustness of the KML$_{traj}$ framework is evaluated by adjusting the tuning parameter λ, and by analyzing the confidence levels in a worst case scenario. Additionally, the robustness of the KML$_{traj}$ framework is evaluated using the 'Cody' dataset, containing the trajectories of 35 manipulations. Subsequently, Section 4-1-4 explains why the KML$_{traj}$ framework is better suitable to be employed on a real world robot application than the current state-of-the-art `articulation` package. Finally, the results are discussed in Section 4-1-5.

4-1-1 Model fitting

The 3D trajectory data of the performed manipulation experiments are shown in Figure 4-1. A more detailed 2D visualization of the trajectory data is given in Appendix B. Using the trajectory data of each manipulation, the KML$_{traj}$ framework finds the best fitting prismatic ($\mathcal{M}^{pris}$) and rotational models ($\mathcal{M}^{rot}$). The correctness of the found kinematic models can be evaluated by plotting the trajectory together with the kinematic models. By visually inspecting these plots, it can be easily concluded that all of the kinematic models are correctly found.

Moreover, using this visualization, it can be shown that indeed, the least-squares solution ($\lambda = 0$) find $\mathcal{M}^{rot}$ with an undesirable large radius when opening a drawer, as illustrated in Figure 4-2. Similarly, it can be observed that the proposed minimization problem, as described in Equation (2-4), encourages the optimization algorithm to search for a model with a larger rotation angle and thus a smaller radius. This is shown in Figure 4-3. Furthermore, two examples of the kinematic models of a pick-and-place (PaP) and a door opening task can be found in Appendix B.

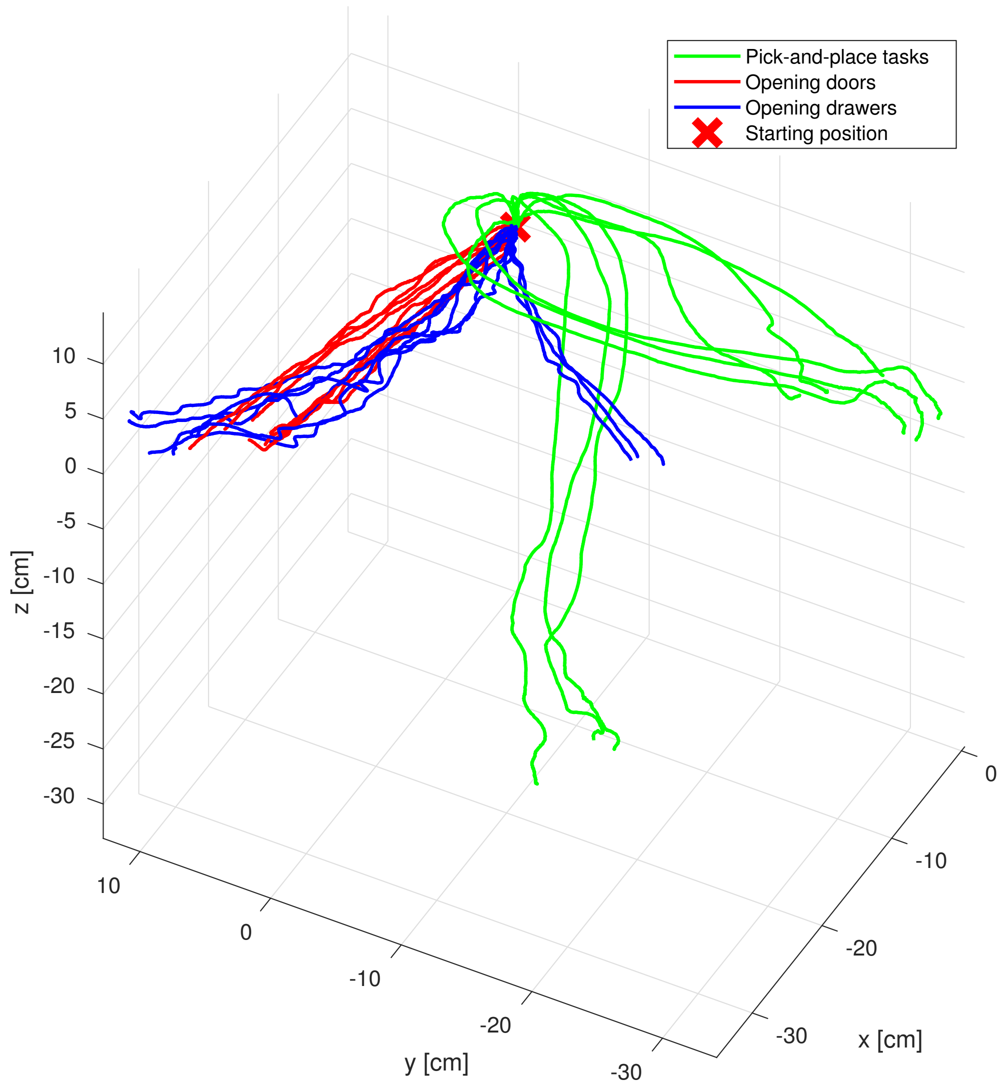


Figure 4-1: A 3D visualization of the 27 manipulations performed during the experiments, as described in Section 3-2. A more detailed 2D visualization of the trajectory data is given in Appendix B.

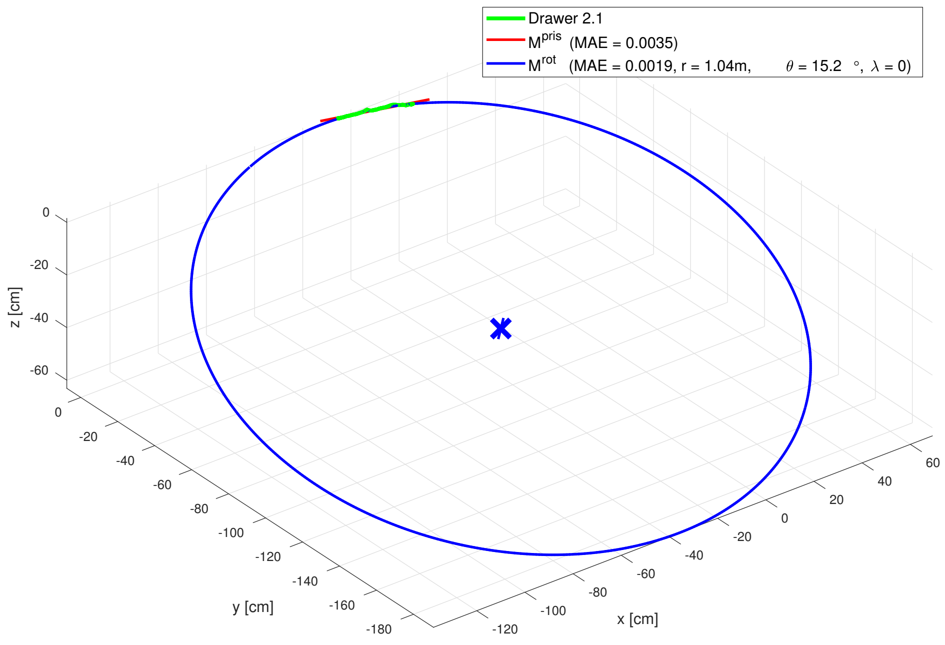


Figure 4-2: Visualizing the 'Drawer 2.1' trajectory and the candidate models $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ using $\lambda = 0$. With $\lambda = 0$ the minimization becomes a least-squares problem without any penalty on the rotation angle. Therefore, the drawer is better described by $\mathcal{M}^{rot}$ with a large radius ($r = 1.05\,\mathrm{m}$) and small rotation angle ($\theta = 15.2°$).

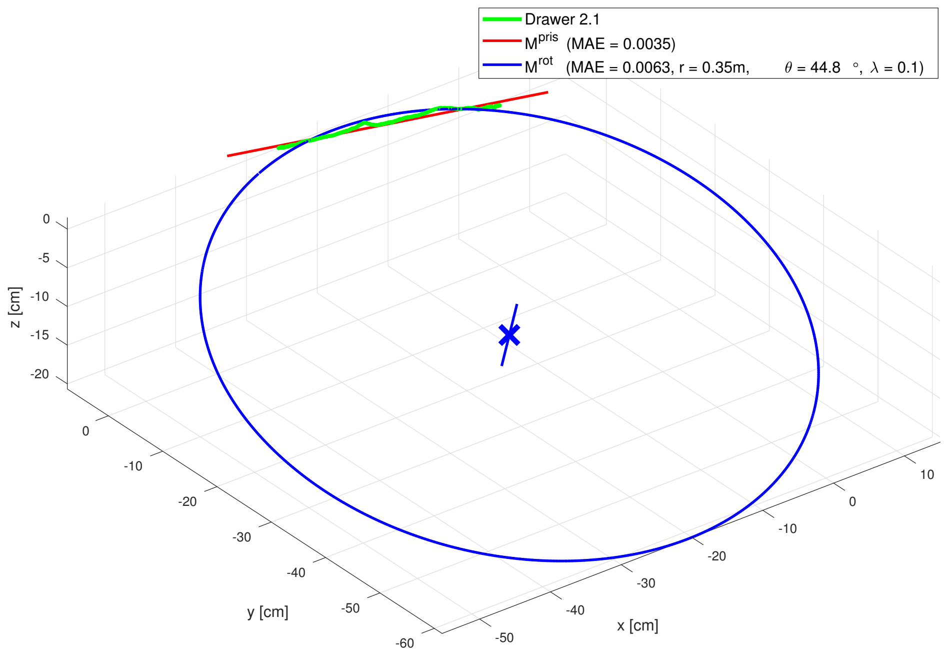


Figure 4-3: Visualizing the 'Drawer 2.1' trajectory and the candidate models $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ using $\lambda = 0.1$. With $\lambda = 0.1$ the optimization function finds a rotational model with a smaller radius ($r = 0.35\,\mathrm{m}$) and larger rotation angle ($\theta = 44.8°$). Consequently, the drawer is correctly classified as a prismatic object.

4-1-2 Classification

After the best fitting prismatic and rotational models are learned, the quality of these models are evaluated using the classification approach, as described in Section 2-1-4. When classifying the objects according to the kinematic models with the lowest Mean Absolute Error (MAE), all prismatic and rotational movements are correctly classified. However, as described in Section 2-1, such a binary classifier cannot be used on the robot, as it is susceptible to learning incorrect or uncertain kinematic models. The resulting confidence levels on the other hand, give more insight into the quality of the learned models and should prevent the robot from learning incorrect or uncertain models. The confidence levels of the manipulations are illustrated in Figure 4-4. In this section, the results and the key values of the confidence levels will be discussed.

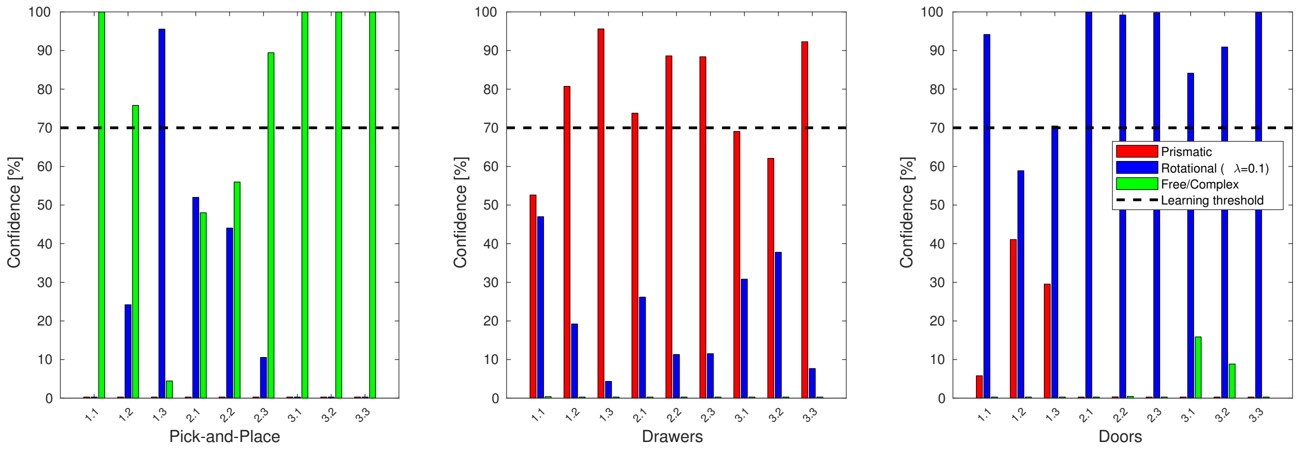


Figure 4-4: The resulting confidence levels of all manipulation tasks. The confidence levels of each manipulation is represented by three bars showing the confidence of the prismatic (left/red bar), rotational (middle/blue bar) and free/complex (right/green bar). The learning threshold is adjustable, allowing robot engineers to decide how well the predicted model should fit before the manipulation robot learns a kinematic model. If the confidence level of all models are below this threshold, no model should be learned.

Because of the used learning threshold of 70%, only 6 out of 9 prismatic models of the 'opening drawer' tasks are actually learned. As described in Section 2-1-4, the best fitting kinematic model is only learned if the difference between MAE^{pris} and MAE^{rot} is significant enough. This can be easily verified by comparing the average MAEs of the models that are learned and those that are not. As shown in Table 4-1, on average, the prismatic models that are correctly learned have a large difference between MAE^{pris} and MAE^{rot} (3.97 mm), whereas the models that are not learned have a small difference (1.23 mm).

Table 4-1: Classification results for prismatic movements based on the confidence levels and the chosen learning threshold.

	MAE$^{pris}_{avg}$ (mm)	MAE$^{rot}_{avg}$ (mm)	Difference (mm)
Correctly learned $\mathcal{M}^{pris}$ (6/9)	3.50	7.47	3.97
Not learned (3/9)	4.23	5.47	1.23

Performing the same exercise on the 'opening door' tasks, 8 out of the 9 rotational models are correctly learned with a large difference between MAE^{pris} and MAE^{rot} (7.89 mm), as can be seen in Table 4-2. One of the rotational models is not learned as its prismatic counterpart also fits well, resulting in a small difference in MAEs (1.40 mm).

Table 4-2: Classification results for rotational movements based on the confidence levels and the chosen learning threshold.

	MAE$^{pris}_{avg}$ (mm)	MAE$^{rot}_{avg}$ (mm)	Difference (mm)
Correctly learned $\mathcal{M}^{rot}$ (8/9)	12.80	4.91	7.89
Not learned (1/9)	4.70	3.30	1.40

On average, the MAEs of the correctly learned prismatic or rotational models are 58% lower than the MAEs of its counterpart. Whether this difference is large enough to learn a kinematic model depends on the robot application. Therefore, it is important to adjust the probability functions and learning threshold to fit the specific robot application.

As mentioned in Section 2-1-4, the KML$_{traj}$ framework cannot reliably be used to recognize free-space or complex movements, as these movements can have strong resemblance with prismatic or rotational movements. Because of this, one PaP movement is misclassified as a rotational object. Table 4-3 shows that indeed the found rotational model has a low MAE (6.60 mm).
Nevertheless, by means of elimination, it is still possible to classify a movement as free-space/complex. If both $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ do not fit well (high MAEs), it can be concluded that the manipulated object belongs to the free/complex object class.[1] As a consequence, 6 out of the 9 PaP movements are correctly classified as a free-space/complex. It should, however, be noted that these results very much dependent on the trajectory of each PaP movement. For example, if all of the PaP trajectories would have moved in a straight or circular fashion, none of the PaP tasks would be classified as free/complex.

Table 4-3: Classification results for free-space movements based on the confidence levels and the chosen learning threshold.

	MAE$^{pris}_{avg}$ (mm)	MAE$^{rot}_{avg}$ (mm)
Correctly learned $\mathcal{M}^{free/complex}$ (6/9)	44.17	21.06
Not learned (2/9)	66.50	10.10
Incorrectly learned (1/9)	14.70	6.60

4-1-3 Robustness analysis

This section evaluates the robustness of the KML$_{traj}$ framework by adjusting the tuning parameter λ, and by analyzing the confidence levels in a worst case scenario. Additionally, the robustness of the KML$_{traj}$ framework is evaluated using the Cody dataset, containing the trajectories of 35 manipulations.

[1]As mentioned in Section 2-1-4, it is impossible to further distinguish the free-space objects from the complex objects using solely trajectory data.

Changing λ

As described in Section 2-1-3, λ can be varied to either increase or decrease the influence of the cost function $J(\theta)$. In general, when increasing λ, the rotation angle of the found $\mathcal{M}^{rot}$ will increase until θ_h is reached, whereas lowering λ decreases the overall influence of the cost function. Furthermore, it was explained that an ordinary least-squares approach ($\lambda = 0$) should not be used to fit rotational models, as a prismatic movement can be approximated very accurately by a circle with an unrealistically large radius and thus a small rotation angle.

As expected, when choosing $\lambda = 0$, none of the prismatic movements (opening drawers) are correctly classified when using a binary classifier based on the lowest MAE. When increasing λ, the rotation angle of the found $\mathcal{M}^{rot}$ will increase until θ_h is reached. As a consequence, the rotational models fit worse, leading to more correctly classified prismatic movements. The classification results for changing λ is shown in Table 4-4. In the performed experiments, the results for the free-space (PaP tasks) and rotational movements (opening doors) are not affected by changing λ, since all of the rotational models already have large rotational angles ($\theta > \theta_h$).

Table 4-4: The number of correctly classified prismatic and rotational objects is analyzed for different λ when using a binary classifier based on the lowest MAE. $\lambda = 0$ shows that the least-squares solution always finds a better fitting $\mathcal{M}^{rot}$. For any $\lambda > 0.1$ the results remain unchanged as θ_h is reached.

λ	prismatic	rotational	average θ for prismatic manipulations
0	0/9	9/9	13°
0.01	7/9	9/9	38°
0.02	8/9	9/9	40°
0.05	8/9	9/9	44°
$\geq$ **0.1**	**9/9**	**9/9**	**45°** ($= \theta_h$)

Worst case scenario

In the worst case scenario of $\lambda = 0$, for all prismatic manipulations, $\mathcal{M}^{rot}$ fits a little bit better than $\mathcal{M}^{pris}$. However, as the difference between the MAEs of the prismatic and rotations models are not significant enough (on average only 1.18 mm), the KML$_{traj}$ framework does not learn any of the incorrectly classified rotational models, as shown in Table 4-5 and Figure 4-5.

It can be concluded that, even though $\lambda > 0.1$ has the best classification results in terms of the lowest MAE, choosing different λ will have no harmful consequences, as the learning threshold prevents the KML$_{traj}$ framework from learning incorrect models.

Table 4-5: Classification results for the prismatic movements (opening drawers) using $\lambda = 0$. None of the candidate models are learned as the difference between MAE^{pris} and MAE^{rot} is not significant enough.

	MAE$^{pris}_{avg}$ (mm)	MAE$^{rot}_{avg}$ (mm)	Difference (mm)
Not learned (9/9)	3.74	2.57	1.18

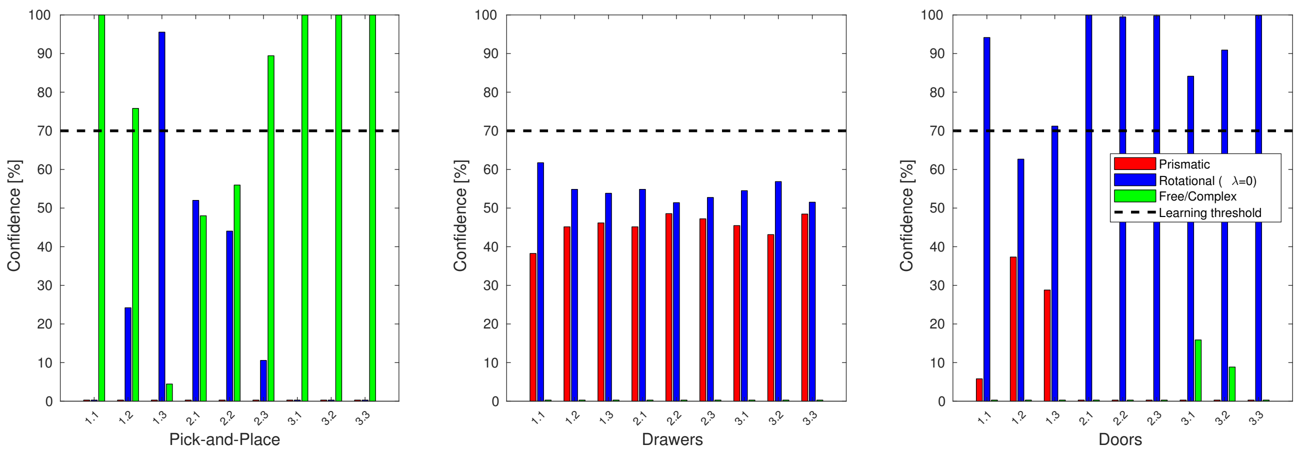


Figure 4-5: Resulting confidence levels of all manipulation tasks when choosing $\lambda = 0$. The confidence levels of each manipulation is represented by three bars showing the confidence of the prismatic (red/left bar), rotational (middle/blue bar) and free/complex (right/green bar). Due to the learning threshold, none of the incorrect rotational models are learned for the 'open drawer' tasks.

The Cody dataset

The KML$_{traj}$ framework is also validated using the Cody dataset [11], which contains the trajectories of 35 manipulation tasks, as shown in Appendix C. All of 35 manipulation were correctly classified (21/21 rotational and 14/14 prismatic movements). During these experiments the robot obtained a rigid grasp. Consequently, the Cody dataset shows smaller deviations from the object constraints than the manipulations described in Chapter 3. As a result, the confidence levels of the correct models are much higher. A more detail analysis of the above mentioned results are given in Appendix C.

4-1-4 Improved current state-of-the-art

Using the manipulations, as described in Chapter 3, the novel approach of the KML$_{traj}$ framework can be compared with the current state-of-the-art `articulation`[2] package created by Sturm et al. [11]. As the `articulation` package does not provide a confidence level of the learned models, it is only possible to compare the `articulation` package with the KML$_{traj}$ framework using binary classification. As described in Section 4-1-3, the binary classifier of KML$_{traj}$ framework selects the kinematic model that has the lowest MAE. The results of this binary classification are shown in Table 4-4. The binary classifier of the `articulation` package selects the kinematic model that has the lowest BIC score.

The results of the `articulation` package are evaluated for different values of its tuning parameter σ_z, as shown Table 4-6. It was difficult to find the most suitable $\sigma_z = 0.05\,\text{m}$, as small deviations from this value resulted in a strong bias towards one of the two models. The results show that larger σ_z values results in a classification bias towards prismatic models, whereas lower σ_z results in a classification bias towards rotational models. Sturm et al. argued

[2]http://wiki.ros.org/articulation

that the `articulation` package correctly classified their data for any σ_z between 0.02 and 0.2 cm. For the dataset described in Chapter 3, however, no σ_z retrieved a 100% classification score. This is most likely because the trajectories of the performed experiments show larger deviations from the actual object constraints than the dataset that was used to validate the `articulation` package (see Appendix C).

Table 4-6: Results of the `articulation` package for different values of the tuning parameter σ_z. The best found value of $\sigma_z = 0.5$ still misclassified 2 rotational movements. Furthermore, small deviations from this optimal value resulted in a strong bias towards one of the two models. Larger σ_z values results in a classification bias towards prismatic models, whereas lower σ_z results in a classification bias towards rotational models.

σ_z	prismatic	rotational
0.01	0/9	9/9
0.02	1/9	9/9
0.05	**9/9**	**7/9**
0.1	9/9	4/9
0.2	9/9	0/9

The tuning of the parameter σ_z is made especially difficult because the relation between σ_z and the final BIC score is dependent on the number of samples N. In Table 4-7, it is verified that the `articulation` package is dependent on the number of samples by downsampling the trajectory data by a factor 5. In general, the `articulation` package favours $\mathcal{M}^{pris}$ for lower number of samples due to the $k \log(N)$ term in Equation (1-2). Because of this dependency, the `articulation` package is harder to tune, and less robust to changes in data acquisition, such as changes in the sampling frequency.

It is validated that the results of the KML$_{traj}$ framework (see Table 4-4), on the other hand, do not change when downsampling the trajectory data by a factor 5. Except for the obvious fact that the optimization function might find a model which is slightly different than before.

Table 4-7: The results of the `articulation` package is dependent on the number of samples. When downsampling the trajectory data by a factor 5 the average number of data point reduces from $N_{avg} = 472$ to $N_{avg,downsampled} = 94$. After downsampling, the results of the `articulation` package change dramatically. For the downsampled scenario the best choice for σ_z has now become $\sigma_z = 0.02$.

σ_z	prismatic	rotational
0.01	0/9	9/9
0.02	**8/9**	**9/9**
0.05	9/9	6/9
0.1	9/9	0/9
0.2	9/9	0/9

When comparing Table 4-6 with Table 4-4, it can be concluded that the KML$_{traj}$ framework outperforms the current state-of-the-art `articulation` package when using a binary classifier, as the KML$_{traj}$ framework correctly classified all of the prismatic and rotational movements.

Furthermore, in contrast to the `articulation` package, the results of the KML$_{traj}$ framework are independent on the number of samples. Therefore, the KML$_{traj}$ framework is easier to tune and more robust to changes in the data acquisition.

4-1-5 Discussion

In this section the results of the KML$_{traj}$ framework are discussed. The first three topics discuss the advantages of the KML$_{traj}$ framework compared to the state-of-the-art. The last topic discusses the main limitation of the used approach.

The confidence level is a much better tool for classification than a binary classifier.

The confidence level has three main benefits. First of all, the confidence level evaluates the quality of the learned kinematic model. Secondly, a kinematic model is only learned if this model fits significantly better than the alternative models. If multiple models are likely to be correct, none of the models are learned in order to avoid misclassification. Finally, as the total confidence of all models sum to 100%, the quality of the models can be quickly evaluated by an operator or robot engineer.
A binary classifier on the other hand has none is these benefits. On the contrary. A simple binary classifier is prone to misclassification, learns every model independently of its quality, and gives no feedback on the quality of the learned models.

The KML$_{traj}$ framework is more robust than the state-of-the-art.

As the magnetic tool is used during the experiment, the resulting trajectories occasionally show large deviations from the constrained object path. This is mostly due to the fact that the operator cannot directly see the robot and its environment during the manipulation and is therefore dependent on the visual feedback of the cameras and the haptic feedback. In spite of these large deviations, the KML$_{traj}$ framework is still able to correctly classify all of the manipulated objects, in contrast to the `articulation` package. Therefore, it can be concluded that the KML$_{traj}$ is more robust to deviations from the actual constrained object path. Nevertheless, it is important to mention that more experiments should be performed in order to further substantiate this conclusion.

The confidence levels can be easily adjusted to different robot application.

The probability functions that are used to calculate the confidence levels can be easily adjusted to different robot applications by adjusting μ and σ. Users can adjust these parameters to fit their robot application. For example, when the robot application requires a very accurate model, a small increase in MAE can have a large effect on the usability of the corresponding kinematic model. Therefore, the μ and σ parameters should be decreased, such that only very accurate kinematic models are learned by the robot.

The results of the KML$_{traj}$ framework depend on the length of the trajectory.

The MAE is used in the probability functions to calculate the confidence levels. A downside of this approach is that, even though the MAE represent the average error between the observed trajectory and the kinematic model, the maximum MAE is still limited by the length of the trajectory. For example, if the observed trajectory is very small, the largest possible MAE will also be very small. As a consequence, the probability functions should be redesigned if the observed trajectory is significantly smaller than originally anticipated.

4-2 KML$_{force}$

In this section the results of the KML$_{force}$ framework are discussed. The goal of the KML$_{force}$ framework is to recognize free-space movements using the force data recorded during the manipulations. In order to recognize the free-space movement, the raw force data first has to be pre-processed (Section 2-2-1), such that only the external forces caused by the manipulated object are left. A detailed description of the pre-processing steps performed on Marco are given in Appendix A.

In Section 4-2-1 the performed manipulations are classified using a binary classifier. Subsequently, Section 4-2-2 evaluates the robustness of the KML$_{force}$ framework by changing the tuning parameters, which define the decision boundaries. Finally, the results are discussed in Section 4-2-3.

4-2-1 Classification

As explained in Section 2-2, prior to the classification, the force data is split into segments of $d_{seg} = 10\,\mathrm{cm}$, such that partially free-space movements can also be detected. Using this segmentation length, the 27 manipulations were segmented into 860 segments having an average of 520 samples per segment. The mean and standard deviation of each segment is given as input to the binary classifier. The input data of all force segments can be visualized using three 2D plots, as depicted in Figure 4-6. In this figure, each subplot shows the mean and standard deviation of the forces in either the x, y or z direction. Consequently, one segment is represented by 1 point (*) in each subplot. The binary classifier classifies a segment as a free-space movement if each of the three points lays within the corresponding decision boundaries (black lines), i.e., if the conditions described in Equation (2-15) hold for at least one segment.

The mean and standard deviation of the pick-and-place segments (green) are clustered around the expected values, as described in Section 2-2-3. Only a few segments are to be found outside of this cluster. These anomalies are caused by the contact transitions during the PaP tasks when picking up or placing down the mug. From the 860 classified segments (across all 27 manipulations), 394 belong to the pick-and-place tasks and 464 belong to the the prismatic (opening drawers) and rotational (opening doors) tasks. The results of the segment classification are summarized in Table 4-8.

Table 4-8: Classification results of the force segments using the KML$_{force}$ framework.

	Free-space movements	Constrained movements
Correctly classified	341	464
Incorrectly classified	55	0
Total number of segments	396	464

The binary classifier correctly classifies all segments from the constrained (prismatic and rotational) movements. Furthermore, 341 out of 394 pick-and-place segments are recognized as free-space movements. Out of the 55 incorrectly classified segments, approximately 18 are caused by the contact transitions (picking-up and putting down) of each pick-and-place task. The other incorrect classifications are caused by the drift of the f/t sensor in x direction during the manipulation. This is a known problem for the used f/t sensor, as the wires are inadequately connected to the robot. The classifier will most likely yield better results if this problem is accounted for, as the mean forces are expected to lay closer to zero.

A movement is classified as free-space if at least one of its segments is classified as a free-space movement. As a consequence, all free-space movements (9/9) and all constrained movements (18/18) are correctly classified.

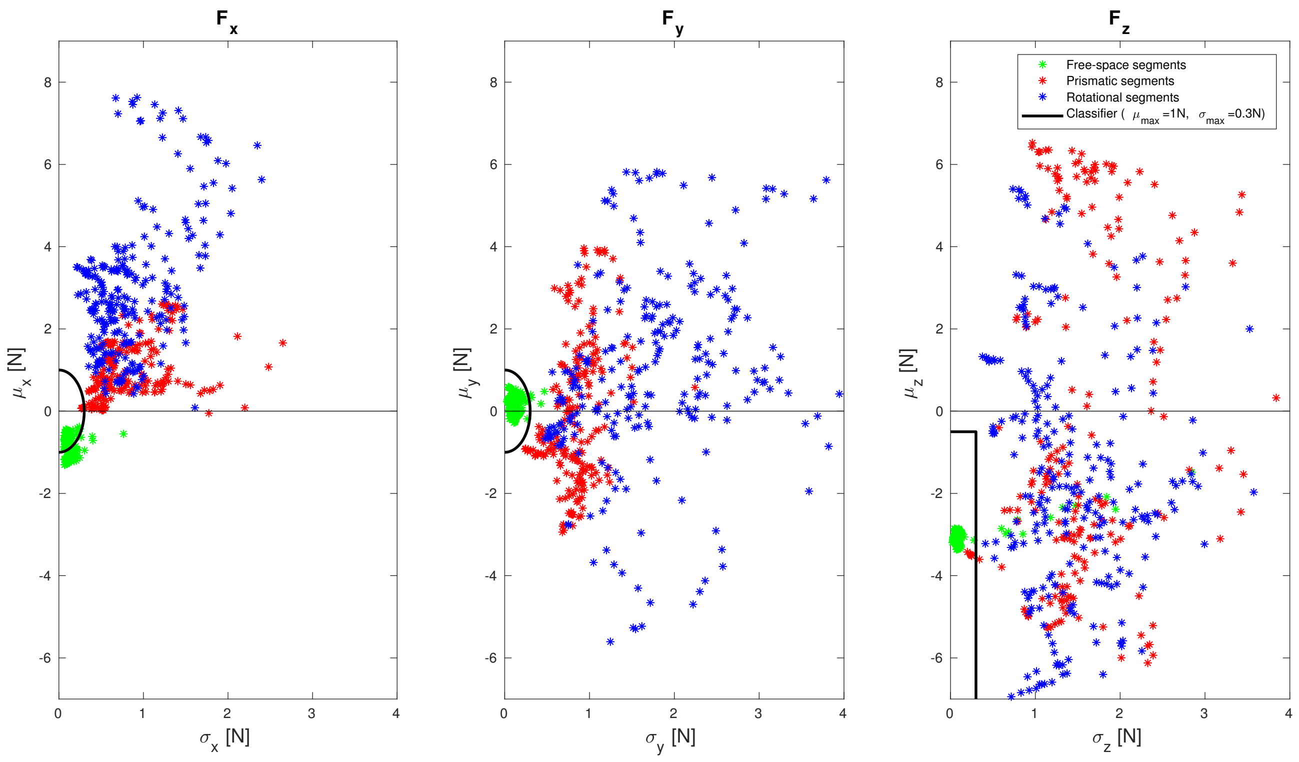


Figure 4-6: Mean and standard deviation of the segmented trajectories. One segment is represented by 1 point (*) in each subplot. In order to classify the segment as a free-space movement each of the three point that represent one segment should lay within the decision boundaries (black lines).

4-2-2 Robustness analysis

The results in the previous section were based on the following tuning parameters: $d_{seg} = 10\,\text{cm}$, $\mu_{max} = 1\,\text{N}$ and $\sigma_{max} = 0.3\,\text{N}$. In this section the robustness of the KML$_{force}$ framework is tested by varying either d_{seg}, μ_{max} or σ_{max}, while keeping the other variables constant. The robustness is expressed as the range of the tuning parameters for which 100% of the movements are correctly classified. It should be noted that the obtained results are dependent on the quality of the f/t sensor and the smoothness of the robot manipulations, as will be discussed in Section 4-2-3.

The results of the robustness analysis are summarized in Table 4-9. First of all, all manipulations are correctly classified up until the point where the length of the segments become smaller than 3 cm. This is logical as it is more likely that the measured forces have a smaller deviation when measuring a smaller part of the manipulation. This illuminates the fact that the KML$_{force}$ framework is less robust for small trajectories. However, as the trajectories of a manipulation robot are typically much larger ($d_{tot} \gtrapprox 20\,\text{cm}$), this does not have to be a troublesome limitation. A simple solution to this limitation would be to only learn a kinematic model if the trajectory is of a certain minimal length. Secondly, it was found that for the tuning parameter μ_{max}, perfect results are obtained for any value greater than or equal to 0.5 N. Because of the drift that occurs during the experiments, some free-space movements are misclassified as constrained when $\mu_{max} < 0.5\,\text{N}$. Finally, σ_{max} can be chosen anywhere between 0.05 and 0.9 N. This range is rather large, considering the fact that for many of the constrained segments, the standard deviation (in either the x, y or z direction) is smaller than 0.9 N (see Figure 4-6).

It can be concluded that the KML$_{force}$ framework is very robust to changes in each of the tuning parameters. Furthermore, the appropriate tuning parameters can be easily determined by assessing the sensor quality and the expected length of the free-space movements.

Table 4-9: The robustness of the KML$_{force}$ framework is tested by varying either d_{seg}, μ_{max} or σ_{max}, while keeping the other variables constant. Each row of the table shows the range in which one variable can change while preserving the 100% classification score.

σ_{max} (N)	μ_{max} (N)	d_{seg} (cm)
0.3	1.0	$[3, d_{tot}]$
0.3	$[0.5, \infty]$	10
$[0.05, 0.9]$	1.0	10

4-2-3 Discussion

In this section the results of the KML$_{force}$ framework are discussed.

The KML$_{force}$ reliably recognizes free-space movements.

As can be seen in Figure 4-6, the mean and standard deviation of the forces during a free-space movement lay close to the expected values described in Section 2-2-2. Furthermore, the free-space and constrained movements are easy to discriminate, since the constrained movements have significantly larger standard deviations. Consequently, the KML$_{force}$ framework can reliably recognize objects moving through free-space. This, however, also directly indicates the main limitation of the KML$_{force}$ framework, namely that the free-space objects have to move through free-space and thus have no contact with the environment, in order to be correctly classified. More specifically, the free-space object has to be moving through free-space for at least d_{seg} cm before it will be correctly classified.

The classification results are (mostly) independent of the used filter, as long as the decision boundaries are adjusted accordingly.

The mean and standard deviation of the force segments are calculated after the signal has been filtered with a low-pass filter, as described in Appendix A-3. Naturally, the classification results are affected by this filter, because the standard deviation of the force data will decrease. However, when a filter is correctly designed, it only removes the noise, while leaving the rest of the signal mostly untouched. As a consequence, the standard deviation of the free-space and constrained segments will be lowered approximately the same amount due to filtering. Consequently, it is expected that different filters have almost no effect on the classification results, as long as the decision boundaries are adjusted accordingly. Analogous, unfiltered data can be used as well.

The robustness of the KML$_{force}$ framework is dependent on the quality of the f/t sensor and the smoothness of the robot manipulation.

As the binary classifier is based on the mean and standard deviation of the input forces, the quality of the f/t sensor can have a large effect on the robustness of the KML$_{force}$ framework. For example, if the noise level of the f/t sensor increases significantly, a free-space movement could be misclassified as constrained. Moreover, drift also plays a major role in the robustness of the classifier. For example, if the sensor drifts more than the chosen μ_{max} value, a free-space movement will also be misclassified as constrained.
Additionally, when the used robot performs smoother manipulations (lower forces), the framework becomes less robust, as the mean and standard deviation of the constrained movements are likely to move closer to zero. As a consequence, some of the constrained movements could be classified as free-space.

The classifier is designed using world knowledge instead of using machine learning.

The created classifier is based on the characteristic properties of free-space movements and knowledge about the sensor quality. Machine learning, on the other hand, could also be used to learn a classifier. There are two downsides to using machine learning. First of all, in order to learn a classifier, a large dataset is required to avoid sparse regions. Secondly, it can be difficult to find a classifier that correctly classifies unseen instances because the machine learning algorithm can both overfit or underfit the training data. Consequently, it is more sensible to use the available knowledge about the free-space characteristics to design a classifier.

4-3 Combining KML$_{traj}$ and KML$_{force}$

Finally, the results of the KML$_{traj}$ (Section 4-1-2) and KML$_{force}$ (Section 4-2-1) frameworks are combined to form the multi-modal KML framework, as describe in Section 2-3. When combining the obtained results, all free-space (mugs), prismatic (drawers) and rotational (doors) objects are correctly classified. This is done in two steps, as illustrated in Figure 2-8.

First, the KML$_{force}$ framework determines whether the object is a free-space or constrained object using a binary classifier. As a result, each of the 9 pick-and-place tasks are correctly

recognized as free-space movements. As a free-space object has no constraints, the learned kinematic model is empty. Additionally, the KML$_{force}$ framework correctly classifies all drawers and doors as constrained objects.
Secondly, as the KML$_{force}$ framework is unable to further differentiate the constrained objects into either prismatic, rotational or complex objects, these manipulations are given as input to the KML$_{traj}$ framework. The KML$_{traj}$ framework correctly classifies all the drawers as prismatic and all the doors as rotational objects. Because of the learning threshold of the KML$_{traj}$ framework, 14 out of the 18 kinematic models are actually learned and attached to the corresponding object.

After the results of the KML$_{traj}$ and KML$_{force}$ frameworks are combined, all of the manipulations are correctly classified. Consequently, it can be concluded that the KML framework is capable of recognizing free-space, prismatic and rotational objects, and learn their kinematic models. Additionally, the KML framework is able to asses the quality of the learned models and can prevent the robot from learning incorrect or uncertain models. As a consequence, 24 of the 27 kinematic models are actually learned by the robot.

4-3-1 Discussion

In Sections 4-1-5 and 4-2-3 the results of the KML$_{traj}$ and KML$_{force}$ frameworks are discussed in detail. This section provides a more general discussion of the assumptions, results and implications of the final KML framework.

In order to create an understandable and adjustable framework, a well balanced number of tuning parameters is required.

One of the goals of this Master's thesis is that the framework is both understandable to robot engineers, and adjustable to different robot applications. Whether this goal is achieved depends on the number and understandability of each tuning parameter. If the number of tuning parameters is too small, the robot engineer might not have enough freedom to adjust the framework to the used robot application. On the other hand, too many or too complex tuning parameters can make the framework hard to comprehend. In total, the proposed KML framework has 12 different tuning parameters that can easily be determined by either assessing the quality of the force/torque sensor, or by assessing the general purpose of the robot application. Because of the simplicity of the used parameters, the framework is believed to be both easily understandable and well adjustable. The used tuning parameters are summarized in Appendix D.

The generalizability of the learned kinematic models depends on the used grasping technique.

In many robot applications, a non-rigid grasp is used during the manipulation. Consequently, the observed trajectory deviates from the actual object constraints. As the specific deviations found during these manipulations are dependent on the used grasping tool, the learned kinematic models can only be used on other robots if they use a similar grasping tool. On the other hand, if the robot has a rigid grasp during the manipulations, the observed trajectory is the same as the actual object constraints. As a consequence, the learned kinematic model can be used by any robot that has a rigid grasp during manipulation.

Chapter 5

Conclusion & Future work

5-1 Conclusions

In this thesis, the novel Kinematic Model Learner (KML) framework is presented that aims to solve some of the most pressing problem for robots learning kinematic models based on given demonstrations, as described in Section 1-3. The created KML framework is a combination of two separate frameworks that complement each other: KML$_{traj}$ and KML$_{force}$. The designed frameworks are experimentally validated by performing a total of 27 demonstrations on the care robot Marco using tele-operation. The trajectory and force data of these manipulations were used as inputs to validate each framework separately. Additionally, the KML$_{traj}$ framework is also evaluated using the Cody dataset, containing the trajectories of 35 different manipulation tasks. The main conclusions of this research are:

- The KML$_{traj}$ framework can classify and learn the kinematic models of the prismatic and rotational objects more robustly than the state-of-the-art `articulation` package. Additionally, the confidence levels used in the KML$_{traj}$ framework enable the robot or a human operator to instantly evaluate the quality of the learned models. Furthermore, the learning threshold makes the framework even more robust, as it prevents the robot from learning incorrect or uncertain models. The biggest limitation of the KML$_{traj}$ framework, however, is that trajectory data alone is not sufficient to reliably recognize free-space objects, as these objects often move in a circular or straight fashion. In order to solve this problem, the KML$_{force}$ framework is created.

- The KML$_{force}$ framework can robustly recognize free-space movements because of the characteristic properties of these movements. A robustness analysis has shown that the binary classifier used by the KML$_{force}$ framework can reliably classify free-space and constrained movements. In order to deal with partially free-space movements, the trajectory is split into segments. This enables the framework to recognize all of the performed pick-and-place tasks.

- The final KML framework combines the strengths of the KML$_{traj}$ and KML$_{force}$ frameworks. As a result, the KML framework is capable of recognizing free-space, prismatic and rotational objects based on a given (tele-)demonstration, and learn their kinematic models. Additionally, the KML framework is able to asses the quality of the learned models and can prevent the robot from learning incorrect or uncertain models. Additionally, by means of elimination the framework should also be able to classify complex movements. This latter is, however, not yet validated with experiments and should be tested in future work. Finally, the framework can be easily adjusted to different robot applications as the effects of the tuning parameters are easy to understand and can be determined either by assessing the robot applications or by performing simple experiments.

5-2 Future Work

Although promising results have been found towards making robots learn kinematic models based on demonstrations, this work does not complete this topic. Many more developments can be studied in future research. The following topics pose as candidates for future research:

- The learned kinematic models do not provide the robot with any information on the forces that are necessary to manipulate the objects. Using the learned kinematic model and available force data it is possible to differentiate the forces into normal and tangential forces. One interesting research direction would be to use these normal and tangential forces to create a dynamic model of the manipulated object. If both the kinematic and dynamic model of an object are known, the robot knows the path it has to follow and the required forces to realize this movement [3].

- Currently the framework finds the best fitting kinematic model using a naive least squares optimization function with an additional cost function for the rotation angle. The experiments performed in this study have shown that this method performs well. However, least-squares solutions are known to be sensitive to outliers and therefore might not be the best approach. A possible solution to this problem, is to use a Bayesian approach for fitting the models [19]. Future studies could research the possible improvements when using the Bayesian approach to find the best fitting kinematic models.

- In theory, the KML framework should be able to correctly recognize complex movements. This is, however, not validated in this study, as no objects of the complex class were manipulated in the experiments. Future research could perform more extensive experiments, where also complex objects are manipulated. Furthermore, the robustness of the framework could be validated using different robot manipulators.

- The KML framework solely uses initial demonstrations to classify an object and learn its kinematic model. However, the robot could also use additional demonstrations or user feedback to improve the task performance. Users could provide feedback, such as the correct plane of rotation or the estimated radius/opening angle of a door, using a specially designed coactive interface. This poses two interesting research topics. First of all, future research could investigate the best way to gather additional user feedback. Secondly, future research could investigate the possibility to combine multiple manipulations of the same object, such that a more accurate kinematic model can be obtained.

Appendix A

Pre-processing force/torque sensor

A-1 Computational elimination of non-contact forces and torques

Because the force/torque (f/t) sensor is attached to the wrist of the manipulator, the raw sensor data includes the inertial, centrifugal, Coriolis and gravitational forces and torques caused by the weight and the motion of the gripper that is attached to the sensor. As explained in Section 2-2-1, in order to obtain the f/t measurements that are solely caused by the manipulated object, the forces and torques caused by the end-effector and the offset caused by the drift of the sensor should be removed computationally.

The framework that currently removes the gravitational forces/torques on robot Marco is not taking the configuration of the gripper into account. As a consequence, the current framework works very poorly. Therefore, a new gravity compensation framework is designed which uses the mass, center of mass (COM), and robot configuration to obtain solely the external forces. The used method to obtain the external forces is described in more detail by Stilman [28].

As the velocity and acceleration of the gripper is generally low, the inertial, centrifugal and Coriolis forces and torques are negligible. The gravitational forces/torques caused by the gripper, however, have significant influence on the raw sensor data and can be removed in two steps. First of all, the forces caused by the end-effector are removed by measuring the gripper's mass on a scale ($m_{gripper}$= 0.405 kg), and by subtracting the resulting gravitational forces (w.r.t. $\Psi_{f/t}$, as illustrated in Figure 3-1) from the raw sensor data. Secondly, the non-contact torques are removed by multiplying the gravitational forces with the COM of the gripper and rotating these torques into the sensor frame $\Psi_{f/t}$. The COM of the gripper is calculated by measuring three static gripper configurations with and without the gripper attached to the sensor. Using these experiment it was found that the gripper is close to symmetric in the y and z directions, whereas the COM in the x direction of the sensor frame is calculated to be 0.053 m.

Furthermore, it was discovered that the sensor has significant issues with drift in between and during experiments. The effect of the drift is minimized by regularly removing the offset between the measurements and the true state of the sensor. The offsets are computed by measuring the forces and torques of a stationary empty gripper and removing the gripper's own weight.

The new gravity compensation framework was evaluated with an experiment where an empty gripper rotates extensively around the wrist of the robot. As is shown in Figure A-1, the new framework outperforms the old one as the obtained forces are (as expected for an empty gripper) close to zero, independently of the gripper configuration.

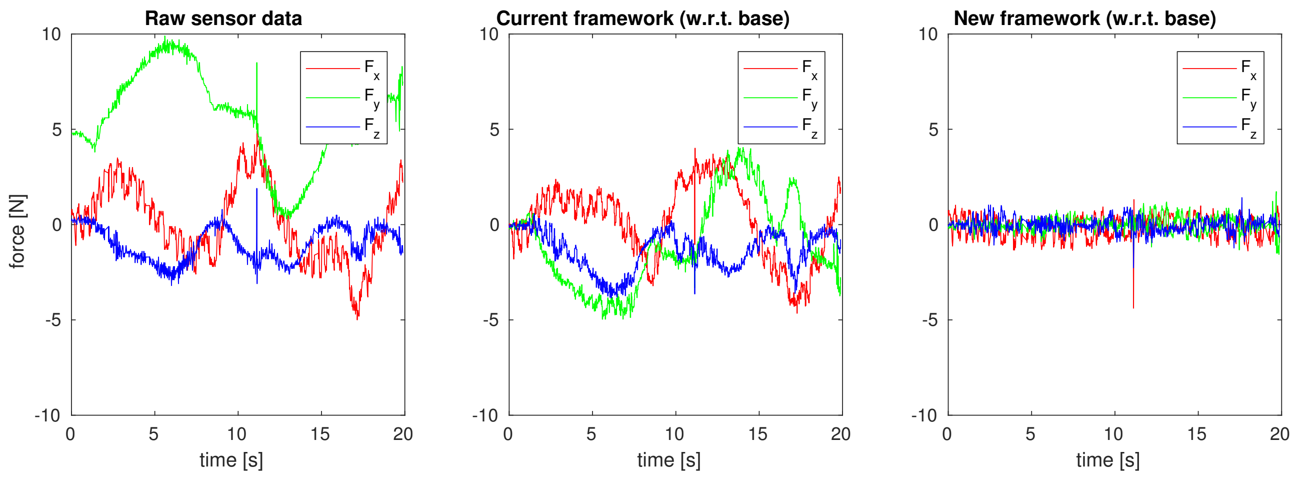
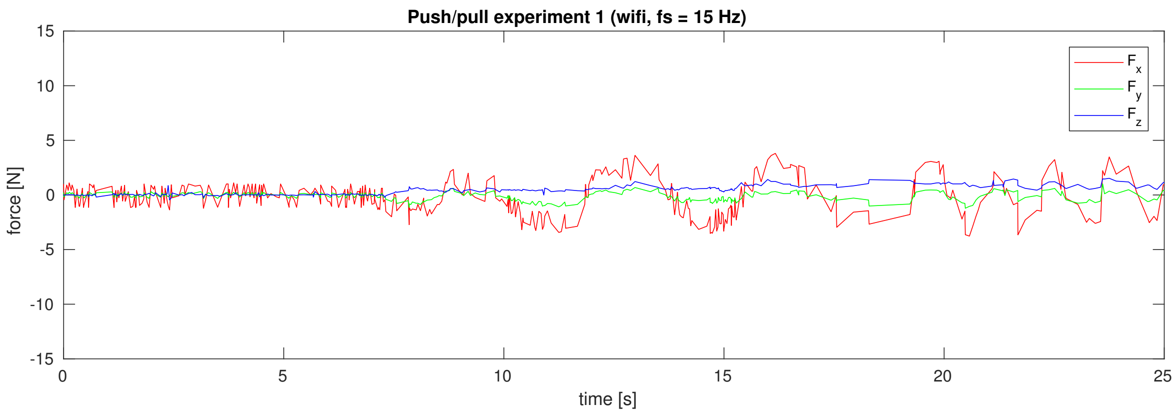


Figure A-1: Rotating the gripper extensively around the wrist. The raw data measured by the f/t sensor is shown in the left figure. The new gravity compensation framework (right) outperforms the old one (middle) as the obtained forces are (as expected for an empty gripper) closer to zero, independently of the gripper configuration.

A-2 Latency

Latency is an important topic in robotics because a small delay in data acquisition can have a huge influence on the robot's performance. These delays can, for example, causes aliasing or oscillations when tele-operating the robot. Therefore, the latency of the robot is evaluated by performing two different stationary pull/push experiments while obtaining the data using either a wifi and a LAN connection. In these experiments a human pushes and pulls the gripper (while pointing forward) in different frequencies.

Based on the resulting data it is concluded that the frequency of the data acquired using wifi is not high enough (only 15 Hz). Because of this low sampling frequency the signal is significantly distorted due to aliasing. Therefore, the f/t data will be acquired using a LAN cable. This resulted in an increase of sampling frequency from 15 Hz to 100 Hz. The different signals are shown in Figure A-2.



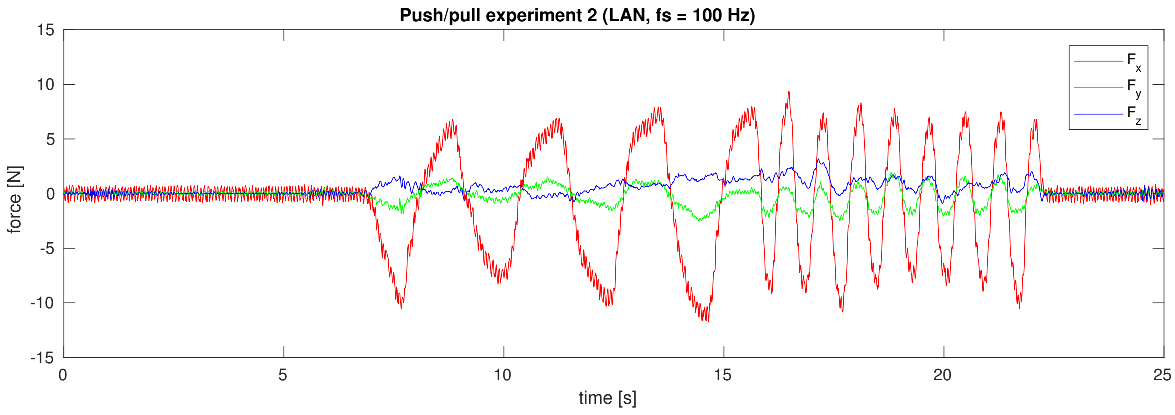


Figure A-2: Force data of the two push/pull experiments using wifi (top) and a LAN cable (bottom). The quality of the data is higher for the LAN cable, as the sampling frequency is 100 Hz instead of the 15 Hz obtained using wifi. In the first 5 seconds of the first experiment aliasing is clearly visible. **Note:** the amplitude and frequency of the forces are different because they represent two different push/pull experiments performed by a human.

A-3 Filtering force data

The noise of the sensor is filtered using a first-order Butterworth (low-pass) filter with a cut-off frequency of 4 Hz. With this filter the higher frequency noise signals are cut-off. Consequently, the noise on the force data is reduces while maintaining the forces caused by the object manipulation. The filter is validated using the seconds pull/push experiment as shown in Figure A-2. When analyzing the filtered signal, as shown in Figure A-3, it is clearly visible that the noise of the data is removed while leaving the characteristic push/pull forces intact.

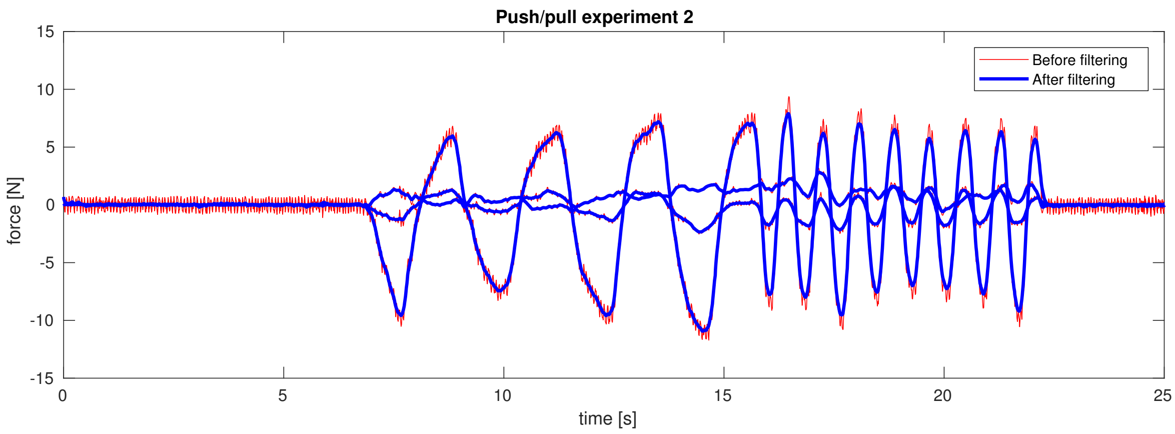


Figure A-3: Force data of a push/pull experiment before (red) and after (blue) filtering using a first-order Butterworth (low-pass) filter with a cut-off frequency of $4\,\mathrm{Hz}$.

Appendix B

Trajectory data in more detail

Even though the 3D trajectories, as shown in Figure 4-1, gives an good intuition about the recorded trajectory data, it does not show much detail and does not distinguish between the 9 different manipulation tasks. Figure B-1 displays the trajectory data in more detail. Furthermore, examples of the fitted models for a 'pick-and-place' and 'opening door' task are shown in Figures B-2 and B-3.

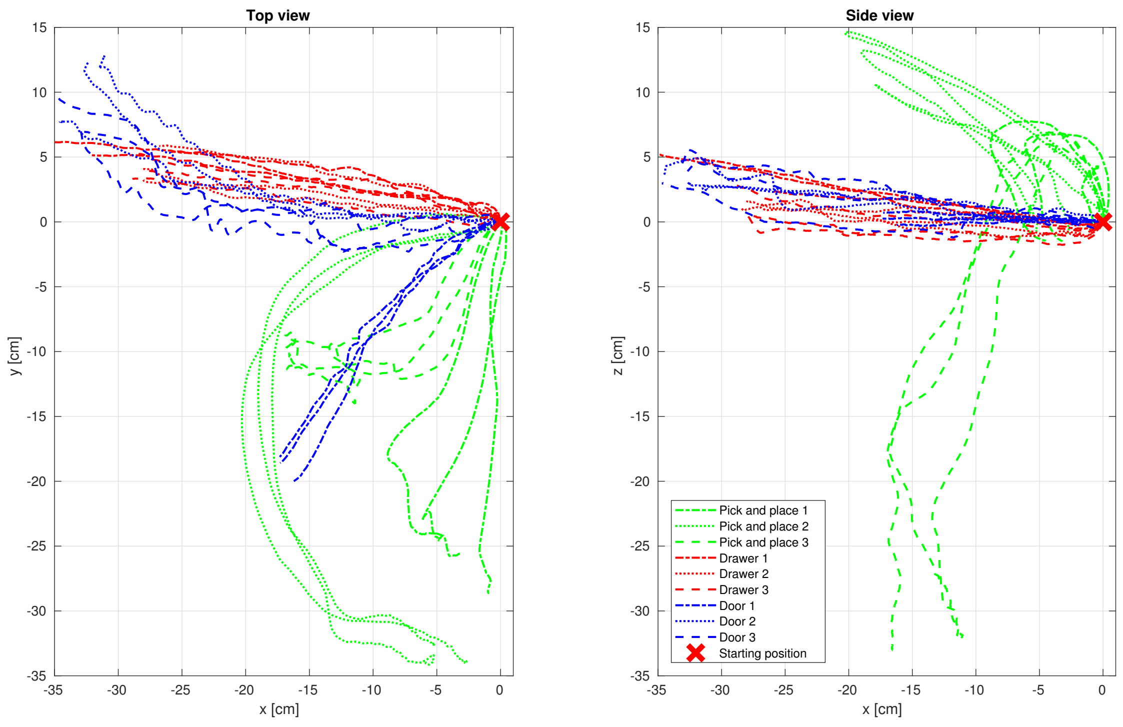


Figure B-1: A more detailed visualization of the trajectories data obtained during the manipulation tasks, as described in Section 3-2. The left figure shows the top view of the trajectories, whereas the right figure shows the trajectories from a side view. The starting point of each manipulation is set as zero.

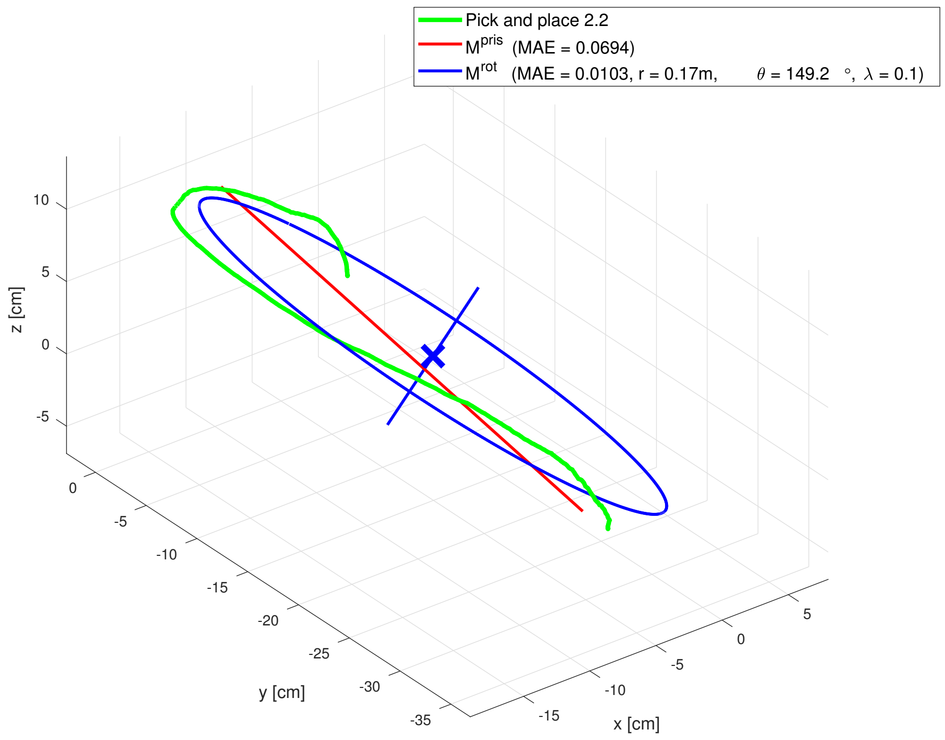


Figure B-2: Visualizing the 'Pick-and-place 2.2' trajectory and the candidate models $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ using $\lambda = 0.1$. Both $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ do not fit well. However, as $\mathcal{M}^{rot}$ has a moderate error of $10.3\,\mathrm{mm}$ the door is neither classified as a free-space nor a rotational object.

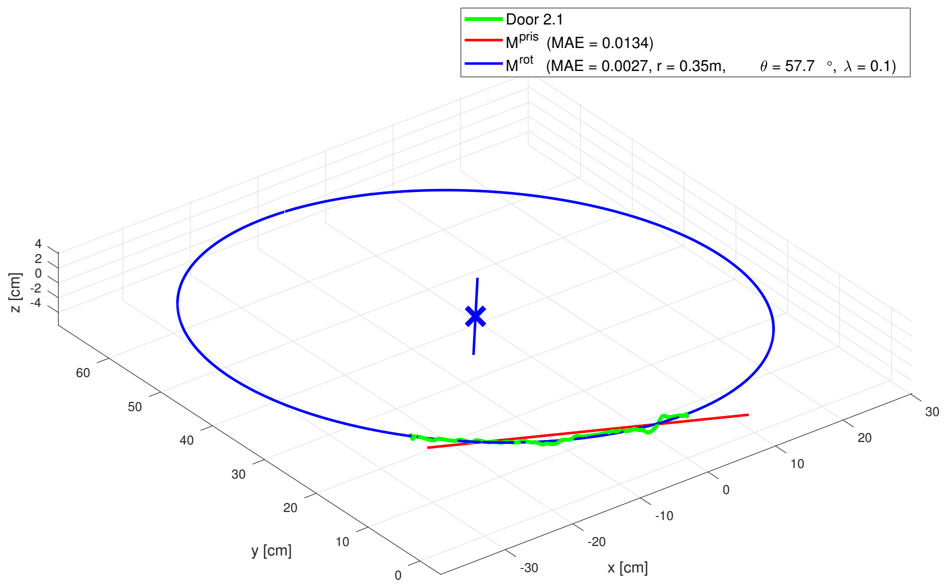


Figure B-3: Visualizing the 'Door 2.1' trajectory and the candidate models $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ using $\lambda = 0.1$. The rotational model clearly fits better (MAE$^{rot} = 2.3\,\mathrm{mm}$) than the prismatic model (MAE$^{pris} = 13.4\,\mathrm{mm}$). Consequently, the door is correctly classified as a rotational object.

Appendix C

Validation KML$_{traj}$ using the Cody dataset

The KML$_{traj}$ framework is also evaluated using the dataset provided by Sturm et al. [2]. In this dataset, the robot Cody opens: a cabinet door that opens to the right, a cabinet door that opens to the left, a dishwasher, a drawer, and a sliding cabinet door. These manipulations are all repeated 7 times, resulting in 35 trajectories, as shown in Figure C-1. The robot manipulates the objects with a hook using Equilibrium Point Control (EPC) [21], which is a form of impedance control.

Compared to the experiments performed in Chapter 3, the rotational movements are much better distinguishable from the prismatic movements. Consequently, the confidence levels are also much higher, as can be seen in Figure C-2.

Based on the confidence levels and the learning threshold, only two kinematic models are not learned. One kinematic model of the 'sliding cabinet' movement is not learned as the Mean Absolute Error (MAE) of the prismatic model is relatively high (8.20 mm). Moreover, one rotational model of the 'dishwasher' movement is not learned. When visually inspecting the model, it quickly becomes clear that the plane $\mathcal{P}$ was not correctly fitted on the trajectory. This anomaly is probably the result of the optimization function not finding the global minimum.

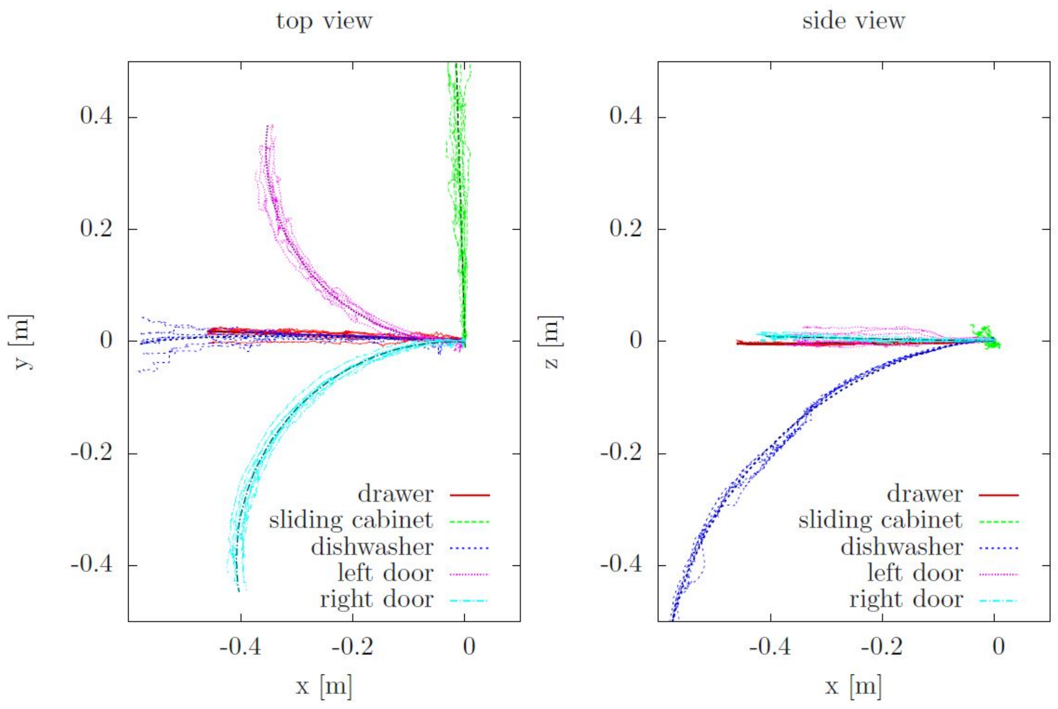


Figure C-1: Observed trajectories of robot Cody operating 5 different objects. From [2].

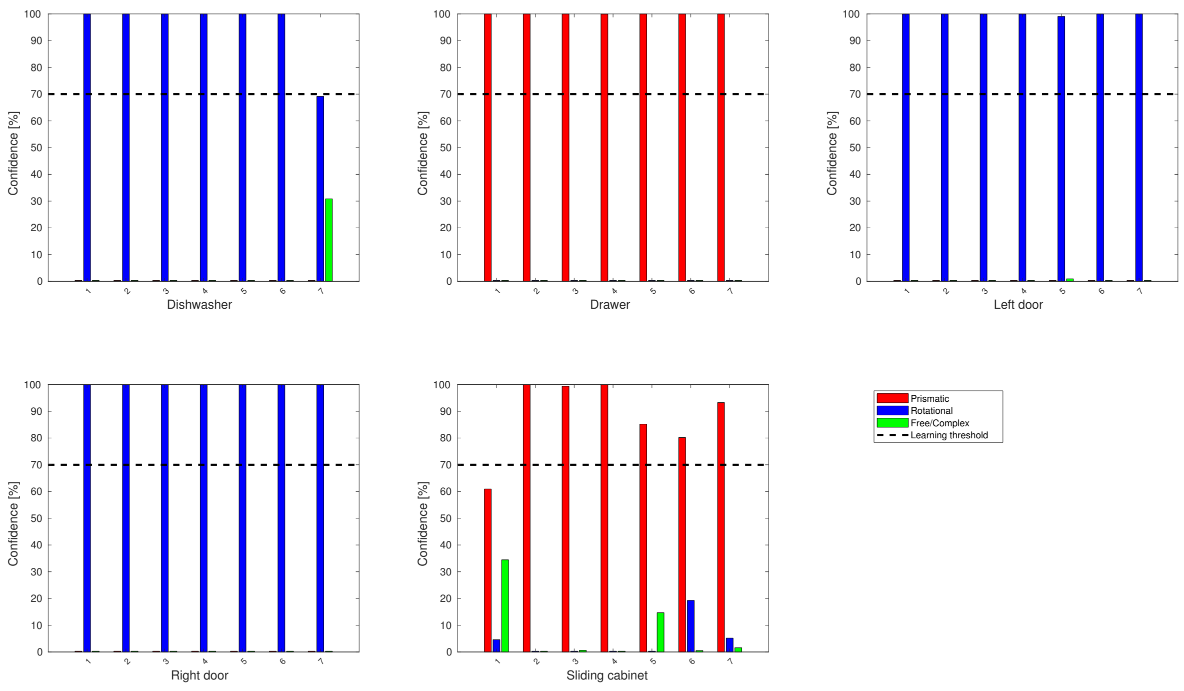


Figure C-2: After the KML$_{traj}$ framework learns the best fitting models $\mathcal{M}^{pris}$ and $\mathcal{M}^{rot}$ that using the Cody dataset [2] the confidence levels are calculated. The confidence levels of each manipulation is represented by three bars showing the confidence of the prismatic (left/red bar), rotational (middle/blue bar) and free/complex (right/green bar).

Appendix D

Tuning parameters

In total, the proposed Kinematic Model Learner (KML) framework has 12 different tuning parameters that can easily be determined by either assessing the quality of the force/torque sensor, or by assessing the general purpose of the robot application. The KML$_{traj}$ framework has 8 tuning parameters, which are summarized in Table D-1. The 4 tuning parameters of the KML$_{force}$ framework are summarized in Table D-2.

Table D-1: A summary of the tuning parameters of the KML$_{traj}$ framework.

KML$_{traj}$	**Tuning parameters**	**Purpose/meaning**
Cost function (Section 2-1-3)	λ	Weighting factor of the cost function $J(\theta)$. The weighting factor can be varied to either increase or decrease the influence of the cost function.
	θ_l, θ_h	Shaping the cost function $J(\theta)$. These values are based on the expected rotation angles of the robot application when manipulating rotational objects.
Classification (Section 2-1-4)	$\mu_{fc}, \sigma_{fc}, \mu_{pr}, \sigma_{pr}$	Shaping the probability functions that describe the likelihood that a candidate model is correct. The mu values describe the 50% likelihood point, whereas sigma determines the steepness of the curve.
	Learning threshold	The learning threshold allows robot engineers to decide how well the predicted model should fit before the manipulation robot learns a kinematic model. An kinematic model is only learned if the confidence level is above this learning threshold.

Table D-2: A summary of the tuning parameters of the KML$_{force}$ framework.

KML$_{force}$	**Tuning parameters**	**Purpose/meaning**
Segmentation (Section 2-2-2)	d_{seg}	In order to recognize partially free-space movements, the manipulation is segmented into overlapping windows of d_{seg} cm. The segmentation length d_{seg} should be smaller than the expected length of the free-space movement.
Classification (Section 2-2-3)	μ_{max}, σ_{max}, $\mu_{z,max}$	Shaping the decision boundaries used for the binary classifier of the KML$_{force}$ framework. These values can be determined by assessing the drift and the noise of the f/t sensor and by determining the minimal weight of the objects that should be detected.

Bibliography

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] J. Sturm, A. Jain, C. Stachniss, C. Kemp, and W. Burgard, "Operating Articulated Objects Based on Experience," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Georgia Institute of Technology, 2010, pp. 2739–2744.

[3] R. Martín-Martín and O. Brock, "Building Kinematic and Dynamic Models of Articulated Objects with Multi-Modal Interactive Perception," *AAAI Symposium on Interactive Multi-Sensory Object Perception for Embodied Agents*, no. March, 2017.

[4] United Nations, "World Population Prospects - The 2017 Revision - Key Findings and Advance Tables," 2017.

[5] European Commission, "Horizon 2020 - The Framework Programme for Research and Innovation," 2011.

[6] M. van Osch, D. Bera, K. van Hee, Y. Koks, and H. Zeegers, "Tele-operated service robots: ROSE," *Automation in Construction*, vol. 39, pp. 152–160, 2014.

[7] R. Martin Martin and O. Brock, "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors," *IEEE International Conference on Intelligent Robots and Systems*, no. September 2014, pp. 2494–2501, 2014.

[8] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, and E. Berger, "Autonomous door opening and plugging in with a personal robot," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 729–736, 2010.

[9] M. Beetz, M. Lorenz, and M. Tenorth, "CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1012–1017.

[10] S. Chitta, B. Cohen, and M. Likhachev, "Planning for autonomous door opening with a mobile manipulator," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1799–1806, 2010.

[11] J. Sturm, C. Stachniss, and W. Burgard, "A Probabilistic Framework for Learning Kinematic Models of Articulated Objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, 2011.

[12] K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme, "Active articulation model estimation through interactive perception," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 3305–3312, 2015.

[13] G. Subramani, M. Zinn, and M. Gleicher, "Inferring geometric constraints in human demonstrations," *arXiv preprint arXiv:1810.00140*, no. CoRL, pp. 1–14, 2018.

[14] D. Katz, A. Orthey, and O. Brock, "Interactive perception of articulated objects," in *Experimental Robotics.* Springer, 2014, pp. 301–315.

[15] K. Hausman, F. Balint-Benczedi, D. Pangercic, Z.-C. Marton, R. Ueda, K. Okada, and M. Beetz, "Tracking-based interactive segmentation of textureless objects," *2013 IEEE International Conference on Robotics and Automation*, pp. 1122–1129, 2013.

[16] S. Pillai, M. R. Walter, and S. Teller, "Learning Articulated Motions From Visual Demonstration," *arXiv preprint arXiv:1502.01659*, 2015.

[17] J. Sturm, C. Stachniss, V. Pradeep, C. Plagemann, K. Konolige, and W. Burgard, "Learning Kinematic Models for Articulated Objects," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2009, pp. 1851–1856.

[18] J. Sturm, C. Stachniss, V. Predeap, C. Plagemann, K. Konolige, and W. Burgard, "Towards Understanding Articulated Objects," *Workshop Integrating Mobility and Manipulation at Robotics Science and Systems RSS*, 2009.

[19] J. Sturm, "Approaches to Probabilistic Model Learning for Mobile Manipulation Robots," *Springer*, vol. 89, 2013.

[20] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[21] A. Jain and C. C. Kemp, "Improving robot manipulation with data-driven object-centric models of everyday forces," *Autonomous Robots*, vol. 35, no. 2-3, pp. 143–159, 2013.

[22] A. Jain, H. Nguyen, M. Rath, J. Okerman, and C. C. Kemp, "The complex structure of simple devices: A survey of trajectories and forces that open doors and drawers," *2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2010*, no. October 2010, pp. 184–190, 2010.

[23] A. Jain and C. C. Kemp, "Pulling open novel doors and drawers with equilibrium point control," *2009 9th IEEE-RAS International Conference on Humanoid Robots, Humanoids*, pp. 498–505, 2009.

[24] R. Diankov, S. S. Srinivasa, D. Ferguson, and J. Kuffner, “Manipulation planning with caging grasps,” *2008 8th IEEE-RAS International Conference on Humanoid Robots, Humanoids*, pp. 285–292, 2008.

[25] D. G. Luenberger, *Optimization by vector space methods.* John Wiley & Sons, 1997.

[26] W. Gander, M. J. Gander, and F. Kwok, *Scientific Computing - An Introduction using Maple and MATLAB.* Springer Science & Business, 2014, vol. 11.

[27] C. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Climate Research*, vol. 30, pp. 79–82, 2005.

[28] M. Stilman, “Gravity and Drift in Force/Torque Measurements,” Georgia Institue of Technology, Tech. Rep., 2014.

[29] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis, “Coactive Design: Designing Support for Interdependence in Joint Activity,” *Journal of Human-Robot Interaction*, vol. 3, no. 1, p. 43, 2014.

[30] G. A. van der Hart, “Improving task performance of an operator performing a domestic tele-manipulation task by providing haptic feedback - Literature Report, TU Delft,” Tech. Rep., 2016.

Glossary

List of Acronyms

DOF	Degrees of Freedom
HiT	Heemskerk Innovative Technology
ADL	Activities of Daily Living
LfD	Learning from Demonstration
COM	center of mass
RMSE	Root-Mean-Square Error
BIC	Bayesian information criteria
CEP	Cartesian Equilibrium Point
DOF	Degree of Freedom
KML	Kinematic Model Learner
PaP	pick-and-place
CDF	cumulative distribution function
MAE	Mean Absolute Error
EPC	Equilibrium Point Control
SVD	Singular Value Decomposition

List of Symbols

α	Parameter vector which describes the parameters of a kinematic model
μ	Mean/tuning parameter
Ψ_W	World frame
$\Psi_{f/t}$	Force/torque sensor frame
σ	Standard deviation/tuning parameter
θ	Rotation angle
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and standard deviation σ
$\mathcal{P}$	A plane that best fits the observed trajectory points in terms of having the lowest sum of squared residual errors.
$\mathbf{x}$	The observed end-effector trajectory of a manipulation.
C	Confidence level
$F_{\mu,\sigma}$	A cumulative distribution function based on a normal distribution $\mathcal{N}(\mu, \sigma^2)$
$f_{\mu,\sigma}$	The probability density function of a normal distribution $\mathcal{N}(\mu, \sigma^2)$
$m_{gripper}$	The mass of the gripper attached to the f/t sensor
m_{obj}	The mass of an object.
q	Scaling factor
X	Random variable
z	A projection of the observed trajectory x onto a plane $\mathcal{P}$
$\mathcal{M}^{complex}$	Kinematic model of a complex object
$\mathcal{M}^{pris}$	Kinematic model of a prismatic object
$\mathcal{M}^{rigid}$	Kinematic model of a rigid object
$\mathcal{M}^{rot}$	Kinematic model of a rotational object