Combining Human Demonstrations and Motion Planning for Movement Primitive Optimization

Kombination von menschlichen Bewegungen und Bewegungsplanung zur Optimierung von Bewegungsprimitiven

Master-Thesis von Dorothea Koert

Tag der Einreichung:

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Guilherme Maeda, Rudolf Lioutikov

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science
Institute for
Intelligent Autonomous Systems

Combining Human Demonstrations and Motion Planning for Movement Primitive Optimization
Kombination von menschlichen Bewegungen und Bewegungsplanung zur Optimierung von Bewegungsprimitiven

Vorgelegte Master-Thesis von Dorothea Koert

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Guilherme Maeda, Rudolf Lioutikov

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 31.03.2016

_____

(D.Koert)

Abstract

The ability to generalize robot motions to different workspace settings is an important requirement for a future robotic coworker. In particular, extracting those motions out of human demonstrations provides intuitive ways for people inexperienced in robotics to teach new tasks to a robot. This thesis presents a step towards flexible robot coworkers that are capable of incorporating human demonstrations to generalize robot movement primitives.
The thesis proposes to combine trajectory optimization and human demonstrations to generate human-like robot motions that adapt to both unforeseen obstacles and changes in the workspace. Specifically, this thesis presents an approach to connect core ideas found in Stochastic Trajectory Optimization to the existing framework of Probabilistic Movement Primitives (ProMPs). The proposed algorithm extends Relative Entropy Policy Search (REPS) by using a Kullback-Leibler (KL) metric to minimize the deviation from the demonstrations and an signed distance field to avoid obstacles. In contrast to most common motion planning algorithms the variance of the human demonstrations is preserved during the optimization process, seamlessly connecting and preserving properties of ProMPs.
The method is evaluated on a number of examples of redrawing handwritten letters, as well as on a pick-and-place application for a KUKA lightweight arm with 7 degrees of freedom. The experiments show that the method is able to compute valid solutions in both task and joint space. The results indicate that the algorithm is able to compute Probabilistic Movement Primitives that avoid obstacles not present during the demonstrations.

Keywords: Imitation Learning, Motion Planning, Trajectory Optimization, Probabilistic Movement Primitives

Zusammenfassung

Die Fähigkeit eines Roboters, Bewegungen an variable Arbeitsumgebungen anzupassen, stellt eine Schlüsselqualifikation für zukünftige Anwendungen dar. Im Umgang mit Robotern unerfahrenen Menschen, bietet "Lernen durch Imitation" eine intuitive Möglichkeit, einem Roboter neue Aufgaben beizubringen.
Durch die Verarbeitung menschlicher Bewegungen zu generalisierbaren Bewegungsprimitiven trägt diese Thesis zur Entwicklung flexibler Mensch-Roboter Interaktion in verschiedenen Arbeitsräumen bei. Der vorgestellte Ansatz verbindet menschliche Demonstrationen mit einer Trajektorienoptimierung. Der Algorithmus erzeugt generalisierbare Bewegungsmuster für Roboter, die sich sowohl an unvorhergesehene Hindernisse als auch an Veränderungen im Arbeitsumfeld anpassen.
Die Thesis verknüpft Grundkonzepte stochastischer Trajektorienoptimierung mit dem bestehenden Framework von Probabilistischen Bewegungsprimitiven (ProMPs). Im vorgestellten Algorithmus wird eine Kullback-Leibler (KL) Metrik genutzt, um Relative Entropy Policy Search (REPS) um eine Minimierung der Abweichung zu menschlichen Demonstrationen und euklidischen Abständen zu Hindernissen zu erweitern. Im Gegensatz zu anderen Bewegungsplanungsalgorithmen bleibt die Varianz in den menschlichen Demonstrationen über die Optimierung hinaus erhalten.
Der Algorithmus wurde sowohl in Versuchen zur Reproduktion handgeschriebener Buchstaben, als auch in einem pick-and-place Szenario auf einem 7-gelenkigen KUKA Roboterarm evaluiert. Die Optimierung berechnet zulässige Lösungen im Kartesischen- und im Gelenkwinkelraum. Die Experimente haben gezeigt, dass die Methode in der Lage ist ProMPs unter der Vermeidung zusätzlicher Hindernisse zu berechnen.

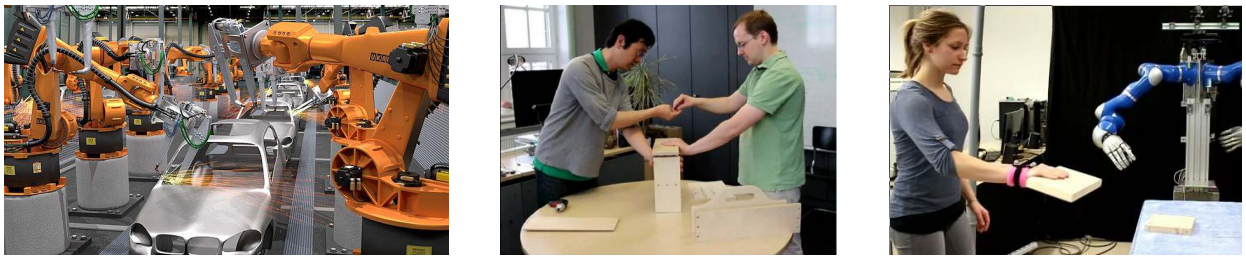
Schlüsselwörter: Lernen durch Imitation, Bewegungsplanung, Trajektorienoptimierung, Probabilistische Bewegungsprimitive

Contents

1 Introduction

1.1 Motivation

Robot coworkers are a central piece in a growing field of applications that demand collaboration between humans and supportive machines. In particular, human-robot collaboration imposes additional requirements compared to standard robotic applications, where robots usually act isolated from humans. As a main difference robots will more frequently interact with humans who are inexperienced with robotics. Those could be elderly people for whom robots could provide support in an independent household keeping or factory workers who need support in specific tasks. Figure 1.1 illustrates this shift of application areas from factories to collaborative settings.
One main critical part of connecting robots to such new applications is to make people accept their robotic co-worker and to make them feel comfortable and motivated to include the robot in their everyday routines. In a cooperative work, both partners should be able to anticipate each others motions and intentions. As humans are mostly used to work with a human partner, they tend to expect a humanoid robotic co-worker to act similar to a human. Recent research reveals that humans start to anthropomorphize moving robots [1], in particular when their appearance changes from machine-like factory robots to highly humanely body shapes [2]. This considerations motivate research towards human-like motions of humanoid robots.
A core question is how non-robot experts could teach robots the required abilities for a certain task and how one can obtain human-like robotic movements out of human demonstrations. While methods such as kinesthetic teaching can be time-consuming and sometimes even infeasible for inexperienced workers and non-backdriveable robots, a very natural and intuitive way of teaching is to learn directly from human demonstrations. Those should ideally be demonstrations where the human movement is not influenced or constrained in any sense by the existence of a robot learner. As human apprentices learn their part of a task by watching experienced workers, it seems sensible to let a robot learn from watching humans executing a specific task. Figure 1.1 (b) illustrates such a demonstration in a collaborative setting.



(a) (b) (c)

Figure 1.1: (a) In common applications robots act isolated from humans. Therefore, their motion is mainly constrained and optimized for a specific task such as an assembly of car parts.[1] (b,c) When robots start to come in direct contact with humans they should adapt their motions according to human demonstrations in order to remain predictable coworkers.

Another issue is how to enable the robots to generalize the given human demonstrations to react on changes in the environment. In this case, capturing the variance in the human motions can provide multiple solutions to one specific planning problem. Paraschos et al. introduced Probabilistic Movement Primitives [3] in order to ensure both, generalization in terms of endpoint or via point adaptation and representation of variance in the motions. However, it remains a main drawback of Probabilistic Movement Primitives that they do not directly deal with motion planning problems, such as obstacles in the robot's workspace, which might have not been present in the demonstrations.
Motion planning has been a topic of research for many years, and many existing algorithms mainly focus on robot

[1] http://www.technovalves.in/industries-we-serve/ (15.2.2016)

issues such as feasibility, joint constraints or energy minimization. Differently, this thesis incorporates human demonstrations and concentrates on finding a way to solve motion planning problems with respect to human-like motions. The aim is to connect motion planning to the framework of probabilistic movement primitives to obtain generalizable building blocks for robot motions.
To force smooth solutions, stochastic optimization methods such as STOMP and Pi2 rely on additional mechanisms and prior assumptions on the solution. By using a KL metric to minimize the divergence with the distribution of the demonstrations, the proposed algorithm eliminates the need for such prior knowledge in the optimization process and obtains all required information out of the given demonstrations. The method incorporates human demonstrations and Probabilistic Movement Primitives inside a motion planning algorithm to contribute to the goal of achieving natural human motions for humanoid robots. In particular, the proposed algorithm is able to deal with variance in the demonstrations and is suited for robot co-workers as the resulting robot motion is directly adapted from humans.

1.2 Overview of Sections

This section provides an overview of the structure of this thesis.

Chapter 2 introduces the basic mathematical concepts and related work. It explains aspects of information theory, stochastic optimization and relative entropy policy search and continues with an overview of recent work done in the fields of motion planning, learning from demonstrations, human robot collaboration and movement primitives.

Chapter 3 presents the proposed algorithm and states the resulting optimization problem. A closed form solution is derived, following the Lagrangian method. In the second part, basic properties of the algorithm and results obtained on toy examples of drawing a letter with a planar two link robot are discussed.

Chapter 4 continues by connecting the developed algorithm to the framework of probabilistic movement primitives and presents results from experiments using a 7 degree-of-freedom KUKA lightweight arm. The basic software architecture used for the experiments and the core hardware components are introduced and the chapter evaluates the results obtained in the experiments.

Chapter 5 summarizes the results of the approach and discusses directions for investigation as future work. The chapter resumes the main advantages of the proposed method and lists possible ideas for further improvement.

2 Foundations and Related Work

This chapter provides an overview of related work. It points out the basic concepts of **Information Theory**, **Stochastic Optimization** and **Relative Entropy Policy Search** which form the foundation of the proposed algorithm. An introduction to **Motion Planning**, **Learning from Demonstrations** and **Human Robot Collaboration** and common algorithms from those areas are presented. The chapter concludes with references to the frameworks of **Dynamic and Probabilistic Movement Primitives** as basic building blocks for complex motions.

2.1 Information Theory and KL Divergence

Information theory was introduced in 1948 by Shannon and proved to be a powerful mathematics tool with deep connections not only to communication and control, but also to probability theory and statistical inference [4]. One of its basic concepts is given by the definition of entropy for a distribution or a random variable. The Shannon entropy of a variable X is given by

$$H[X] = -\sum_x P(X = x)\log(P(X = x)) = -\mathbb{E}[\log(P(X))]. \tag{2.1}$$

This equation is an interpretation of the variability of X as a measure of how uncertain one is about X's value.
The idea of entropy in information theory provides means to compare distributions [5]. For two probability distributions, p and q, the relative entropy of p with respect to q is called Kullback-Leibler (KL) divergence of p from q. For discrete samples it is given by

$$D(p \parallel q) = \sum_{x \in X} p(x) \log\left(\frac{p(x)}{q(x)}\right). \tag{2.2}$$

Equation 2.2 expresses the amount of information loss when p is used to approximate q. In most applications q represents the true underlying distribution of data and observations and p is the model that is used to approximate q. The KL divergence has some properties of a metric on the space of probability distribution: it is non-negative, with equality only when the two distributions are equal, but it is not symmetric, and it does not obey the triangle inequality. For the case of two multivariate Gaussians $\mathcal{N}_0$ and $\mathcal{N}_1$ it is given in closed form by

$$D(\mathcal{N}_0 \parallel \mathcal{N}_1) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - \delta + \ln\left(\frac{|\,\Sigma_1\,|}{|\,\Sigma_0\,|}\right)\right], \tag{2.3}$$

where μ_0 and μ_1 denote the means, Σ_0 and Σ_1 denote the covariances and δ denotes the dimensionality.
In this thesis the KL divergence is used to determine the deviation of the robot's current policy from the distribution of human demonstrations.

2.2 Stochastic Optimization and Relative Entropy Policy Search

Stochastic optimization provides means to optimize a system's cost (or reward) under the inherent presence of noise and uncertainty [6]. It is, therefore, suitable for optimizing the policies obtained from human demonstrations. The core problem is defined by finding a vector $\boldsymbol{\Theta}^*$ out of the allowed set Γ that minimizes some loss function $\Lambda(\boldsymbol{\Theta})$

$$\boldsymbol{\Theta}^* \equiv \underset{\boldsymbol{\Theta}\in\Gamma}{\mathrm{argmin}}\ \Lambda(\boldsymbol{\Theta}) = \{\boldsymbol{\Theta}^* \in \Gamma : \Lambda(\boldsymbol{\Theta}^*) \leq \Lambda(\boldsymbol{\Theta}) \quad \text{for all} \quad \boldsymbol{\Theta} \in \Gamma\}. \tag{2.4}$$

Any maximization of a reward function can be transformed in this problem by simply switching the sign. In general, stochastic algorithms apply noise in the measurement $\Lambda(\boldsymbol{\Theta})$ or include random decisions in the search direction during the optimization process. In contrast deterministic optimization presumes availability of perfect information about the loss function and its derivatives and computes the search direction in a deterministic manner for each time step. Common examples for stochastic optimization are random search [7], stochastic approximation [8] and

genetic algorithms [9].
The principle ideas of stochastic optimization can also find applications in the field of policy search. Policy search is a reinforcement learning technique that iteratively improves the agents policy by incorporating new knowledge gained from e.g. past trials. In particular Peters et al. proposed a method to additionally avoid information loss during the policy improvements by constraining the difference between two updates [10]. They introduce Relative Entropy Policy Search (REPS) as a policy gradient approach with a KL bounded update step. The basic problem statement is given by

$$\max_{\pi} \int \pi(\boldsymbol{\Theta}) R(\boldsymbol{\Theta}) d\boldsymbol{\Theta} \tag{2.5}$$
$$\text{s.t.} \quad \int \pi(\boldsymbol{\Theta}) \log\left(\frac{\pi(\boldsymbol{\Theta})}{q(\boldsymbol{\Theta})}\right) \mathrm{d}\boldsymbol{\Theta} \leq \varepsilon, \qquad \int \pi(\boldsymbol{\Theta}) d\boldsymbol{\Theta} = 1,$$

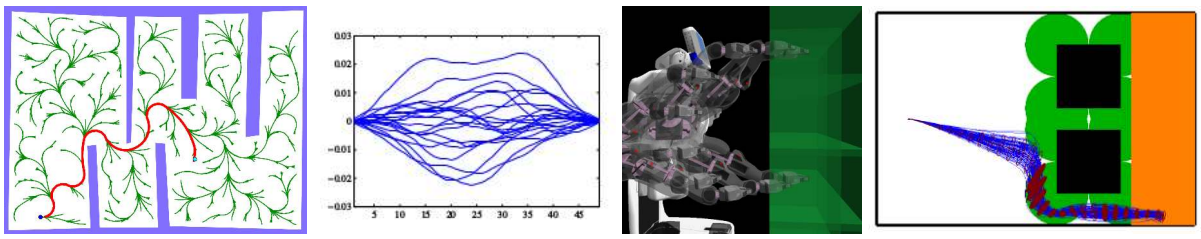
where $\pi(\boldsymbol{\Theta})$ denotes the new policy, $q(\boldsymbol{\Theta})$ denotes the old policy, $\mathscr{R}(\boldsymbol{\Theta})$ denotes a reward function and ε is the upper bound on the KL divergence between the old an the new policy. Using the method of Lagrangian multipliers this constrained optimization problem can be solved efficiently and in closed form solution:

$$\pi(\boldsymbol{\Theta}) \propto q(\boldsymbol{\Theta}) \exp\left(\frac{R(\boldsymbol{\Theta})}{\eta}\right), \tag{2.6}$$

where η denotes the Lagrangian multiplier of the KL-bound constraint. During a stochastic optimization process a number of noisy roll outs can be used to approximate the integrals and provide the maximum likelihood update for parameters that describe the policy.
This thesis presents an extension of REPS to additionally incorporate the distance between the distributions of demonstrations and the current policy.

2.3 Motion Planning

Path planning and trajectory optimization are a long known field of research in robotics and have been studied extensively. The basic problem statement is to find a solution for driving a robot from an initial configuration to a goal configuration, while obeying certain constraints and avoiding obstacles in the robot‘s environment [11]. In order to optimize trajectories sampling based motion planners such as probabilistic roadmaps [12], rapidly exploring random trees [13] or stochastic trajectory optimization [14] have shown to be sufficient also in high dimensional space and on several robot applications [15]. Figure 2.1(a) illustrates a solution for path planning with RRT.



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

Figure 2.1: (a) Rapidly Random Tree for path planning [1](b,c) Stochastic trajectory optimization [14] (d) Solution from belief state planning including variance [21]

Common to all these approaches is the fact that human demonstrations are not taken into account into the optimization process. Some approaches try to constraint the solutions to a certain type of motion, e.g. human-like or smooth, by adding additional assumptions into their optimization such as smoothness, energy minimization or time efficiency [14]. Finding the trade off between those different cost objectives often becomes a crucial hand tuning problem. Incorporating human demonstrations into the motion planning approach offers a way of avoiding those artificial assumptions. Extracting information out of few human trajectories for motion planning and trajectory

[1] http://mrs.felk.cvut.cz/research/motion-planning (30.2.2016)

optimization in particular has been used for different applications such as car navigation on a parking lot or helicopter control [16], [17], [18]. In those cases the demonstrations were used to extract the weighting of predefined cost terms. In [19] Ye and Alterovitz present a framework to extract certain constraints out of motions based on the variance in the motions. However, all of these approaches use the distribution of the demonstrations as an input but reduce it to a single trajectory at the output as they often only use the mean in the optimization. The idea of including variance inside the motion planning algorithm is also known in belief state planning [20],[21] as illustrated in fig 2.1(d). Those algorithms also offer a solution that does not only consist of a single trajectory but of a distribution with a certain variance depending on the current belief state. However, those solutions do not take human demonstrations into account during their optimization. Another way to preserve the variance during the optimization process is provided by using the Kullback-Leibler divergence as a metric to stay close to demonstrations [22]. This thesis connects this idea of using a KL metric to mimic human demonstrations with the main structure of stochastic trajectory optimization [14].

2.4 Learning from Demonstrations

The idea of learning from demonstration has been a recent topic of research [23]. According to Argall, et al. [24], learning from demonstrations can be categorized by two main criteria: record mapping and embodiment mapping. Figure 2.2(a) illustrates the basic concept of this categorization [24]. The first regards the recording of the teacher's execution, and the second regards how the student, i.e. the robot, executes the recorded motion with its own embodiment. The choice of the record mapping determines how the motion of the demonstrator is captured during his/her demonstration and which information is handed to the robot later.

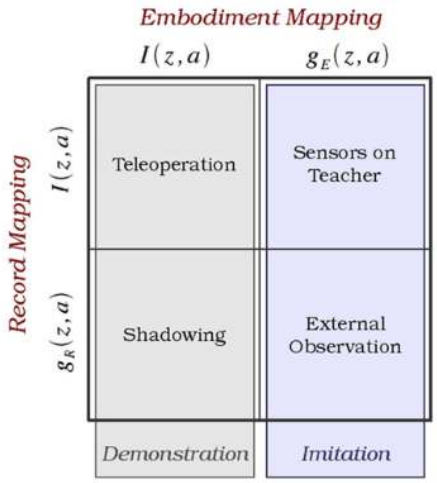
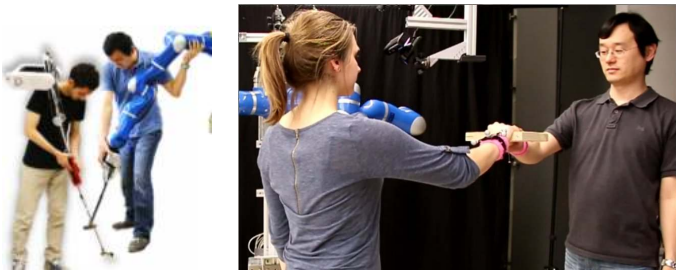


Figure 2.2: Learning from demonstrations can be categorized in record and embodiment mapping [24].

In the case of humanoid robots the question often boils down to the number of joints to track and how to extract them from the given demonstration. One method of obtaining those demonstrations is given by kinesthetic teaching as shown in figure 2.3(a).
As a drawback kinesthetic teaching tends to constrain the demonstrator's movement and might not even be feasible depending on the robot model or the task. On the other hand, a very natural and intuitive way of teaching is learning directly from unconstrained natural human demonstrations. Several approaches concentrate on joint to joint mapping while recording either the skeleton with Kinect camera such as Luo [25] or a marker tracking approach [26]. However, one remaining problem with joint to joint mapping is that this approach constrains the robot's solution and in certain tasks, it is not even important to imitate the whole movement but to reach or track a certain end-effector position during the execution.
An alternative approach is given by recording only the human end-effector position during the demonstration and solve the correspondence problem by searching the joint positions through inverse kinematics [27], [28], [29]. Figure 2.3(b) illustrates such a scenario where the end-effector positions of two coworkers are recorded using a



(a) (b)

Figure 2.3: (a) Using kinesthetic teaching tends to restrict the motions during the demonstrations. (b) In opposite, recording only the end-effector position with a motion capturing system provides freedom to the demonstrators and results in more natural human motions.

motion capturing system.
In order to post process the demonstrations and learn an optimal robot behaviour several methods exist such as reinforcement learning or extraction of movement primitives. The idea of movement primitives is explained more detailed in section 2.5.

2.5 Movement Primitives

Movement primitives enable the decomposition of complex movements to compact parametrization of robot policies [30]. In this concept a core idea is to encode a recorded trajectory in a way that can be used to generate different variations of the original movement in temporal as well as in spatial context.

2.5.1 Dynamic Movement Primitives

Dynamic Movement Primitives [30] follow the basic idea of a PD-control signal that is extended by a non-linear forcing function. The basic equations are given by

$$\ddot{y} = \tau^2 \alpha_y \left(\beta_y (g - y) \frac{\dot{y}}{\tau} \right) + \tau^2 f(z); \qquad \dot{z} = -\tau \alpha_z z. \tag{2.7}$$

In this equation y, $\dot{y}$ and $\ddot{y}$ denote the desired position, velocity and acceleration of a desired variable and α_y and β_y can be seen as parameters of the PD-controller. The desired goal position is given by g and a time-constant τ defines the speed of the execution together with the phase variable z. The shape of the movement is encoded in the forcing term $f(z)$, which consists of a certain number of basis-functions $\phi_i(z)$, shape-parameters w_i and a phase-variable z

$$f(z) = \frac{\sum_{i=1}^{K} \phi_i(z) w_i}{\sum_{i=1}^{K} \phi_i}; \qquad \phi_i(z) = \exp\left(-\frac{1}{2\sigma_i^2} (z - c_i)^2 \right) \qquad i = 1...K, \tag{2.8}$$

where K is the number of basis functions, c_i defines their centres and σ_i their widths.
As time goes towards infinity this forcing term vanishes due to the decrease of z which guarantees the stability of the system. A DMP can be learned out of given demonstration by linear regression. Different frameworks allow for goal dependent modulation, time scaling, via point traversal or adaptation to the environment by using distance fields [31], [32].

2.5.2 Probabilistic Movement Primitives

To capture variability in motions a probabilistic framework was introduced by Paraschos et. al. [3]. In this framework a Probabilistic Movement Primitive (ProMP) is used to describe multiple ways to execute a motion, resulting in a probability distribution over trajectories.
In order to learn a ProMP the given demonstration trajectories are modelled using a set of basis functions Φ_t and a weight vector w

$$\boldsymbol{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \boldsymbol{\Phi}_t \boldsymbol{w} + \boldsymbol{\epsilon}_y; \qquad \text{with } \boldsymbol{\epsilon}_y \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_y). \tag{2.9}$$

In this equation $\Phi_t = [\phi_t, \dot{\phi}_t]$ denotes the time-dependent basis matrix positions q_t and velocities $\dot{q}_t$ and ϵ_y defines the error of the trajectory representation given as zero-mean i.i.d. Gaussian noise.
The shape of the trajectory is defined by weighting the basis functions with a vector $\mathbf{w}$. This weight vector is computed by linear ridge regression:

$$\boldsymbol{w} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}_t^T \boldsymbol{y}, \tag{2.10}$$

where λ denotes a small factor.
Similar to DMPs, a phase variable z is introduced to decouple the movement from the time signal and to incorporate temporal scaling. The phase $z(t)$ can be given by any arbitrary function that is monotonically increasing with time. Using this phase variable the components of the matrix $\boldsymbol{\Phi} = [\phi_t, \dot{\phi}_t]$ are defined by

$$\phi_t = \phi(z_t); \qquad \dot{\phi}_t = \phi\prime(z_t)\dot{z}_t; \qquad \phi_i(z_t) = \frac{b_i(z)}{\sum_{j=1}^{K} b_j(z)} \qquad i = 1...K. \tag{2.11}$$

The choice of the basis functions $b_i(z)$ can vary depending on the type of the motion. E.g. [3] proposes Gaussian basis functions b_i^{G} for stroke-based movements and Von-Mises basis functions b_i^{VM} for rhythmic movements

$$b_i^{\mathrm{G}}(z) = \exp\left(-\frac{(z_t - c_i)^2}{2h}\right); \qquad b_i^{\mathrm{VM}}(z) = \exp\left(-\frac{\cos(2\pi(z_t - c_i))}{h}\right), \tag{2.12}$$

where h represents the width and c_i the centres of the basis functions.
Using this basis-functions and the learned weight vector the distribution of $\boldsymbol{y}$ at time t can be written as

$$p(\boldsymbol{y}_t|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Psi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_y). \tag{2.13}$$

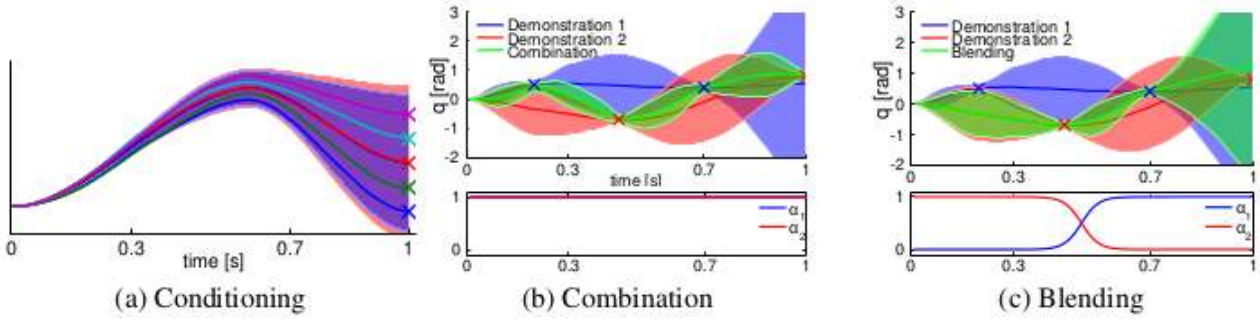
Subsequently, a Gaussian distribution for $p(w|\Theta) = \mathcal{N}(w|\mu_w, \Sigma_w)$ over the parameters w is assumed where Θ denotes the mean μ_w and variance Σ_w of the trajectory-weights w of the demonstrations. By taking the integral over w the trajectory weights can be marginalized out which leads to the distribution of the state $p(y_t|\Theta)$ for time step t

$$p(\boldsymbol{y}|\boldsymbol{\Theta}) = \int p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{\Theta}) d\boldsymbol{w} = \int p(\boldsymbol{y}|\boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{\Theta}) d\boldsymbol{w} \tag{2.14}$$

$$p(\boldsymbol{y}_t|\boldsymbol{\Theta}) = \int \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Psi}_t^T \boldsymbol{w}, \boldsymbol{\Sigma}_y) \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) d\boldsymbol{w} = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Psi}_t^T \boldsymbol{\mu}_w, \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t + \boldsymbol{\Sigma}_y). \tag{2.15}$$

At last the parameters $\boldsymbol{\Theta} = [\mu_w, \Sigma_w]$ of the time dependent distribution remain to be learned, e.g. by using a maximum likelihood estimate. At the end of this learning process the ProMP defines the movement by mean and variance at each time point.

Figure 2.4 illustrates adaptation to unseen situations such as a new goal state or via point.



(a) Conditioning (b) Combination (c) Blending

Figure 2.4: Different applications for the ProMP framework are given by conditioning, combination and blending [3].

This can be obtained using conditioning on a certain state $\boldsymbol{y}_t^*$ at time point t. Therefore, additional an observation $\boldsymbol{x}_t^* = (\boldsymbol{y}_t^*, \boldsymbol{\Sigma}_y^*)$ is added to the probabilistic model by applying Bayes theorem

$$p(w|x_t^*) \propto \mathcal{N}(y_t^*|\Psi_t w, \Sigma_y^*)p(w). \tag{2.16}$$

For Gaussian trajectory distribution again a Gaussian distribution is obtained given by

$$\boldsymbol{L} = \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t (\boldsymbol{\Sigma}_y^* + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_t)^{-1} \tag{2.17}$$

$$\boldsymbol{\mu}_w^{\text{new}} = \boldsymbol{\mu}_w + \boldsymbol{L}(\boldsymbol{y}_t^* - \boldsymbol{\Psi}_t^T \boldsymbol{\mu}_w) \tag{2.18}$$

$$\boldsymbol{\Sigma}_w^{\text{new}} = \boldsymbol{\Sigma}_w - \boldsymbol{L}\boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_w. \tag{2.19}$$

Although ProMPs offer advantages of capturing and preserving variance in human motions they are not yet applicable for unforeseen changes in the workspace settings, and therefore, the principles for addressing obstacle avoidance are unclear. This thesis proposes a way to obtain a ProMP from human demonstrations that is additionally able to adapt to changes in workspace settings and to avoid obstacles during the execution.

3 From Human Motions to Demonstration-Guided Trajectory Optimization

This chapter presents a trajectory generation method to generate human-like motions, based on natural human demonstrations. Additionally the proposed method is able to react on changes in the environment such as obstacles. A distribution d is extracted out of a set of demonstrated trajectories, and the current policy q is optimized to stay close to those demonstrations using the KL divergence, while at the same time avoiding obstacles via trajectory optimization.

3.1 Algorithm

Here, a method is presented to capture arbitrary distributions of human demonstrations, including their variance, and adapt those demonstrations to new settings, including additional obstacles and via points. The method leverages on trajectory optimization based on Relative Entropy Policy Search [10] and Stochastic Trajectory Optimization [14]. The method optimizes trajectories under constraints, while at the same time preserving as much of the original distribution as possible without the need of hand-tuned costs. This section presents the underlying optimization problem and illustrates the main structure of the developed algorithm. The method can work on arbitrary distributions, even though in this thesis it is only applied for the Gaussian case. In the second part of this section the algorithm is applied on a number of toy examples.

3.1.1 Trajectory Optimization

Given robot trajectory $\boldsymbol{x} = \{x^{[1]}, ..., x^{[T]}\}$, where $x^{[t]}$ denotes Cartesian or joint state at time step t, the optimization addresses two main objectives. The first is given by staying close to the human demonstrations. This is encoded by the Kullback Leibler (KL) divergence between the distribution of the current policy p and the distribution of the human demonstrations d. This deviation from the demonstrations forms the first part of the cost function

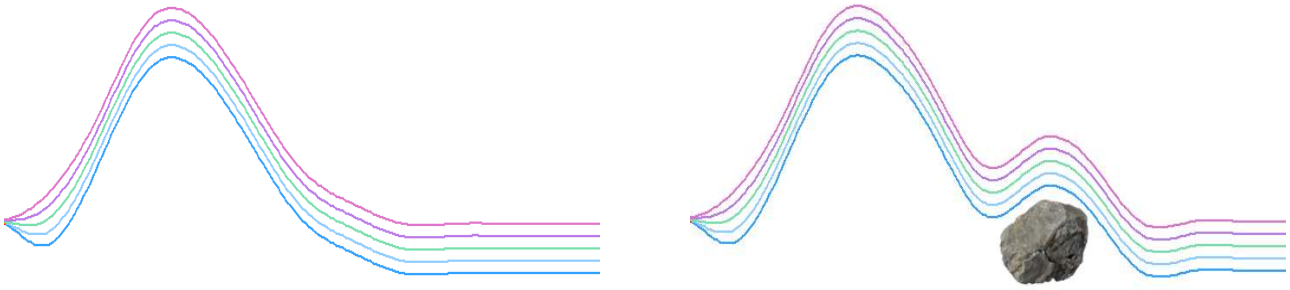
$$\text{cost}_{\text{KL}} = p(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta_D})}\right), \tag{3.1}$$

where $\boldsymbol{\Theta}$ denotes the policy parameters of the new policy ,$\boldsymbol{\Theta_D}$ denotes the policy parameters of the demonstrations, $p(\boldsymbol{x};\boldsymbol{\Theta})$ can bee seen as the likelihood of $\boldsymbol{x}$ under the policy p and $d(\boldsymbol{x};\boldsymbol{\Theta_D})$ denotes the likelihood of $\boldsymbol{x}$ under the demonstrations.
The KL divergence captures the information loss when expressing the demonstrations with the current policy. Moreover, the KL metric provides the advantage of being able to deal with any distribution of demonstrations. For the examples in this thesis, the policy and the distribution of the demonstrations are assumed to be Gaussian. Therefore, the policy can be written as $p(\boldsymbol{x};\boldsymbol{\Theta}) = \mathcal{N}(\boldsymbol{x}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma})$ and the demonstrations can be expressed by $d(\boldsymbol{x};\boldsymbol{\Theta}_D) = \mathcal{N}(\boldsymbol{x}\,|\,\boldsymbol{\mu}_D,\boldsymbol{\Sigma}_D)$, where $\boldsymbol{\mu}$ and $\boldsymbol{\mu}_D$ denote the means and $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_D$ denote the covariance matrices.
The use of a KL metric provides the advantage that the output of the trajectory optimization is not a single trajectory but captures the mean and the covariance of the demonstrations and can therefore be used to build a ProMP, which is explained more detailed in Chapter 4.
The second objective is given by the maximization of a reward function $R(x)$. This reward function can encode any additional task specific constraints, such as the distance to obstacles or via points. For the objective of obstacle avoidance a reward similar to the one presented in STOMP [14] is used

$$R(\boldsymbol{x}) = \sum_{t=1}^{T} \max\{\epsilon_{\text{dist}} - \text{d}_{\text{dist}}(x^{[t]}), 0\}, \tag{3.2}$$

where ϵ_{dist} denotes a safety radius around the obstacle, that should not be touched by the motion, T is the length of the trajectory and d_{dist} denotes the Euclidean distance to the closest obstacle and is extracted from a precomputed

distance field. Although this thesis focuses on obstacle avoidance, the reward function could also be used to meet additional objectives and even task specific constraints, which could be extracted out of the demonstration, such as via points.
Figure 3.1 illustrates the goal of minimizing the deviation to the distribution of demonstrations while avoiding an obstacle.



(a) Demonstrated trajectories (b) Optimized trajectories

Figure 3.1: A given distribution of trajectories should be modified to match different workspace settings under the objective of obstacle avoidance, while at the same time minimizing the deviation of the current policy to the demonstrated distribution.

In order to compute a policy, which meets this goal, a modification of the original Relative Entropy Policy Search (REPS) [10] is proposed as follows

$$\begin{aligned} &\underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \int_x p(\boldsymbol{x};\boldsymbol{\Theta})R(\boldsymbol{x}) - Bp(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta_D})}\right)d\boldsymbol{x} \\ \text{s.t.} \quad &\int_x p(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta_Q})}\right)d\boldsymbol{x} \leq \varepsilon, \qquad \int_x p(\boldsymbol{x};\boldsymbol{\Theta})d\boldsymbol{x} = 1, \end{aligned} \tag{3.3}$$

where ε denotes the upper bound for the deviation of the new policy p from the current policy q, $\boldsymbol{\Theta_Q}$ denotes the policy parameters of q and B denotes a tuning parameter for the trade off between staying close to the demonstrations and avoiding the obstacle. As in Equation 3.1, $p(\boldsymbol{x};\boldsymbol{\Theta})$ denotes the likelihood of $\boldsymbol{x}$ under the new policy, $q(\boldsymbol{x};\boldsymbol{\Theta}_Q)$ denotes the likelihood of $\boldsymbol{x}$ under of the current policy and $d(\boldsymbol{x};\boldsymbol{\Theta}_D)$ denotes the likelihood of $\boldsymbol{x}$ under the demonstrations. The constraints ensure that the policy update does not deviate to much from the old policy, and that the new policy still obeys the rules for probabilities to sum to one.
For the discrete case Equation 3.4 can be approximated by

$$\begin{aligned} &\underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})R(\boldsymbol{x}) - Bp(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta_D})}\right) \\ \text{s.t.} \quad &\sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta_Q})}\right) \leq \varepsilon; \qquad \sum_x p(\boldsymbol{x};\boldsymbol{\Theta}) = 1. \end{aligned} \tag{3.4}$$

An advantage of using REPS as an optimizer is a closed form solution, which can be obtained by using Lagrange's method. The Lagrangian $L(\boldsymbol{x},\eta,\lambda)$ of the given optimization problem 3.4 is given by

$$\begin{aligned} L(\boldsymbol{x},\eta,\lambda) &= \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})R(\boldsymbol{x}) - Bp(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)}\right) + \eta\left[\varepsilon - \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}\right)\right] + \lambda\left(1 - \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\right) \\ &= \eta\varepsilon + \lambda + \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\left[R(\boldsymbol{x}) - B\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)}\right) - \eta\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}\right) - \lambda\right] \end{aligned} \tag{3.5}$$

$$\text{s.t.} \qquad \eta \geq 0$$

where η and λ denote the Lagrange multipliers.
In order to obtain the optimal new policy the Lagrangian is differentiated with respect to $p(\boldsymbol{x};\boldsymbol{\Theta})$

$$\frac{\partial L(\boldsymbol{x},\eta,\lambda)}{\partial p(\boldsymbol{x};\boldsymbol{\Theta})} = R(\boldsymbol{x}) - \mathrm{B}\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)}\right) - B - \eta\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}\right) - \eta - \lambda. \tag{3.6}$$

Setting this to zero and solving for $p(\boldsymbol{x};\boldsymbol{\Theta})$ results in

$$\begin{aligned}
0 &= R(\boldsymbol{x}) - \mathrm{B}\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)}\right) - B - \eta\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}\right) - \eta - \lambda \\
&= R(\boldsymbol{x}) - B - \eta - \lambda - \left(\mathrm{B}\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)}\right) + \eta\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}\right)\right) \\
&= R(\boldsymbol{x}) - B - \eta - \lambda - \left(\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})^{B+\eta}}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)^B q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^\eta}\right)\right).
\end{aligned}$$

Hence, we obtain

$$p(\boldsymbol{x};\boldsymbol{\Theta}) = \exp\left(\frac{R(\boldsymbol{x}) - B - \eta - \lambda}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}. \tag{3.7}$$

By constraining the probabilities $p(\boldsymbol{x};\boldsymbol{\Theta})$ to still sum to one, this transforms to

$$\begin{aligned}
1 = \sum_x p(\boldsymbol{x};\boldsymbol{\Theta}) &= \sum_x \exp\left(\frac{R(\boldsymbol{x}) - B - \eta - \lambda}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}} \\
&= \exp\left(\frac{-B-\eta-\lambda}{B+\eta}\right) \sum_x \exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}.
\end{aligned}$$

Hence, we obtain

$$\exp\left(\frac{-B-\eta-\lambda}{B+\eta}\right) = \frac{1}{\sum_x \exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}}. \tag{3.8}$$

Next, the Lagrangian is rewritten as

$$L(\boldsymbol{x},\eta,\lambda) = \eta\varepsilon + \lambda + \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\left[R(\boldsymbol{x}) - \left(\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\Theta})^{B+\eta}}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)^B q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^\eta}\right)\right) - \lambda\right]. \tag{3.9}$$

Inserting 3.7 in the rewritten Lagrangian results in

$$\begin{aligned}
\eta\varepsilon + \lambda + \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\left[R(\boldsymbol{x}) - \left(\log\left(\frac{\left(\exp\left(\frac{R(\boldsymbol{x})-B-\eta-\lambda}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}\right)^{B+\eta}}{d(\boldsymbol{x};\boldsymbol{\Theta}_D)^B q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^\eta}\right)\right) - \lambda\right] = \\
\eta\varepsilon + \lambda + \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\left[R(\boldsymbol{x}) - \log\left(\exp\left(\frac{R(\boldsymbol{x})-B-\eta-\lambda}{B+\eta}\right)^{B+\eta}\right) - \lambda\right] = \\
\eta\varepsilon + \lambda + \sum_x p(\boldsymbol{x};\boldsymbol{\Theta})\left[R(\boldsymbol{x}) - (B+\eta)\left(\frac{R(\boldsymbol{x})-B-\eta-\lambda}{B+\eta}\right) - \lambda\right] = \\
\eta\varepsilon + \lambda + (B+\eta)\sum_x p(\boldsymbol{x};\boldsymbol{\Theta}).
\end{aligned} \tag{3.10}$$

By inserting $\sum_x p(\boldsymbol{x};\boldsymbol{\Theta}) = 1$ this transforms to

$$\eta\varepsilon + \lambda + B + \eta = \eta\varepsilon + (B+\eta)\log\left(\exp\left(\frac{\lambda+B+\eta}{B+\eta}\right)\right). \tag{3.11}$$

Inserting 3.8 in 3.11 the dual function $D(\boldsymbol{x}, \eta)$ can be computed

$$D(\boldsymbol{x}, \eta) = \eta\epsilon + (B+\eta)\log\left(\sum_x \exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}\right). \tag{3.12}$$

Rewriting the Dual results in

$$D(\boldsymbol{x}, \eta) = \eta\epsilon + (B+\eta)\log\left(\sum_x \exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} \frac{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}\right) \tag{3.13}$$

$$D(\boldsymbol{x}, \eta) = \eta\epsilon + (B+\eta)\log\left(\sum_x q(\boldsymbol{x};\boldsymbol{\Theta}_Q)\exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}-1}\right) \tag{3.14}$$

$$D(\boldsymbol{x}, \eta) = \eta\epsilon + (B+\eta)\log\left(\sum_x q(\boldsymbol{x};\boldsymbol{\Theta}_Q)\exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{-B}{B+\eta}}\right) \tag{3.15}$$

$$D(\boldsymbol{x}, \eta) = \eta\epsilon + (B+\eta)\log\left(\sum_x q(\boldsymbol{x};\boldsymbol{\Theta}_Q)\exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right)\left(\frac{d(\boldsymbol{x};\boldsymbol{\Theta}_D)}{q(\boldsymbol{x};\boldsymbol{\Theta}_Q)}\right)^{\frac{B}{B+\eta}}\right). \tag{3.16}$$

If we solve this using a number of N samples $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ obtained from the old policy q this transforms to

$$D(\boldsymbol{x}, \eta) = \eta\epsilon + (B+\eta)\log\left(\frac{1}{N}\sum_{i=1}^{\mathrm{N}} \exp\left(\frac{R(\boldsymbol{x}_i)}{B+\eta}\right)\left(\frac{d(\boldsymbol{x}_i;\boldsymbol{\Theta}_D)}{q(\boldsymbol{x}_i;\boldsymbol{\Theta}_Q)}\right)^{\frac{B}{B+\eta}}\right). \tag{3.17}$$

Following similar derivation found in Equations 3.12 - 3.17, Equation 3.8 is rewritten for the case of N samples

$$\begin{aligned}
\exp\left(\frac{-B-\eta-\lambda}{B+\eta}\right) &= \frac{1}{\sum_x \exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}} \\
\exp\left(\frac{-B-\eta-\lambda}{B+\eta}\right) &= \frac{1}{\frac{1}{N}\sum_{i=1}^{\mathrm{N}} \exp\left(\frac{R(\boldsymbol{x}_i)}{B+\eta}\right)\left(\frac{d(\boldsymbol{x}_i;\boldsymbol{\Theta}_D)}{q(\boldsymbol{x}_i;\boldsymbol{\Theta}_Q)}\right)^{\frac{B}{B+\eta}}}.
\end{aligned} \tag{3.18}$$

Next, Equation 3.7 is rewritten to

$$\begin{aligned}
p(\boldsymbol{x};\boldsymbol{\Theta}) &= \exp\left(\frac{R(\boldsymbol{x})-B-\eta-\lambda}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}} \\
p(\boldsymbol{x};\boldsymbol{\Theta}) &= \exp\left(\frac{-B-\eta-\lambda}{B+\eta}\right)\exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}.
\end{aligned} \tag{3.19}$$

Finally, computing a minimal $\eta = \eta^*$ from the Dual and inserting 3.18 into 3.19 results in the update rule for the new policy

$$p(\boldsymbol{x};\boldsymbol{\Theta}) = \frac{\exp\left(\frac{R(\boldsymbol{x})}{B+\eta}\right) d(\boldsymbol{x};\boldsymbol{\Theta}_D)^{\frac{B}{B+\eta}} q(\boldsymbol{x};\boldsymbol{\Theta}_Q)^{\frac{\eta}{B+\eta}}}{\frac{1}{N}\sum_{i=1}^{\mathrm{N}} \exp\left(\frac{R(\boldsymbol{x}_i)}{B+\eta}\right)\left(\frac{d(\boldsymbol{x}_i;\boldsymbol{\Theta}_D)}{q(\boldsymbol{x}_i;\boldsymbol{\Theta}_Q)}\right)^{\frac{B}{B+\eta}}}. \tag{3.20}$$

In particular, if the policy is assumed to be Gaussian with $\Theta = (\mu, \Sigma)$, one obtains the new mean μ and the new covariance matrix Σ with a Maximum Likelihood update

$$\mu = \frac{\sum_{i=1}^{N} w_i \boldsymbol{x}_i}{\sum_{i=1}^{N} w_i}; \tag{3.21}$$

$$\Sigma = \frac{\sum_{i=1}^{N} w_i (x_i - \mu)(x_i - \mu)^T}{z}; \tag{3.22}$$

$$w_i = \exp\left(\frac{R(\boldsymbol{x}_i)}{B + \eta^*}\right)\left(\frac{d(\boldsymbol{x}_i; \Theta_D)}{q(\boldsymbol{x}_i; \Theta_Q)}\right)^{\frac{B}{B+\eta^*}} \tag{3.23}$$

$$= \exp\left(\frac{R(\boldsymbol{x}_i)}{B + \eta^*}\right)\exp\left(\log\left(\left(\frac{d(\boldsymbol{x}_i; \Theta_D)}{q(\boldsymbol{x}_i; \Theta_Q)}\right)^{\frac{B}{B+\eta^*}}\right)\right) \tag{3.24}$$

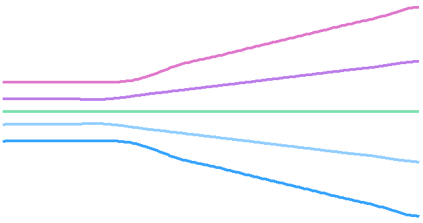
$$= \exp\left(\frac{R(\boldsymbol{x}_i) + B(\log(d(\boldsymbol{x}_i; \Theta_D))) - \log(q(\boldsymbol{x}_i; \Theta_Q)))}{B + \eta^*}\right) \qquad i = 1...N$$

$$z = \frac{\sum_{i=1}^{N} (w_i)^2 - \sum_{i=1}^{N} (w_i)^2}{\sum_{i=1}^{N} w_i}.$$

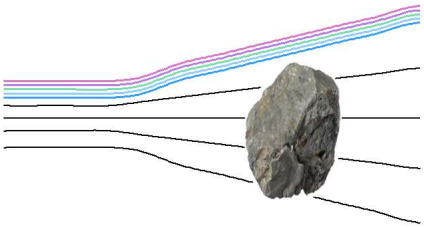
Following this trajectory optimization the algorithm iterates between policy update and policy evaluation steps until convergence of the policy.

3.1.2 Sliding Window Approach

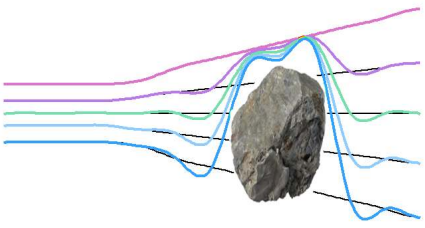
Staying close to demonstrations is a very general term when used in connection with distributions. Figure 3.2 shows two possible ways of staying close to the demonstrations during obstacle avoidance.



(a) Demonstrated trajectories



(b) Optimized trajectories (global)

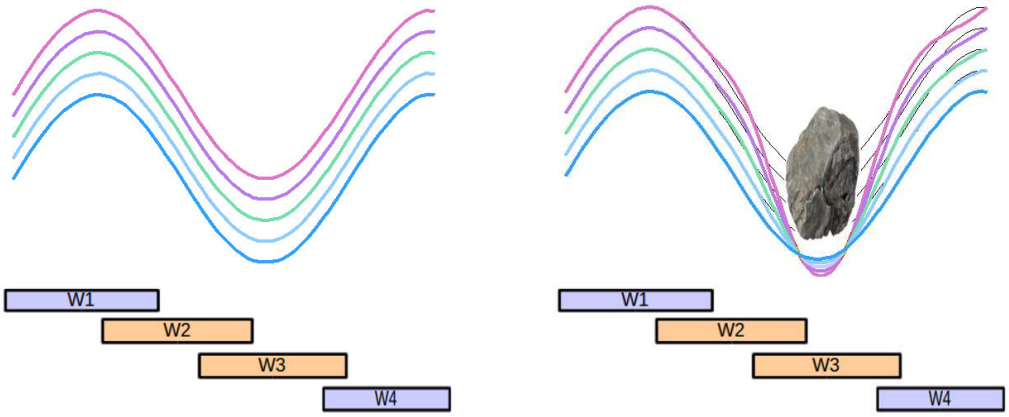


(c) Optimized trajectories (local)

Figure 3.2: Staying close to demonstrations can have different meanings when connected to obstacle avoidance. (a) shows a set of demonstrated trajectories, (b) shows the case that we want to stay close to that part of the demonstrations that is not affected by the obstacle at all, (c) shows a solution if we want to stay close to the demonstrations in every part but the one affected by the obstacle.

In Figure 3.2 (b) the solution picks the demonstrated trajectories, which completely avoid the obstacle. On the other hand, Figure 3.2 (c) deforms the demonstrated trajectories only in the parts affected by the obstacle. Both

solutions 3.2 (b) and 3.2 (c) are valid and their suitability depends on the task at hand.
To obtain an algorithm that is capable of dealing with both types of applications, a sliding window approach is introduced in addition to the policy search presented in 3.1. This window can be adapted in its size and scales how much the algorithm penalizes deformations in regions that are not explicitly affected by the presence of obstacles. Figure 3.3 illustrates the sliding window approach. The windows are initialized with a certain overlap to provide smooth transitions after the reconstruction of full trajectories. Hereinafter the window size will be referred relative to the total trajectory length such as 1/10 or 1/5. The specific window size needs to be adapted dependent on the particular task, but experiments have shown that empirically a 1/5 window provides good results.



(a) Demonstrated trajectories (b) Optimized trajectories

Figure 3.3: Introducing sliding windows into the optimization results in independent parts of the trajectory being optimized according to local costs. (a) shows the partition of the trajectory space into 4 overlapping 1/4 windows where only W3 and W4 are affected by the presence of the obstacle, (b) shows a desired solution where the solution completely mimics the demonstrations in the areas not affected by the obstacle and stays as close as possible to them in the areas affected by the obstacle.

For a number of W windows the sliding window approach results in W overlapping distributions. In particular, if the distributions are assumed to be Gaussian this results in a set of means $M = \{\mu_1, ..., \mu_W\}$ and a set of covariance matrices $C = \{\Sigma_1, ..., \Sigma_W\}$. Figure 3.4 shows the structure of the obtained means and covariance matrices on an example with a 3/5 window.

(a) (b)

Figure 3.4: In the case of Gaussian distributions the sliding widow approach results in a set of overlapping means, shown in (a), and a set of overlapping covariance matrices, shown in (b).

A core question is how to merge the solutions for the single windows to obtain a global mean μ and a global covariance matrix Σ. This thesis presents two approaches to merge the windows.
The first approach is based on naive averaging over the obtained means and covariance matrices. Following

this approach for the example from Figure 3.4, the merging of the means $\boldsymbol{\mu}_a = [a_1, a_2, a_3]$, $\boldsymbol{\mu}_b = [b_1, b_2, b_3]$, $\boldsymbol{\mu}_c = [c_1, c_2, c_3]$ and the covariance matrices $\boldsymbol{\Sigma}_a = \left[a_{ij}\right]$,$\boldsymbol{\Sigma}_b = \left[b_{ij}\right]$, $\boldsymbol{\Sigma}_c = \left[c_{ij}\right]$ results in

$$\boldsymbol{\mu} = \left[a_1 \;\; \frac{a_2 + b_1}{2} \;\; \frac{a_3 + b_2 + c_1}{3} \;\; \frac{b_3 + c_2}{2} \;\; c_3\right]$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & (a22 + b_{11})/2 & (a23 + b_{12})/2 & b_{13} & 0 \\ a_{31} & (a32 + b_{21})/2 & (a33 + b_{22} + c_{11})/3 & (b_{23} + c_{12})/2 & c_{13} \\ 0 & b_{31} & (b_{32} + c_{21})/2 & (b_{33} + c_{22})/2 & c_{23} \\ 0 & 0 & c_{31} & c_{32} & c_{33} \end{bmatrix}. \tag{3.25}$$

The second approach samples N trajectories τ for each window and iteratively merges the trajectories of neighbouring windows $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ by connecting the trajectories closest to one another. The global mean and covariance matrix are computed with a Maximum Likelihood estimate on the merged trajectories. The pseudo code for the second approach is given below.

Window Merging Pseudo Code

Input: $M = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_W\}$, $C = \{\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_W\}$, N

for w = 1:W
    $\boldsymbol{\tau}_w$ = mvnrand(μ_w,Σ_w, N)
$\boldsymbol{\omega}_1 = \boldsymbol{\tau}_1$

for w = 2:W
    $\boldsymbol{\omega}_2 = \boldsymbol{\tau}_w$
    $[\tilde{\boldsymbol{\omega}}_1 \; \tilde{\boldsymbol{\omega}}_2]$=get overlapping parts of $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$
    for k = 1:K
        for l = 1:number of trajectories in $\tilde{\boldsymbol{\omega}}_2$
            Δ(l) = $\sum$(abs($\tilde{\boldsymbol{\omega}}_1$(k)-$\tilde{\boldsymbol{\omega}}_2$(l)))
        δ(k) = (min(Δ))
        match(k) = index of (min(Δ))
        remove match from $\tilde{\boldsymbol{\omega}}_2$
    match(δ < mean(δ)*ε_Δ) = -1
    for k = 1:K
        if match(k)>0
            $\boldsymbol{\tau}_{\text{merged}}$(k) = blendTrajectories($\tilde{\boldsymbol{\omega}}_1$(k),$\tilde{\boldsymbol{\omega}}_2$(match(k)))
    $\boldsymbol{\omega}_1 = \boldsymbol{\tau}_{\text{merged}}$

return $\boldsymbol{\tau}_{\text{merged}}$

Blend Trajectories Pseudo Code

Input: $\boldsymbol{x}_1$, n = length($\boldsymbol{x}_1$), $\boldsymbol{x}_2$, m = length($\boldsymbol{x}_2$), l_o = length(overlap($\boldsymbol{x}_1$, $\boldsymbol{x}_2$)

$\alpha = 1/(l_o+1))$
$\boldsymbol{x}_{12}$(1:n-l) = $\boldsymbol{x}_1$(1:n-l_o)
w = 1
for i = n-l_o+1:n
    $\boldsymbol{x}_{12}$(i) = α (l_o+1-w)$\boldsymbol{x}_1$(i) + α w$\boldsymbol{x}_2$(i-n-l_o))
    w = w+1
$\boldsymbol{x}_{12}$(n+1:n+m-l_o) = $\boldsymbol{x}_2$(l_o+1:end)

return $\boldsymbol{x}_{12}$

Figure 3.5 compares the results of the two approaches on two examples.

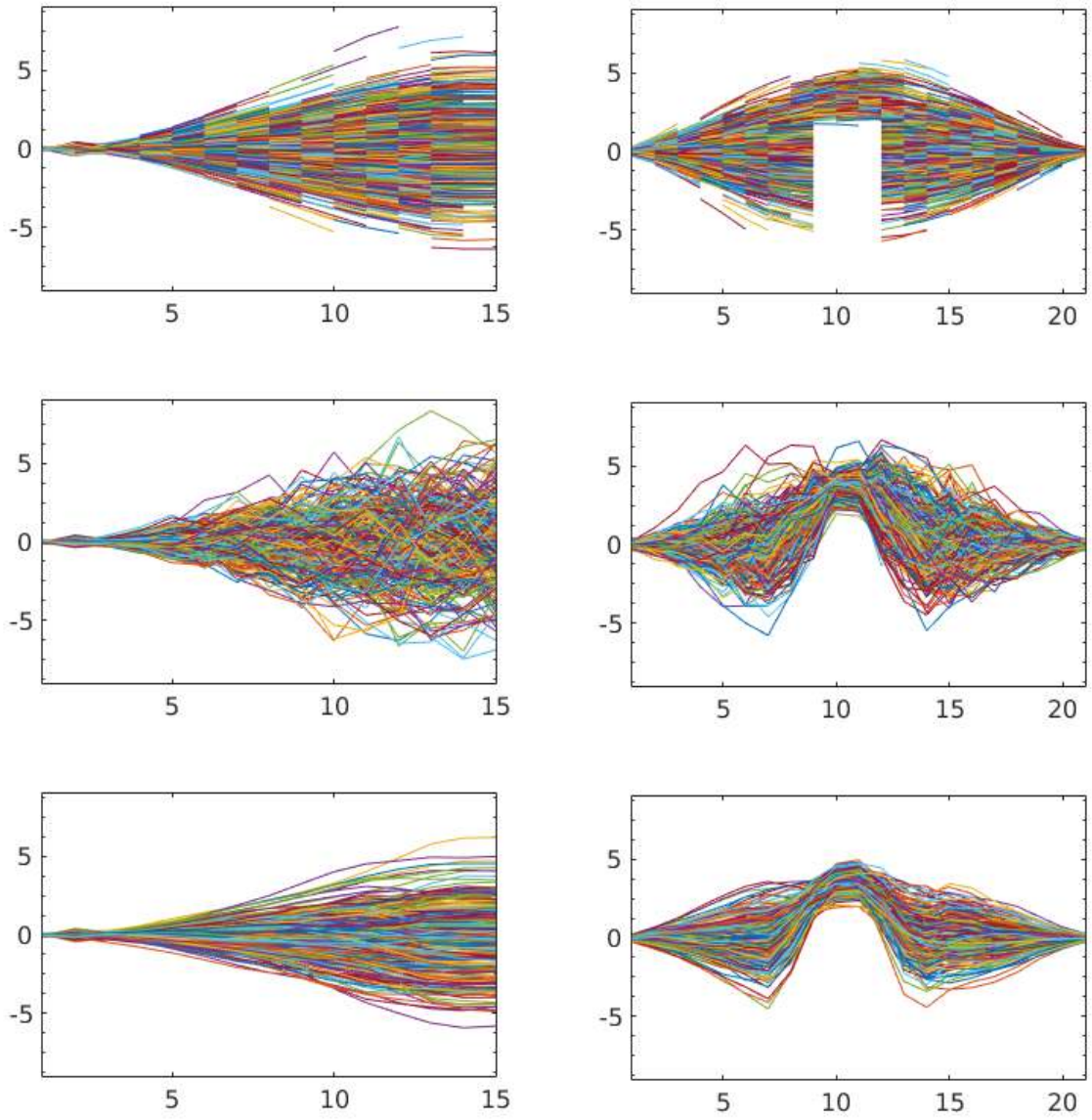


Figure 3.5: The sliding window approach results in overlapping distributions as shown in row one. Comparing the two approaches to merge those part wise solutions for the examples (a) and (b), it shows that the second approach (row three) captures more of the correlation and results in smoother trajectories than the first approach (row two).

For the first approach, shown in the second row, it reveals that the merging is able to reproduce trajectories of similar shape to the ones sampled from the single windows. However, the merged trajectories do not contain the full correlation that was present in the local solutions.
The results of the second approach are shown in the third row. The results indicate that this approach produces much smoother solutions and that the merged trajectories contain most of the correlation of the single windows.
Referring to these results it was decided to use the second approach for merging the windows. However, the second approach is computationally more expensive than the first one and it should be a goal for future work to find a faster way to merge the windows.

3.1.3 Basic Structure

The proposed method follows the main structure of stochastic trajectory optimization [14] and extends the original REPS formulation to account for human demonstrations in the policy update. Figure 3.6 illustrates the single steps of the algorithm.
Working on K demonstrated trajectories for N degrees of freedom, first a Gaussian distribution of the demonstrated

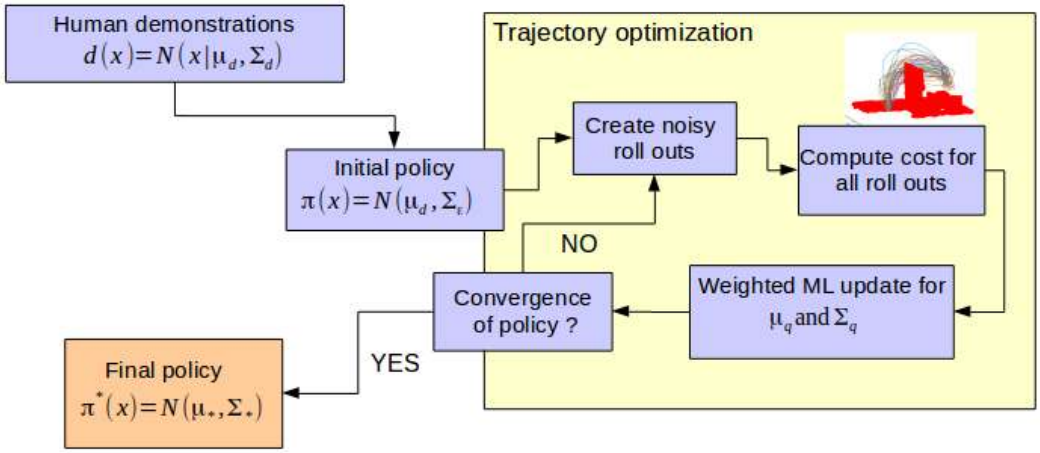


Figure 3.6: The main structure of the proposed algorithm.

trajectories $d^w(x)=\mathcal{N}(\mu_d^w,\Sigma_d^w)$ is obtained for each window. Subsequently the policy is initialized with the mean of the trajectories and an uncorrelated noise covariance matrix $\Sigma_d^w = \text{diag}(\Sigma_d^w + \zeta)$, where ζ denotes some small factor to ensure sufficient exploration. In each policy evaluation step K roll outs are created and the policy gets updated according to a weighted Maximum Likelihood update. The algorithm applies this proposed optimization on each single window till convergence and subsequently merges the results of the windows by connecting the resulting trajectories. A pseudo code of the algorithm is given below.

Algorithm Pseudo Code

Input: $\boldsymbol{\mu}_d^w$, $\boldsymbol{\Sigma}_d^w$

till convergence of all μ_q^w,Σ_q^w
  for each window w
    for k = 1:K
      $\boldsymbol{e}(k,:)^w$=mvnrnd($\boldsymbol{\mu}_q^w$, $\boldsymbol{\Sigma}_q^w$)
      $logd(k)^w$=logmvnpdf($\boldsymbol{e}(k)^w$,$\boldsymbol{\mu}_d^w$, $\boldsymbol{\Sigma}_d^w$)
      $logq(k)^w$=logmvnpdf($\boldsymbol{e}(k)^w$,$\boldsymbol{\mu}_q^w$, $\boldsymbol{\Sigma}_q^w$)
      R(K) = cost($e(k)^w$)
    get η_{opt} from minimizing the Dual
    for 1:K
      $\boldsymbol{\delta}$(k)=exp((1/(B+η_{opt})) * ((-R(k)) + B * (logd(k) - logq(k)))
    $\boldsymbol{\mu}_q^w \leftarrow$ ML update($\boldsymbol{\mu}_q^w$,δ)
    $\boldsymbol{\Sigma}_p^w \leftarrow$ ML update($\boldsymbol{\mu}_q^w$,$\boldsymbol{\mu}_p^w$,$\boldsymbol{\delta}$)
$\boldsymbol{\mu}_q \leftarrow$ merge(μ_q^w) of all windows
$\Sigma_q \leftarrow$ merge($\boldsymbol{\Sigma}_q^w$) of all windows

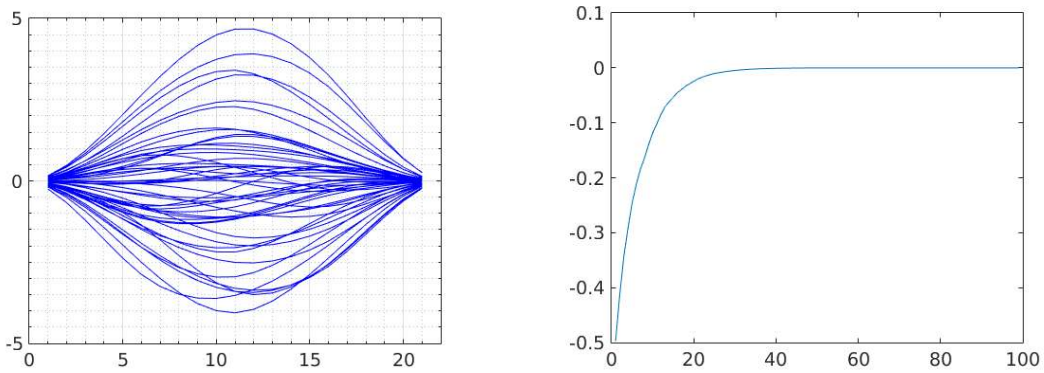
return $\boldsymbol{\mu}_q$, $\boldsymbol{\Sigma}_q$

3.2 Basic Characteristics

Humans differ from machines in the variability they show during the execution of motions. They do not reproduce their trajectories in an exact way but tend to include variance in the movements. The algorithm, which is presented in this thesis, preserves this variance during trajectory optimization.
This section presents results on specific toy examples to determine the basic properties of the proposed algorithm. It reveals the methods ability to converge to a distribution of demonstrations and capture the smoothness without prior assumptions. Moreover, it discusses the idea of incorporating a sliding window inside the optimization.

3.2.1 Eliminating the Need for Prior Assumptions on the Solution

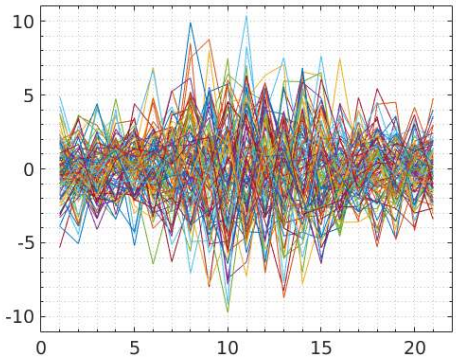
To force smooth solutions, stochastic optimization methods such as STOMP and Pi2 rely on additional mechanisms and prior assumptions on the solution. In STOMP the covariance matrix from which noise is sampled is strongly correlated, thus generating smooth perturbations. In Pi2, the solution is forced to be smooth as the policy is parametrized by radial basis functions. By using a KL metric to minimize the divergence with the distribution of the demonstrations, the proposed method eliminates the need for such prior knowledge in the optimization process and obtains all required information out of the given demonstrations.
Figure 3.7 shows the result of using the proposed algorithm on a set of smooth demonstrations, illustrated in 3.7(a). In this first toy example no obstacles are considered, allowing to evaluate the adaptation to the demonstrated distribution in isolation.



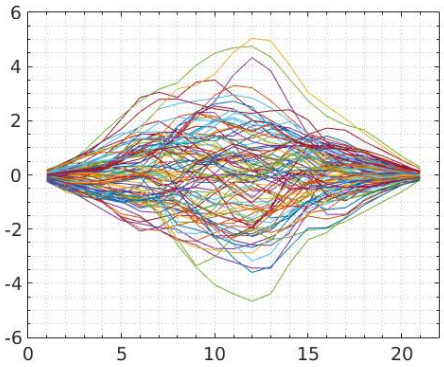
(a) demonstrated trajectories (b) average reward of trajectories

Figure 3.7: The trajectory optimization converges towards the demonstrated distribution as the reward function converges towards zero.

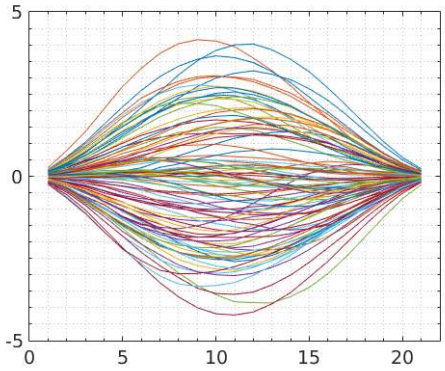
Figure 3.7(b) illustrates the optimization of the reward function. The reward convergence towards zero as the policy results in minimal deviation from the distributions.
Figures 3.8(a), 3.8(b) and 3.8(c) show the method's ability to capture the full range of the demonstrations including variance as well as correlation. The figures show the changes of the trajectory distribution during the optimization process. Even though the policy is initialized with an uncorrelated covariance matrix, the solution converges to the smooth trajectory distribution of the demonstrations. The plots of the trajectories reveal how the correlation of the demonstrations is imitated by the solution, by only using the KL metric.



(a) uncorrelated initial noise



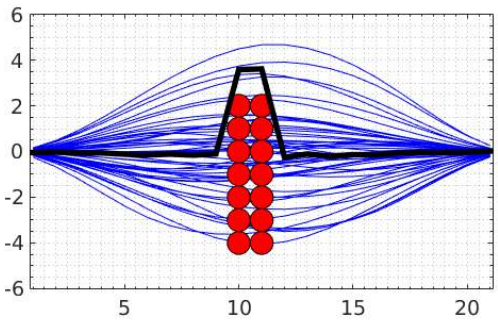
(b) solution after 20 iterations



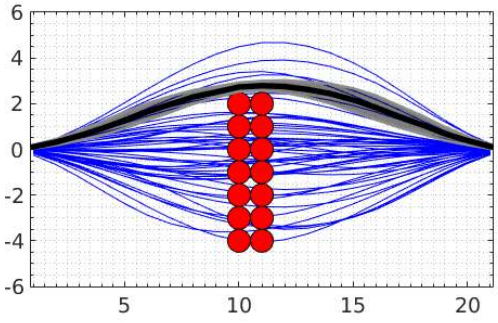
(c) solution after 100 iterations

Figure 3.8: Even though it is initialized with a highly uncorrelated covariance matrix the trajectory optimization converges towards the demonstrated distribution and captures the smoothness and correlation inside the trajectories.

In the second toy example obstacles were added to this set up. Figure 3.9 illustrates the results of this experiment and furthermore compares it to common stochastic trajectory optimization [14]. For the comparison an uncorrelated initial covariance matrix was also used in STOMP. It reveals that in this case STOMP is able to avoid



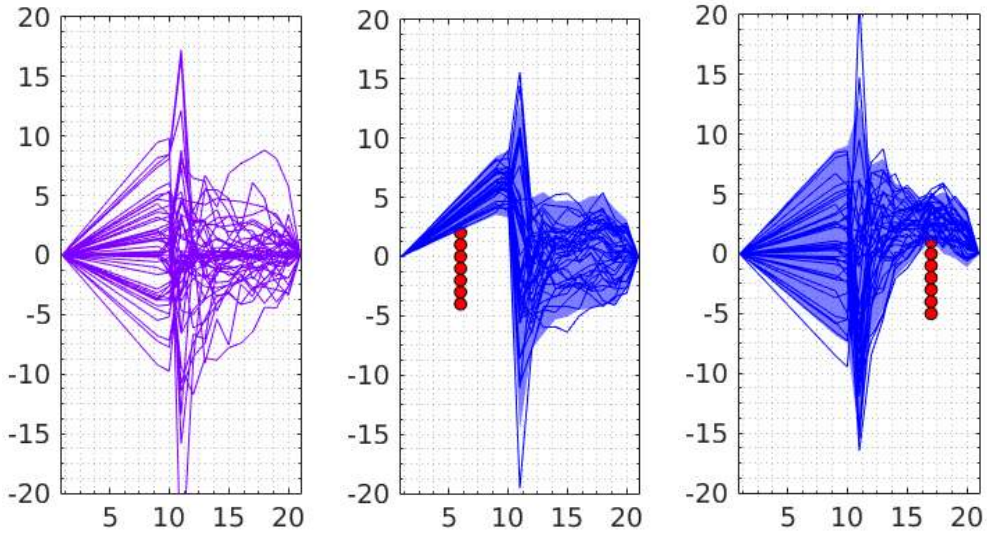
(a) result of STOMP



(b) result of the proposed algorithm

Figure 3.9: (a) Common trajectory optimization algorithms such as STOMP do not consider demonstrations and therefore end up with non-smooth results if no additional prior assumptions are added inside the cost function. Using the KL-metric the solution captures the correlation of the demonstrated trajectories and results in smooth obstacle avoidance, if smoothness was present inside the demonstrations.

the obstacles but does not capture the correlation from the demonstrations and therefore results in a non smooth solution. In comparison, the newly proposed algorithm offers a solution that avoids the obstacle and moreover incorporates the smoothness of the demonstrations. In opposite to STOMP, which only offers a single trajectory, this solution also includes variance.
The third experiment provides a set of demonstrations that are half smooth and half non-smooth. Figure 3.10 (a) shows the demonstrations. In this particular case adding prior assumptions would be a difficult hand tuning problem as the demonstrations are not correlated over the full trajectory but only in certain parts. It points out the advantages of not adding hand coded extra terms to the cost function or the initial covariance matrix but extracting the correlation directly out of the demonstrated trajectories. When a uncorrelated initial covariance matrix is



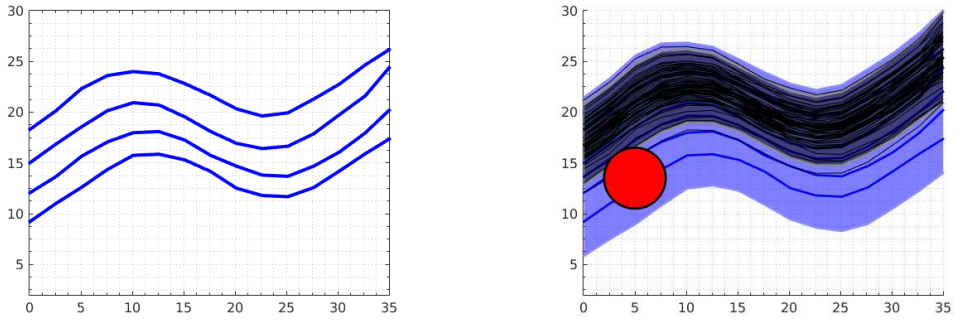
(a) Demonstrations (b) Obstacle left (c) Obstacle right

Figure 3.10: Even on demonstrations with half smooth and half non-smooth trajectories, as shown in (a), the algorithm is able to capture the full correlation. The solution can capture the smooth and the non-smooth part and avoid obstacles in either part (b), (c).

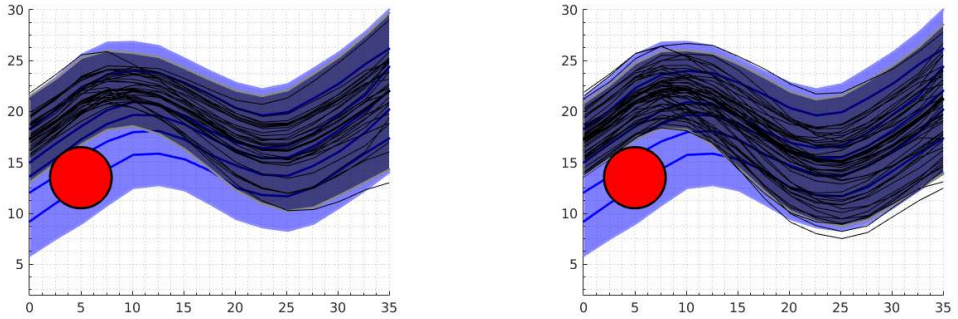
STOMP would not be able to capture the correlation in the first half and when it is used with a correlated initial covariance the second part could not be imitated. The KL metric inside the newly proposed algorithm overcomes this problems and is able to minimize deviations to the demonstrations in the smooth as well as in the non-smooth parts. Figures 3.10 (b) and (c) illustrate this for obstacles in the smooth part and obstacles in the non-smooth part.

3.2.2 Integrating a Sliding Window Approach

Applying REPS inside the trajectory optimization as explained in Section 3.1 introduces a drawback when trying to modify trajectories only locally. Intuitively, the method introduced in Chapter 3 favours the solution in Figure 3.2 (b). To achieve the solution sketched in 3.2 (c), the sliding window approach becomes essential. The window size can be used to tune the influence of obstacles in single region on the hole trajectory.
The optimization is first illustrated on a toy example, where part of the demonstrated trajectories are traversed by an obstacle. Figure 3.11 illustrates the effect of adapting the window size.
Figure 3.11 (b) reveals that without the sliding window the obstacle causes the variance in the hole trajectory to be influenced by the obstacle. In opposite a sliding window results in only local affected solutions as illustrated in figures 3.11 (c) and (d). The figures show that the optimized trajectories for a 7/15 window are only affected in the left part and converge against the demonstrations in the right part. This effect becomes more evident for a 3/15 window. In this case, shown in Figure 3.11 (d), just the part in the direct surrounding of the obstacle is modified by the optimization and the solution imitates the demonstrations in the remaining part.



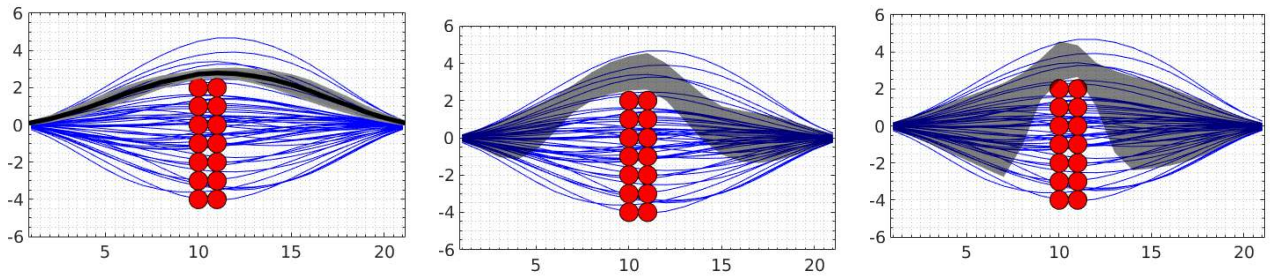
(a) Demonstrated trajectories (b) Optimization without a sliding window



(c) Optimization with a 10/15 window (d) Optimization with a 5/15 window

Figure 3.11: Depending on the chosen window size a local obstacle can have different effects on the full trajectory.

Using the sliding window on the example from 3.9 the solution can be modified to cover more of the variance in the demonstrations in regions not affected by the obstacle. Figure 3.12 compares the solution obtained without a sliding window to the solutions obtained with two different window sizes.



(a) solution without a sliding window (b) solution with a 7/15 window (c) solution with a 3/15 window

Figure 3.12: Using a sliding window the solution can be adapted to lean closer to the demonstrated trajectories in the parts not affected by the obstacles.

The results show how the trajectories lean closer to the obstacle as the variance of the demonstrations can be imitated in the regions not affected by the obstacle. Similar to the example in Figure 3.11 this effect gets even stronger as the window size is decreased.

3.3 Toy Example of Drawing a Letter

This section presents the results of a 2D toy example. In this example demonstrated trajectories of drawing a small letter are supposed to be imitated by a planar 2-link robot arm. The section discusses the applications of the proposed algorithm in task space as well as in joint space.

3.3.1 Application in Task Space

Influenced by experience, humans are used to think in Cartesian directions such as up, down left or right. Therefore, applying the developed algorithm in task space provides an intuitive method and easy specification of dimensional constraints. In particular, in task space it is straightforward to constraint the execution to only vary in e.g the height but not in the x-y direction or the other way around.
Figure 3.13 illustrates the basic concept of the task space application.

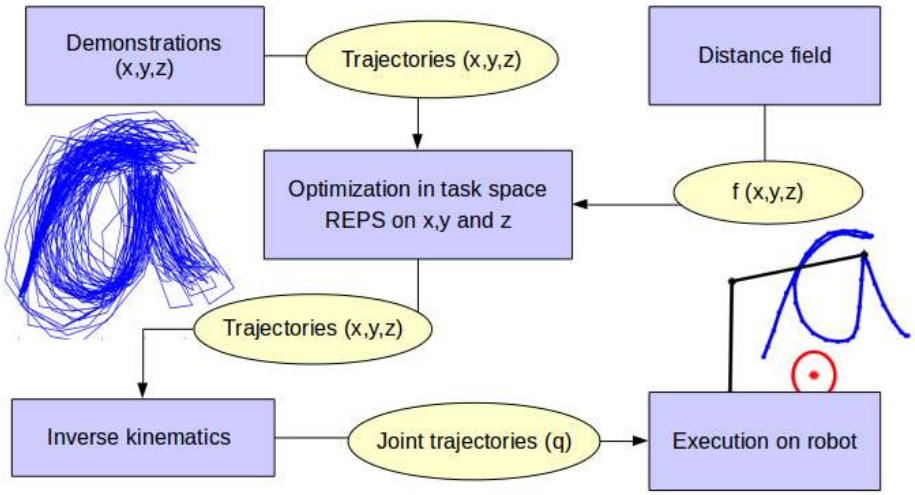
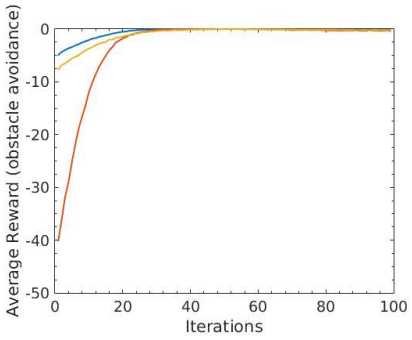
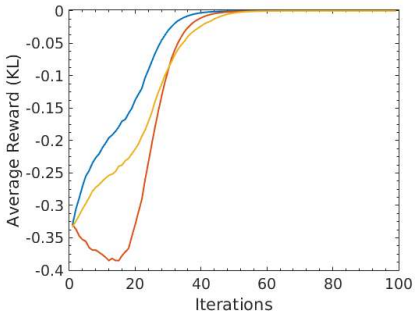


Figure 3.13: Overview for task space application of the algorithm

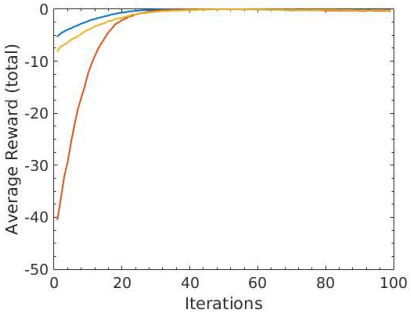
The demonstrations are given as 2D trajectories $\tau_i = (x_i, y_i) \in \mathbb{R}^2$, where i=1..T. The subsequent optimization uses an Euclidean distance metric to avoid the obstacles and a KL metric to minimize the deviation to the demonstrated distribution. To execute the solution on a robot the solution is afterwards transformed to joint space by using inverse kinematics.
In the toy example a set of drawings of the small letter is given as demonstrations. A planar two-link robot arm is supposed to redraw this letter in a workspace setting where an additional obstacle is placed at varying locations. Figure 3.14 reveals how the optimization converges towards a trade off between the KL and the distance metric. It illustrates the average trajectory reward for both objectives and the total reward over 100 iterations and 3 different obstacle positions. The results show the solutions satisfy both, avoidance of the obstacle and similarity to the demonstrations.
Figure 3.15 shows the deformation of the distribution depending on the position of the obstacle. In particular, it illustrates the algorithms capability of avoiding the obstacle and at the same time preserving the variance of the demonstrations in the intervals that are not affected of the obstacle. The algorithm was used with a 11/41 window size. In comparison to a common trajectory optimization method as STOMP the proposed method is not only able to preserve the variance of the demonstrations but also results in solutions that stay closer to the given demonstrations. In particular, this is illustrated by figure 3.15. It compares the solution of the proposed algorithm and the solution resulting from STOMPS trajectory optimization for different obstacle positions. For all obstacle settings the solution stays closer to the distribution of trajectories than the solution of STOMP.



(a) average reward of distance metric



(b) average reward of KL metric



(c) total average reward

Figure 3.14: The average trajectory reward reveals that the solution converges towards a trade off between minimal deviation to the distribution of demonstrations and obstacle avoidance.

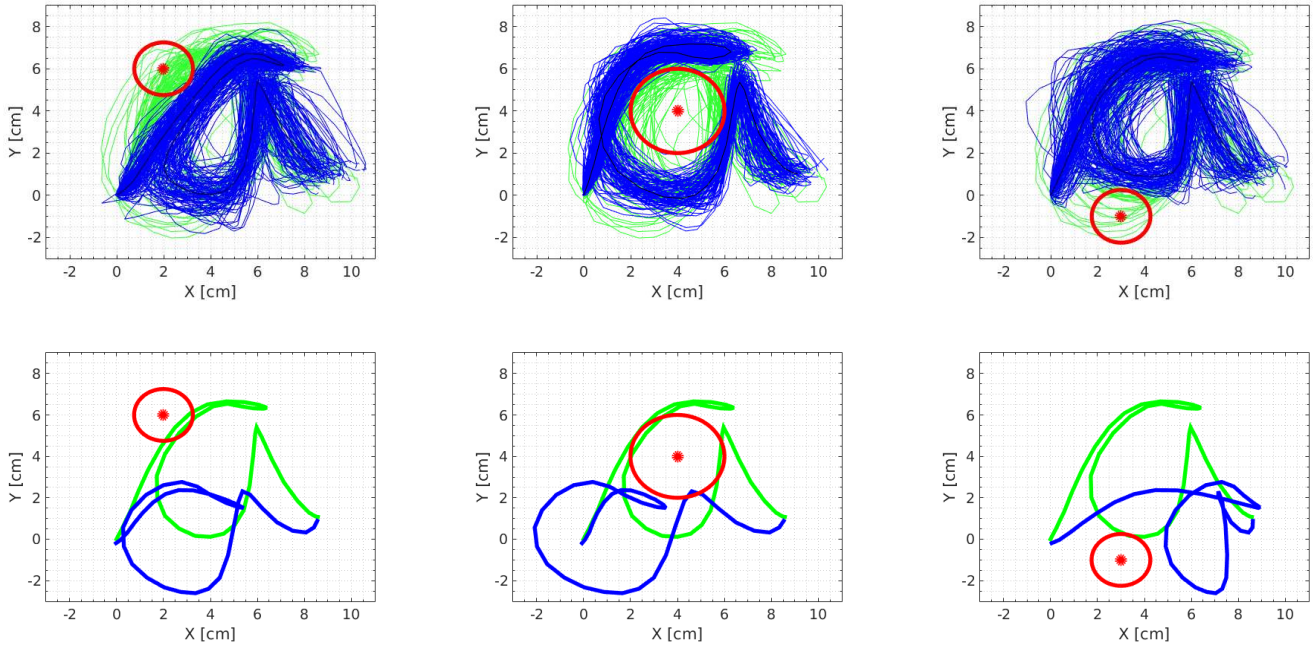
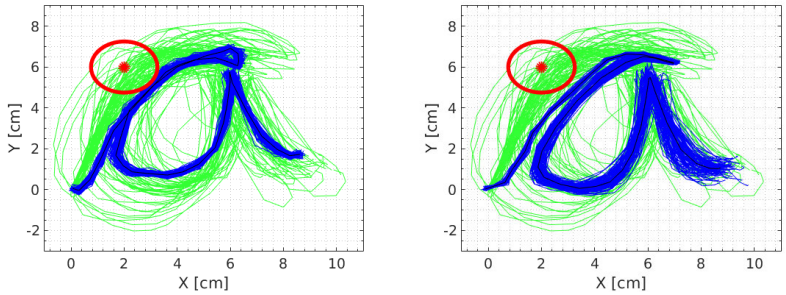
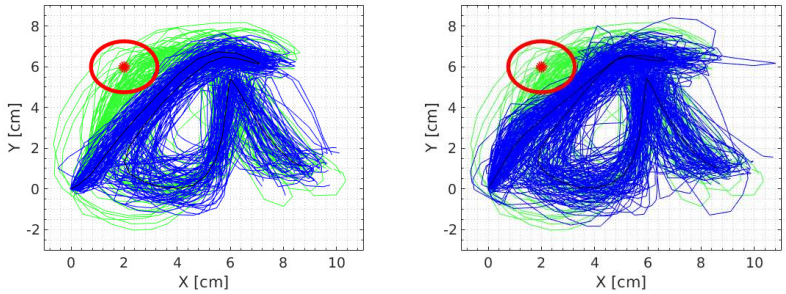


Figure 3.15: The upper row shows solutions of the proposed algorith (blue), the lower one optimized trajectories of STOMP (blue). In opposite to common trajectory optimization methods such as STOMP the proposed algorithm is able to preserve the variance of the demonstrations (green) during the trajectory optimization. Moreover the solutions stay closer to the demonstrations due to the KL metric.

As explained in Section 3.2.2 the window size has a main effect on the shape of the solution. Figures 3.16 and 3.17 illustrate this effect on the given set of demonstrations.

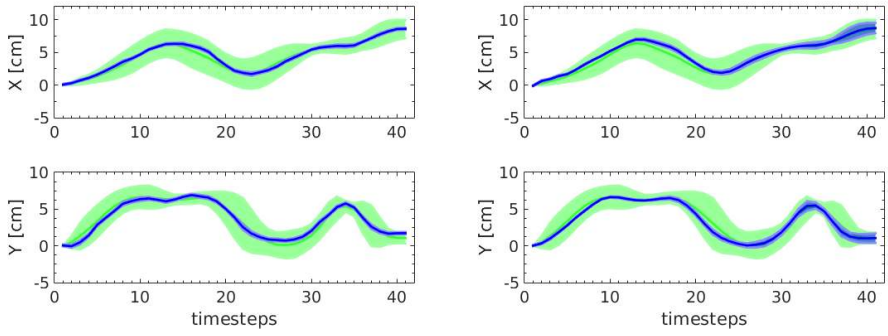


(a) 41/41 window (b) 21/41 window

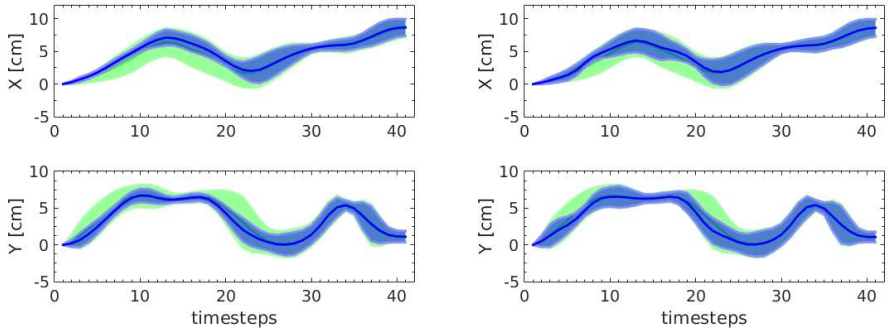


(c) 11/41 window (d) 3/41 window

Figure 3.16: The window size has a main influence of the resulting shape of the trajectories. In particular, it provides a tuning parameter for global effects of local obstacles.



(a) 41/41 window (b) 21/41 window



(c) 11/41 window (d) 3/41 window

Figure 3.17: The effects of the window size show also in the resulting distributions for the single degrees of freedom.

They illustrate how the local presence of an obstacle has influence on the global solution for bigger window sizes such as 41/41 or 21/41. This can be seen on both, the trajectory plots in Figure 3.16 and the plots for the single degrees of freedom in Figure 3.17. In particular, the variance tends to shrink even in regions further away from the obstacle. In opposite smaller window sizes such as 11 or 3 points result in solutions that only locally adapt the demonstrations to the present obstacle and preserve the full variance of the demonstrations in the remaining parts of the trajectories. Moreover, the plots show that the local deformation comes with the trade off of solutions that only contain local correlation between the points. This reveals as the trajectories for smaller window sizes are less smooth then the ones for large window sizes.
Using inverse kinematics the optimized trajectory distribution can be used to create joint trajectories for a planar two link robot. Figure 3.18 shows how the robot redraws the letter and avoids the obstacle by following the optimized trajectory.

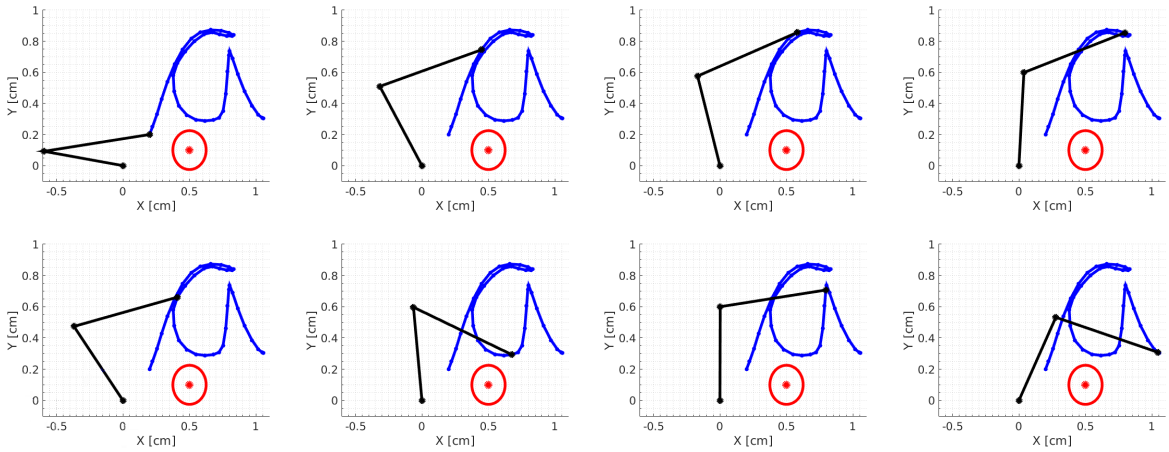


Figure 3.18: Using inverse kinematics the task space solution can be transformed to joint space. In this toy example a planar two link robot becomes able to redraw the demonstrated letter, while at the same time avoiding the obstacle.

However, trajectory optimization in task space might result in problems related to feasibility after the application of inverse kinematics. Those can occur due to joint limits as well as workspace limitations. They can be avoided by incorporating a additional precision metric inside the optimization to constraint the solution to be feasible for the robot or shift the optimization to joint space as explained in the following section.

3.3.2 Application in Joint Space

A major problem concerning trajectory optimization in task space is given by potentially infeasible trajectories after the optimization. The optimized trajectories can be infeasible for the robot due to joint limits or workspace constraints that are not taken into account by the optimization in task space. One way to avoid infeasibility is given by extra terms in the cost function to ensure sufficient accuracy of the robot tracking the trajectories. However, optimizing the trajectories in joint space can avoid trajectories, that violate joint limits or workspace constraints, without the need to modify the cost function. To ensure feasibility the noisy updates can be cut at the joint limits, as proposed in STOMP[14]. As the parameter updates are convex combinations of the noisy roll outs this ensures that the optimization stays within the joint limits.
The basic concept for the application in joint space is illustrated by Figure 3.19. Similar to the example for the task space application the demonstrations are collected in task space. It could also be considered to collect them in joint space (e.g through kinaesthetics teaching) but this may constrain the demonstrations unnecessarily. Moreover, obtaining all trajectories in task space makes them reusable for different robot platforms, as the recorded end effector poses do not require any prior assumptions on the later used robot model.
Subsequently, the trajectories are transformed into joint space trajectories by using inverse kinematics. The optimization takes place in joint space and forward kinematics are applied to obtain the collision distance metric. The obtained solution can then be used to execute the trajectory on a robot. In order to apply the optimization in joint space the data is normalized as in joint space small noise changes have larger influence on the result.

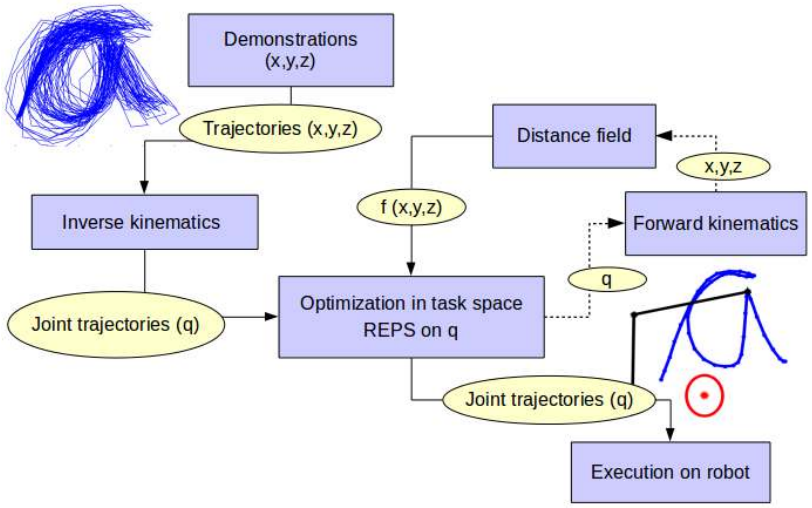
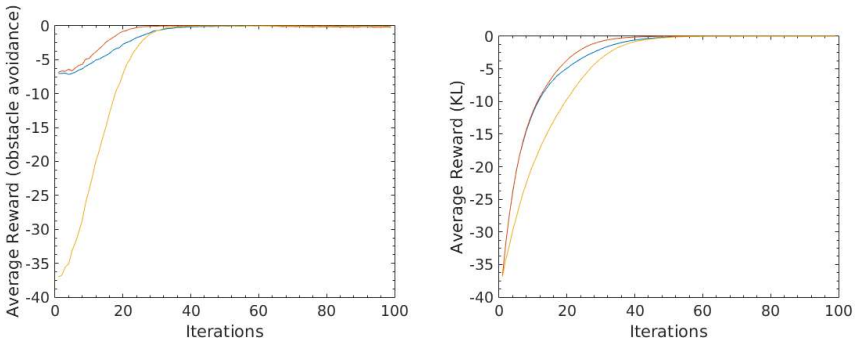
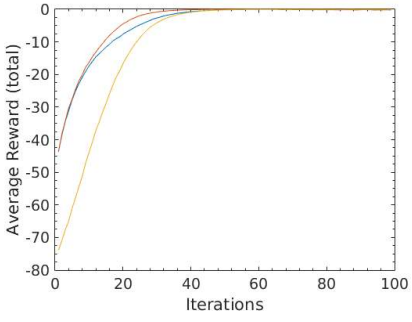


Figure 3.19: Overview for joint space application of the algorithm

In comparison to the application in task space the computation can take longer as forward kinematics need to be applied in every cost function evaluation. On the other hand the resulting solution is guaranteed to be feasible w.r.t the robot kinematics. Similar to the task space example the algorithm is applied on the a data set with a 2 link planar robot. Figure 3.20 presents plots of reward or different obstacle settings.



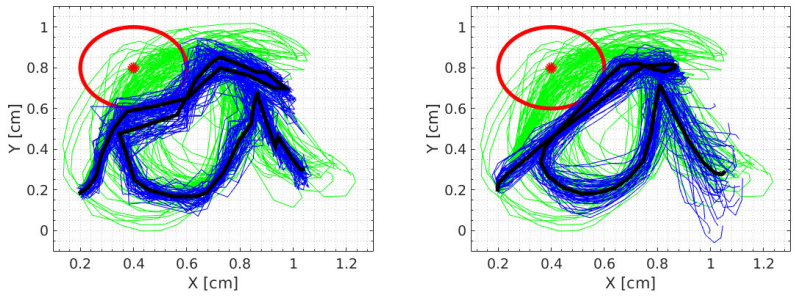
(a) Average reward of distance metric (b) Average reward of KL metric



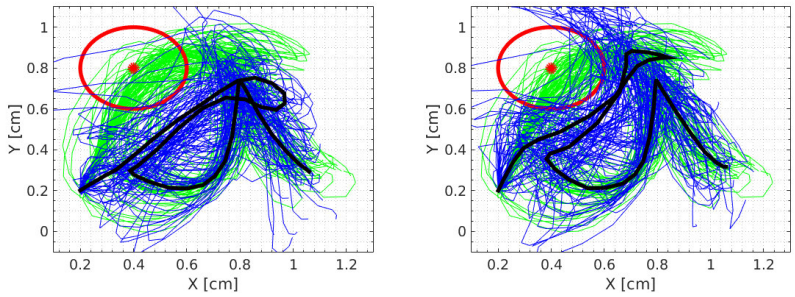
(c) Total average reward

Figure 3.20: The average trajectory reward reveals that the solution converges towards a trade off between staying close to the demonstrations and obstacle avoidance for all obstacle settings

As in the task space example the optimization converges as the cost goes towards zero. A trade off is found between minimizing the deviation from the demonstrated distribution and avoiding the obstacle. The results show again that the algorithm avoids the obstacle and at the same time stays close to the given demonstrations.
Figures 3.21 and 3.22 reveal the effects of the windows size.

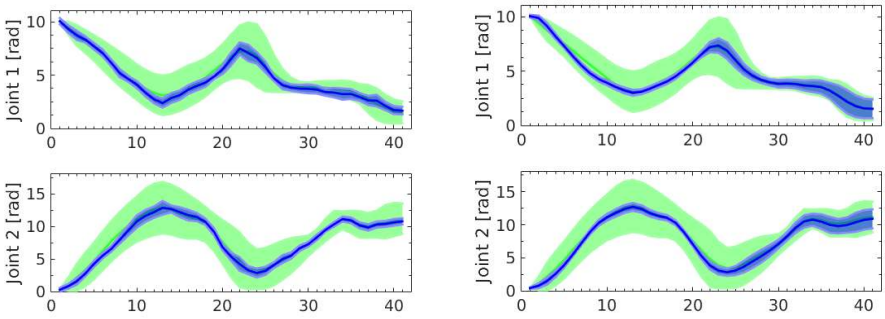


(a) 41/41 window (b) 21/41 window

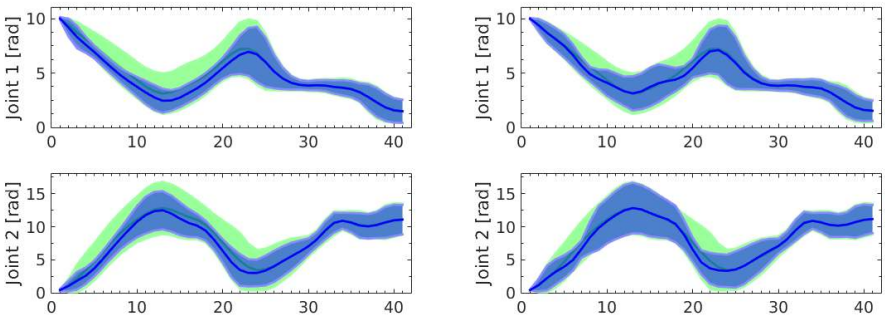


(c) 11/41 window (d) 3/41 window

Figure 3.21: Choosing a smaller windowsize results in local obstacle avoidance, but comes with a tradeoff of less correlated full trajectories.



(a) 41/41 window (b) 21/41 window



(c) 11/41 window (d) 3/41 window

Figure 3.22: The influence of different windowsizes on the shape of the optimized trajectories are shown for the single degrees of freedom.

A smaller window size causes the trajectories to avoid the obstacle locally but the global trajectories tend to get more uncorrelated as illustrated by Figure 3.21 (d). The influence of the window size is further investigated for the single degrees of freedom in Figure 3.22. It shows how the window size influences the solution. A small window size of 3/41 results in trajectories that are only modified at neighbourhood of the obstacle. On the other hand those solutions are less smooth, as they only capture correlation between closely neighbouring points.
In advantage to the task space setting the obtained solution can be directly executed on the robot and is guaranteed to be feasible as the joint limits and task space constraints where already considered during the optimization. Figure 3.23 illustrates how the robot follows a trajectory.

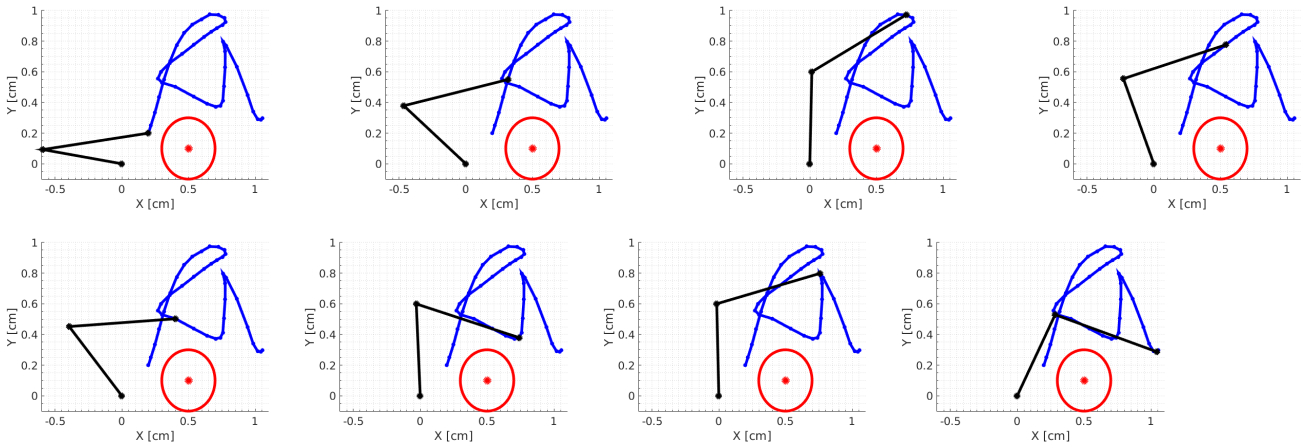


Figure 3.23: Following the optimized trajectories the robot is able to redraw the letter and avoid the obstacle.

Solutions of the problem in task and joint space offer their own advantages and disadvantages. While task space is more intuitive for humans and the constraints might be easier to specify joint space can help to overcome feasibility issues.
However the results of the toy examples revealed the algorithms ability to optimize trajectories in both applications.

4 Connection to Probabilistic Movement Primitives

This section connects the proposed algorithm to the existing framework of Probabilistic Movement Primitives (ProMPs) [3]. A main drawback of Probabilistic Movement Primitives is that the framework does not directly deal with motion planning problems, such as obstacles in the robot's workspace, which might not have been present in the demonstrations. Applying the proposed methods on a set of given demonstrations provides the advantage to avoid obstacles as well as to use the benefits ProMPs, such as conditioning on different via points.

4.1 Main Structure

Common trajectory optimization methods reduce their output to a single trajectory. This makes it difficult to consider their solutions for probabilistic methods, such as the framework of ProMPs. On the other hand, other methods such as belief state planning consider uncertainty in their output but do not connect to demonstrations, which are commonly used to create movement primitives.
This chapter presents a way to connect a trajectory optimization method with ProMPs to achieve generalizable human-like robot motions with the ability to avoid obstacles. Figure 4.1 shows the main workflow of the method.

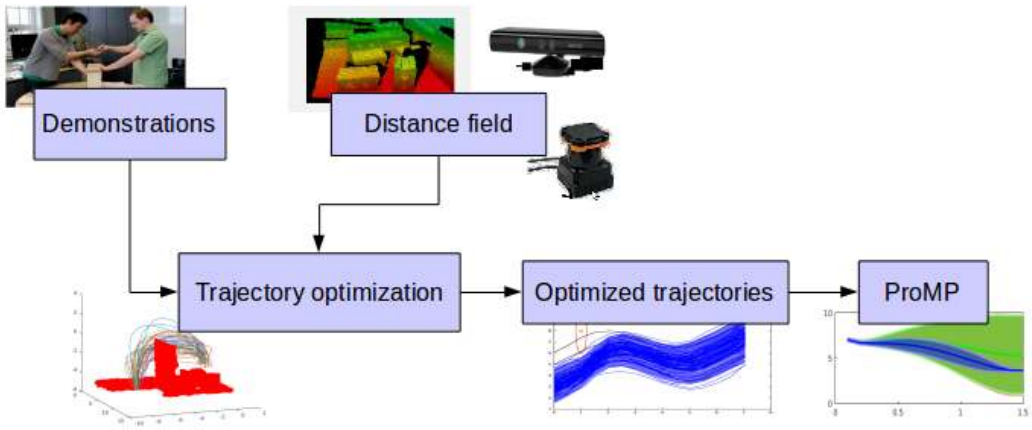


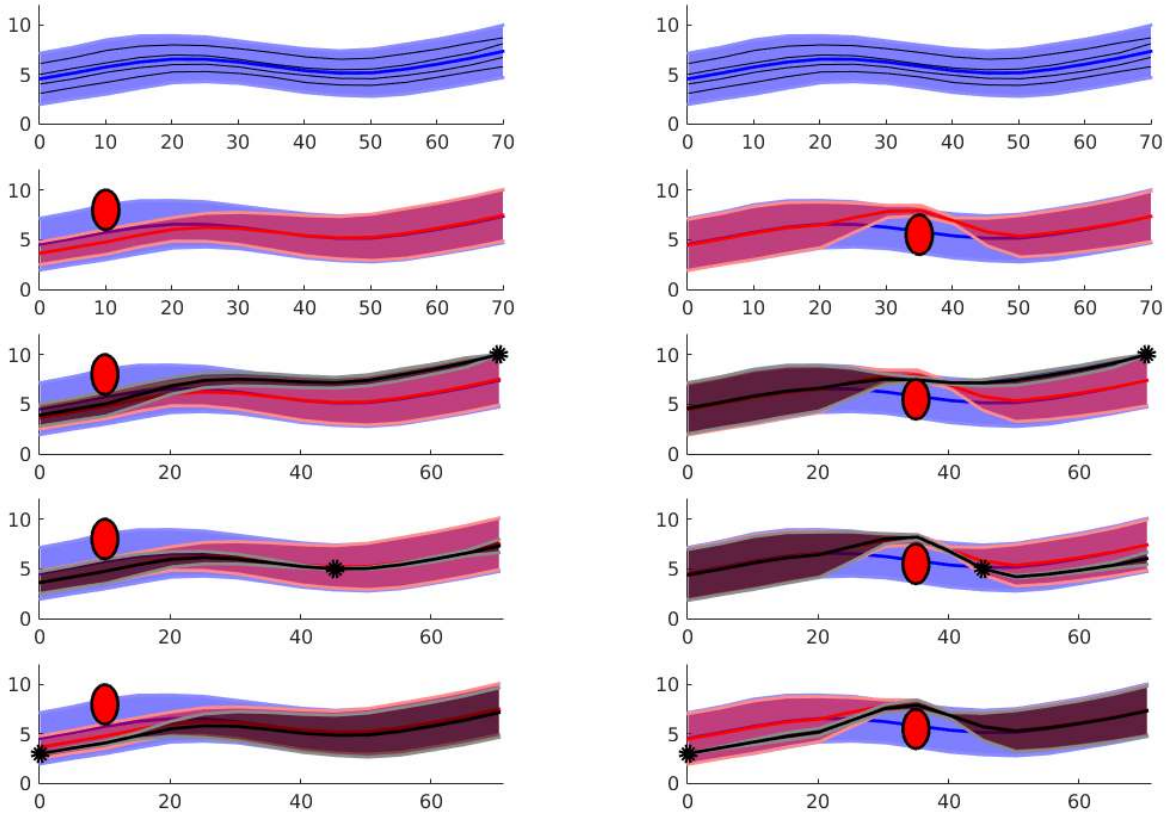
Figure 4.1: The proposed trajectory optimization uses a KL-metric to minimize deviation from the demonstrations and a signed distance field to avoid obstacles. Subsequently, a ProMP is learned on the optimized trajectories.

The trajectory optimization considers a KL metric to the demonstrated distribution as well as an Euclidean distance to obstacles based on a precomputed signed distance field. For an application on a real robot this distance field can be obtained from 3D depth sensors, such as laser scanners or a Kinect camera. Maximum likelihood is used on the optimized trajectory distribution to compute a ProMP, which imitates the demonstrations and moreover avoids the obstacles. This ProMP provides the advantage to efficiently adapt the motion for different via points, by using conditioning, as explained in 2.5.
The proposed method generalizes human demonstrations in two ways: First, it is adaptive to obstacles, which were not present during the demonstrations, and second, it is able to modify the motions e.g for different start or end points.

4.2 Obstacle Avoidance and Probabilistic Movement Primitives on 1D Example

While different ways for obstacle avoidance were presented in trajectory optimization and parametrized representations, such as Dynamic Movement Primitives, no clearly defined method exists for motion planning with

ProMPs. A naive way of avoiding obstacles is conditioning on via points, defined at the vicinity of the obstacle. This restricts the solution to the chosen vicinity. Moreover, in case of multiple or complex obstacles it is not trivial to compute these via points. Furthermore, conditioning on via points defined around the obstacles leads to the collapse of the variance in those regions. Applying the proposed algorithm in connection with ProMPs allows to use more principled ways of collision avoidance. For example, it becomes straight forward to incorporate Euclidean distances and signed distance fields. As a result, the variance does not collapse and the robot imitates the demonstrations in all regions unaffected by the obstacles.



(a) (b)

Figure 4.2: Connecting the proposed trajectory optimization to ProMPs allows for both, obstacle avoidance and conditioning on via points. The original ProMP (blue) is deformed depending on the obstacle position in (a) and (b). The optimized ProMP (red) can then be used for conditioning on via points (black).

Figure 4.2(a) and Figure 4.2(b) show results for a 1D toy example with different obstacle positions (red) and via points (black). They illustrate the ProMP of the demonstrations (blue) and the ProMP after the trajectory optimization for the present obstacle (red). Subsequently, this optimized ProMP is used for conditioning on various via points (black). The plots demonstrate that with the proposed method it becomes feasible to avoid the obstacle and at the same time preserve the distribution of the demonstrations. Using ProMPs and conditioning it is possible to adapt the optimized trajectories to different via points.

4.3 Robot Experiments in 3D

The ability to imitate human demonstrations and to adapt motions to changing workspace setting is an important requirement for future robotic applications. While the framework of ProMPs provides conditioning on different via points, this thesis extends it to additionally avoid obstacles that were not present during the demonstrations. The

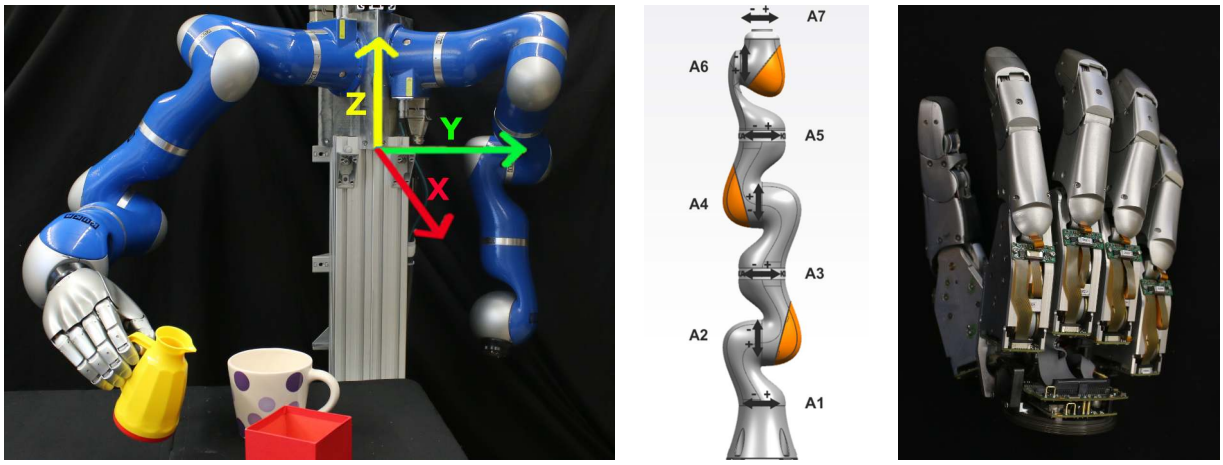
method is evaluated for human demonstrations recorded with a motion capturing system and a seven degree of freedom robotic arm.
This sections presents the hardware components, the experimental set-up, and the results obtained in the experiments.

4.3.1 Hardware

In order to evaluate the algorithm on a real robot this thesis uses a motion capturing system, a depth sensor and a seven degree of freedom (7-DoF) KUKA lightweight arm. This section gives a brief overview of those hardware components.

Robot

The platform used for evaluation of the algorithm consists of a 7-DoF lightweight arm equipped with a DLR HIT hand. The platform is shown in Figure 4.3 (a). The coordinate system is shown with x-axis in red, y-axis in green and z-axis in yellow. The kinematics of the arm are shown in more detail in Figure 4.3 (b). The arm's kinematic structure attempts to approximate the humans kinematics, leading to similar manipulation range and redundancy. Communication with the robot runs at 1 kHz and it can be torque as well as position controlled. Moreover, the robot allows for kinaesthetic teaching, as the arm is active compliant and capable of gravity compensation.



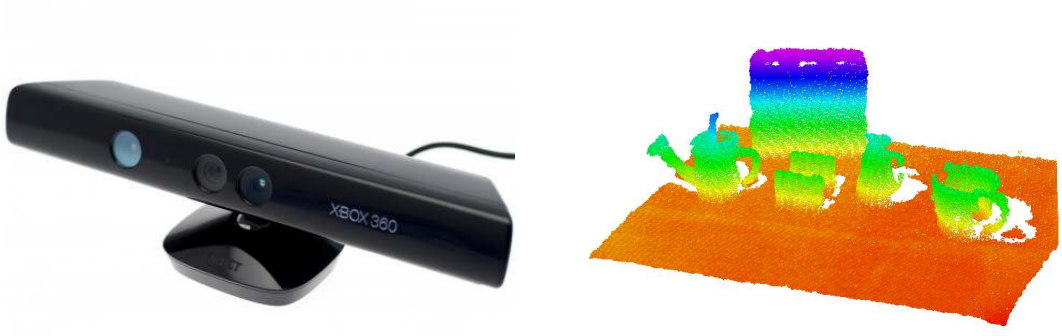
(a) (b) (c)

Figure 4.3: (a) The algorithm is evaluated on the robot Darias, a two armed manipulation platform, equipped with a KUKA lightweight arm and a DLR HIT hand. The coordinate system is illustrated with x-axis in red, y-axis in green and z-axis in blue. (b) Kinematic redundancy in the arm allows for wide manipulation range. (c) The DLR HIT Hand is equipped with five fingers with joints for proximal and distal flexion as well as spread.

The arm is equipped with a DLR Hit hand, as shown in Figure 4.3 (c). The hand is equipped with five fingers, which are controlled using joint impedance control. The compliance of the fingers and arms ensure sufficient reaction to minor uncertainties during the execution of a task. However, the robot is mounted to a fixed base and its workspace is, therefore, constrained to the reach of its arms.

Sensors

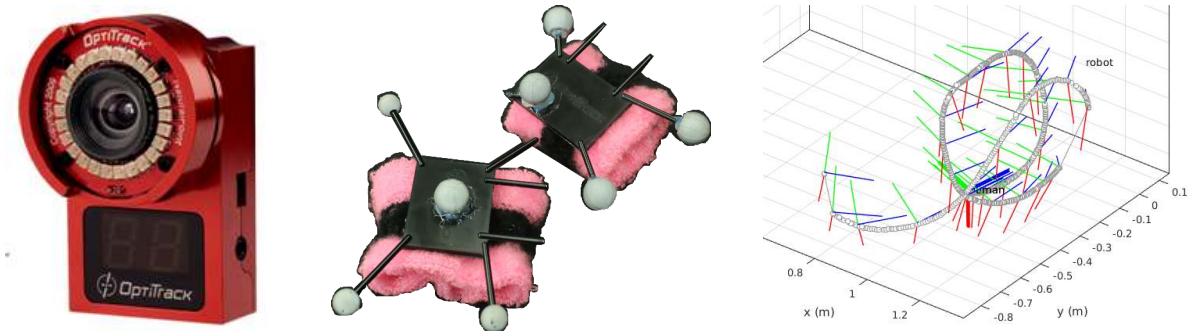
A reliable model of the environment, including obstacles, forms the basis for motion planning. For the robot experiments a Microsoft Kinect camera, as shown in Figure 4.4 (a), was used to obtain 3D point clouds. This camera is able to record depth information in addition to RGB images. This combined sensing is called RGB-D. To record depth sensing an infrared laser projector is combined with a monochrome complementary metal oxide semiconductor (CMOS) sensor. Prime sense technology enables the camera to calculate depth information from the distortion of an irregular pattern that is projected in front of the camera by the laser. More information about



(a) (b)

Figure 4.4: (a) Prime sense technology enables the Microsoft Kinect camera to record RGB-D images. (b) The depth sensings from the Kinect camera are used to obtain 3D point clouds of the robot's workspace.

this technology can be found in [33]. Figure 4.4 (b) shows a 3D point cloud, recorded with the Kinect camera. In this thesis a signed distance field is created out of the obtained 3D point cloud. To compute the distance field the implementation of Moveit's propagation distance field [34] is used. The output of this distance field is post processed as a matrix in Matlab to perform fast distance queries.
An OptiTrack system records human demonstrations. Figure 4.5(a) shows the sensors of this system. In order to track motions different markers as illustrated in Figure 4.5(b) can be used to obtain Cartesian trajectories as illustrated by Figure 4.5(c).



(a) (b) (c)

Figure 4.5: (a) The motion capturing system OptiTrack records motions at 120 Hz. (b) OptiTrack markers can be used to track the wrist position of the demonstrator. (c) The recorded trajectories offer x, y and z position of the end effector for the subsequent trajectory optimization .

The OptiTrack system records at 120 Hz. It is possible to track all joints of a human body with a marker suit. However, in this thesis the focus is only on tracking end effector positions. The joint configurations of the robot can be obtained by applying inverse kinematics on the Cartesian trajectories

4.3.2 Experiments

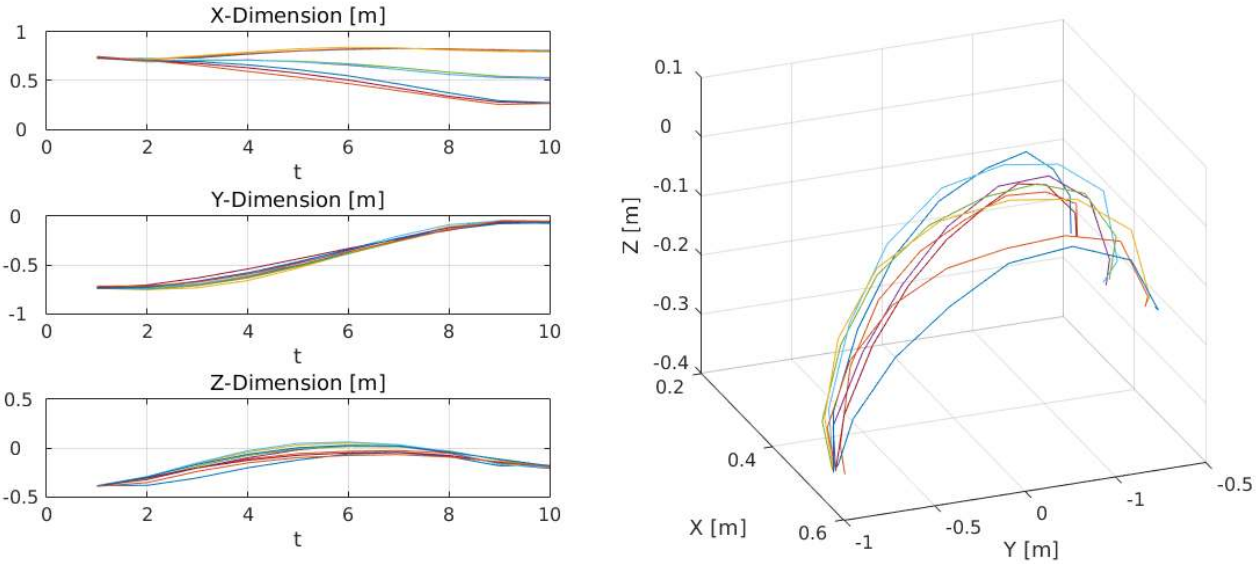
Extracting motions out of human demonstrations provides an intuitive way to teach new tasks to a robot. However, the robot must be able to adapt the demonstrated motions to different workspaces, such as changing via points or obstacles that were present in the demonstrations.
In order to evaluate the proposed method a pick-and-place scenario is considered, as displayed in Figure 4.6. The setting is shown in a more detailed view in Figure 4.6 (b). The setup includes a ball (6) and different delivering

positions (1) - (5). During the experiments a human performed nine deliveries of the ball at different box locations.

(a) (b)

Figure 4.6: (a) The experimental setup consists of a ball and 5 boxes, placed on a table. (b) The robot is supposed to pick the ball (6) and place it in one of the boxes (1) - (5).

Three demonstrated motions each end up in box (1), (3) and (5) as illustrated in Figure 4.7. It should be noted that there are no demonstrations for box (2) and (4). During the demonstrations an OptiTrack marker is placed on the demonstrator's wrist. The marker positions of the human wrist offer 3D Cartesian trajectories. The orientation is considered fixed and the focus is on optimizing x-, y-, and z-coordinates. The recorded trajectories are shown in Figure 4.7. The figure illustrates the motions in the single dimensions x, y and z as well as the 3D motion of the nine demonstrations.



(a) (b)

Figure 4.7: The demonstrations consist of three deliveries for box (1), (3) and (5). (a) shows the demonstrated trajectories isolated for x-, y-, and z-dimension and (b) shows the demonstrated trajectories in 3D space.

For each of the demonstrated boxes three repetitions were recorded to reveal variance in the trajectories. The demonstrations show variance in particular in x- and z-dimension and can be used to train a ProMP. The human demonstrations for the three boxes are shown again in Figure 4.8 from front and top view.

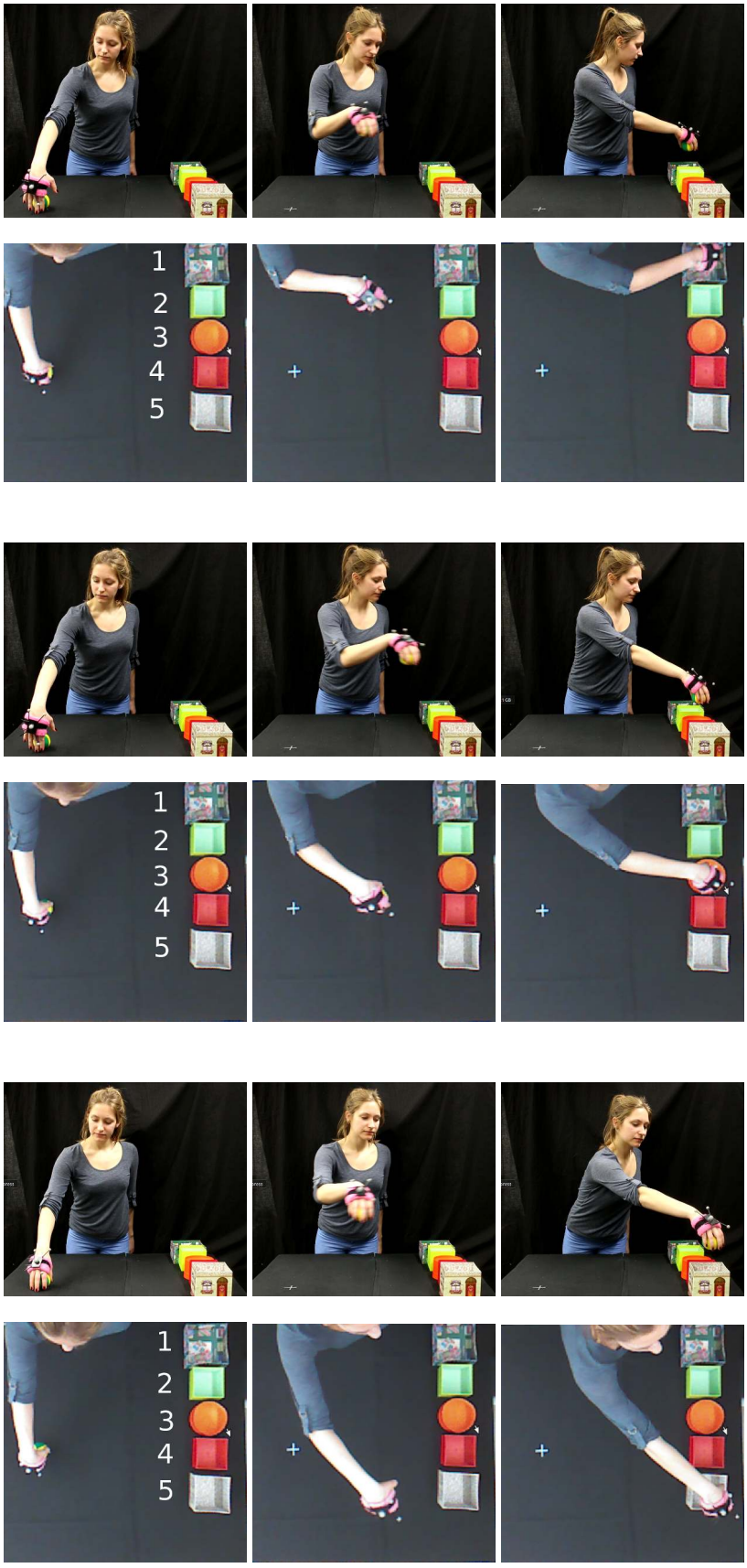
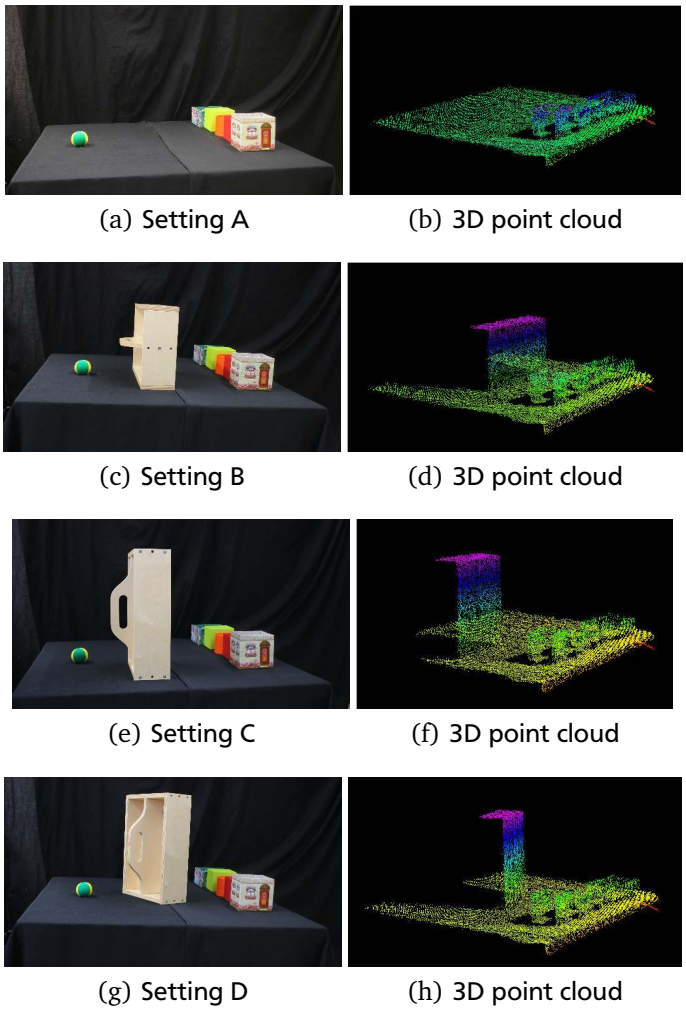


Figure 4.8: The demonstrations are recorded three times for the boxes (1), (3) and (5). There are no deliveries to box (2) and box (4) in the demonstrations. During the demonstrations an OptiTrack marker is placed on the demonstrators wrist to obtain end effector trajectories.

The proposed connection of the new algorithm and ProMPs is evaluated in four different workspace settings. The method incorporates the human demonstrations and was evaluated on different obstacle configurations and different desired delivery positions. Figure 4.9 illustrates the four workspace settings and the corresponding 3D point clouds obtained by a Kinect camera.
The experiments evaluate if it is possible to create robot motions out of human demonstrations that can generalize in respect to two main aspects. First, the motions should be adaptive to different via points, such as the boxes, which can be achieved using the ProMP framework. Second, the motions should adapt to different obstacle positions, which can be achieved by incorporating the proposed trajectory optimization.



(a) Setting A (b) 3D point cloud

(c) Setting B (d) 3D point cloud

(e) Setting C (f) 3D point cloud

(g) Setting D (h) 3D point cloud

Figure 4.9: The proposed method is evaluated in four experiments. The pictures show the setup and the 3D point cloud provided by the Kinect camera. (a) and (b) show setting A, where no additional obstacle is placed in the workspace. (c) and (d) show setting B, where an obstacle placed between the ball location and the boxes. (e) and (f) show setting C, where the obstacle only traverses the demonstrations for box (5) and (g) and (h) show setting D, where the obstacle is put on its side and placed in the middle of the table.

We consider four different obstacle settings. First, a setting with no obstacles is evaluated. This setting allows to observe how the robot reproduces the demonstration, when not constrained by additional obstacles. Next, three different obstacle positions and heights are compared and evaluated for different desired box positions. All four settings A,B,C and D are depicted in Figure 4.9.

4.3.3 Results

The section evaluates the method on four different obstacle settings (A,B,C, D). It discusses the results of experiments for different desired delivery boxes and the evaluation of the optimized trajectories on the robot.

Setting A

In the first setting no obstacle was added in comparison to the demonstration set and deliveries to all box positions (1) - (5) were successful. This section presents the results for box (1) and box (4) as examples in more detail. Figure 4.10 illustrates the resulting distributions for the three degrees of freedom. It shows the ProMPs of the demonstrations (red), the ProMPs after the trajectory optimization (green) and the ProMPs after conditioning on a certain box (blue). After the trajectory optimization the ProMPs recovered the demonstrated distribution. The optimization converges to the demonstrated distribution even though the policy was initialized with an uncorrelated, diagonal, covariance matrix. Using conditioning on the ProMPs it was possible to adapt the solution to different end points, such as positions of different boxes. The proposed method was able to generate trajectories for both, boxes that were demonstrated by the human, such as box (1), and boxes that were not demonstrated, such as box (4).
The plots show that the optimized distributions in y- and z-dimension match the demonstrated distributions completely and also the optimized distribution in x- dimension converges towards the demonstrations. Moreover, it is notable that the trajectory optimization preserves variance which is essential for the subsequent conditioning. The conditioning is illustrated in blue in Figure 4.10.

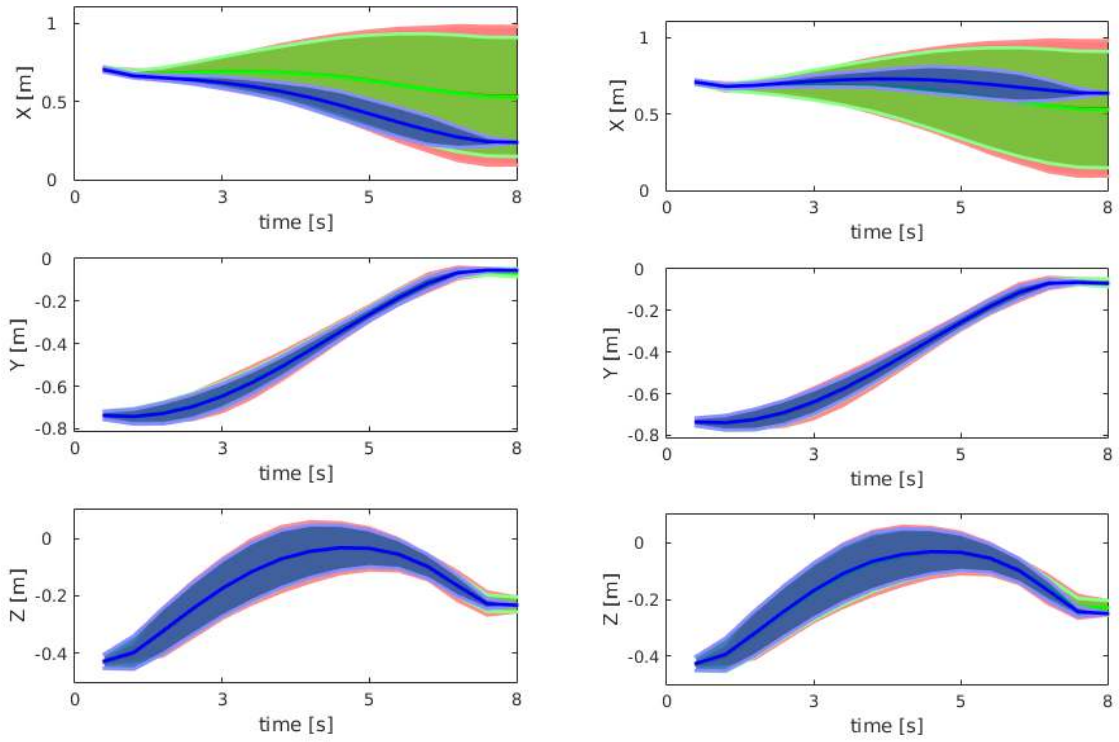


Figure 4.10: The ProMP from the demonstrations (red) is modified by the trajectory optimization (green). Using conditioning on the optimized ProMP the motions adapt to different boxes (blue). (a) shows the results from setting A for box (1) and (b) shows the results from setting A for box (4). For y- and z-dimensions the distributions overlay as the optimized trajectories converge towards the demonstrations.

The ProMP framework can adapt the movement primitive to box 1 and also for box 4, which was not demonstrated by the human. In particular, for the x-dimension it reveals how the shape of the ProMP adapts to different end-points. Using inverse kinematics the optimized trajectories can be transformed to joint space and are executed on the 7-DoF robotic arm. Figure 4.11 shows snapshots of the resulting robot motions for box 1.

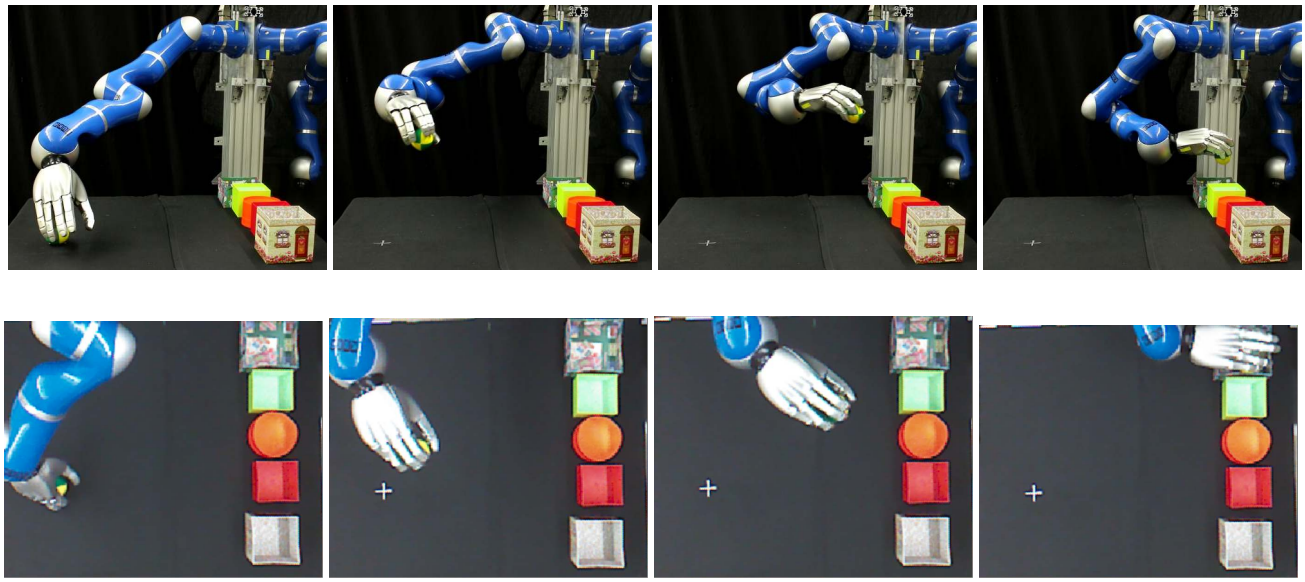


Figure 4.11: In the first setting no obstacle is added to the demonstrated setup. The robot is able to sucessfully deliver the ball to box (1), via conditioning on the optimized ProMP . The resulting motions are shown from frontal and top view.

The motions are shown from top as well as from frontal view. Figure 4.11 illustrates how the motions match the motions that were demonstrated and that the robot successfully delivers the ball to box 1. Figure 4.12 reveals that also the motions for box 4, which was not inside the demonstrations, match the demonstrated movements.

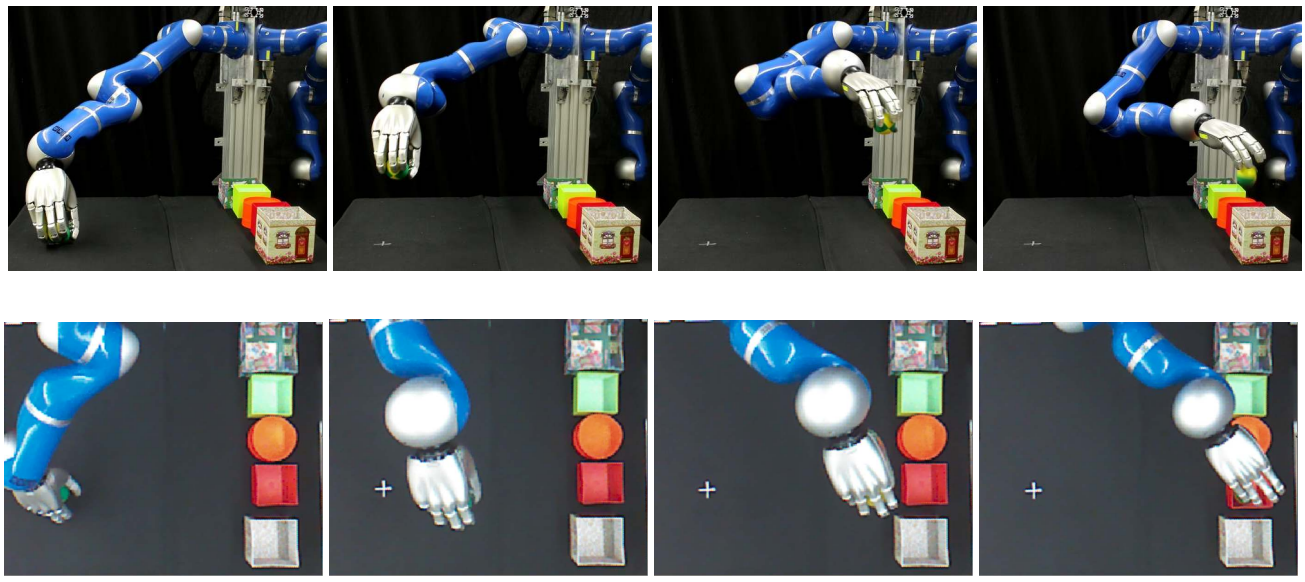


Figure 4.12: Adapting the optimized ProMP enables the robot to deliver the ball to box (4) which was not demonstrated by the human. The resulting motions are shown from frontal and top view.

Moreover, comparing Figures 4.11 and 4.12 to the human demonstrations it is evident that the motions generated

by the trajectory optimization look similar to the motions of the human demonstrator.
The boxes and the ball were not tracked by any sensors. The positions of the ball and the boxes were obtained from the demonstrated trajectories. Note that, the positions of the boxes (2) and (4) were inferred using the ProMP trained on boxes (1), (3) and (5).

Setting B

In setting B a wooden box is placed between the location of the ball and the boxes. The obstacle covers the full x-range of the demonstrations but leaves free space in z-dimension. Experiments were successfully performed for all boxes and this section presents examples for box 2 and box 5 more detailed.
Figure 4.13 compares the ProMPs from the demonstrations, after the trajectory optimization and for conditioning on the boxes 2 and 5. The figure shows that the trajectory optimization mainly deformed the shape of the distribution in z-dimension. The ProMPs for x- and y-dimension stay close to the demonstrations. Moreover, Figure 4.13 shows that the conditioning after the trajectory optimization is still avoiding the obstacle. This can be seen on the example of conditioning on box 5 and box 2. It should again be noted that box 2 was not part of the demonstrations.

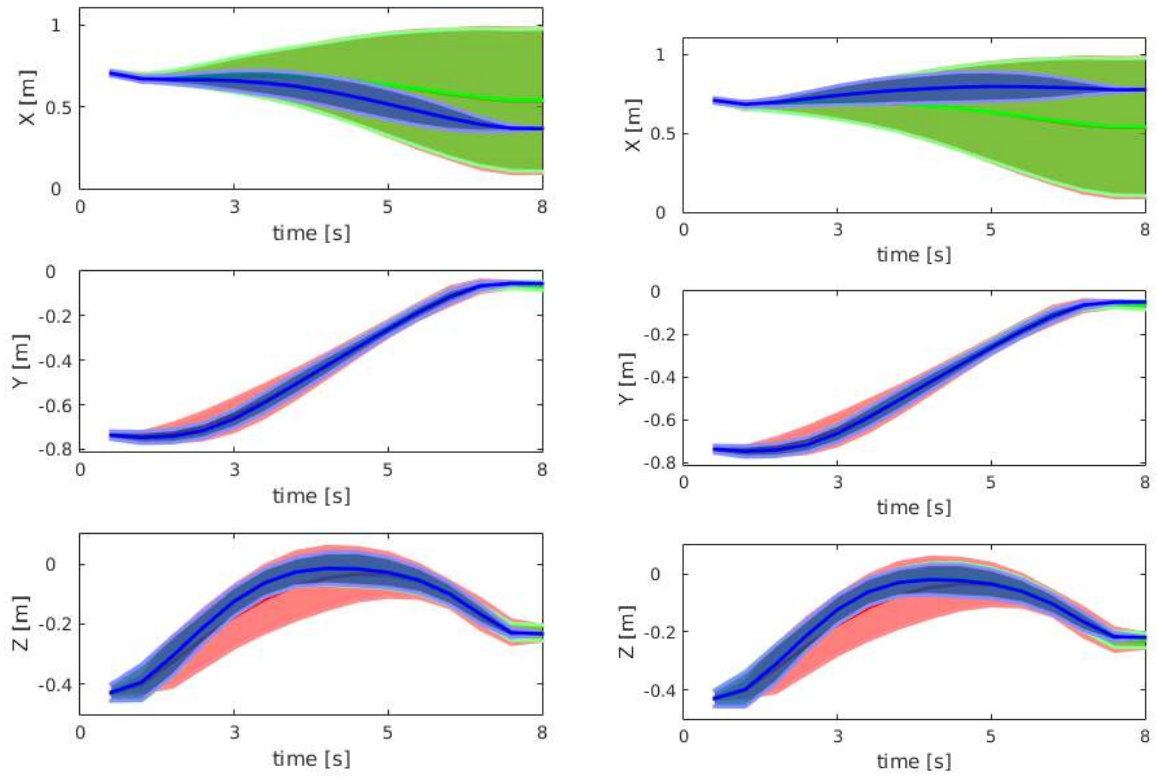


Figure 4.13: The original demonstrations (red) is modified by the trajectory optimization (green), using a ProMP. Conditioning on the optimized ProMP adapts the motions for different boxes (blue). (a) shows the results from setting B for box (2) and (b) shows the results from setting B for box (5). For y-dimension the distributions overlay as the optimized trajectories converge against the demonstrations.

Using inverse kinematics the motions are executed on the robot. Figure 4.14 illustrates the motions for box 5.

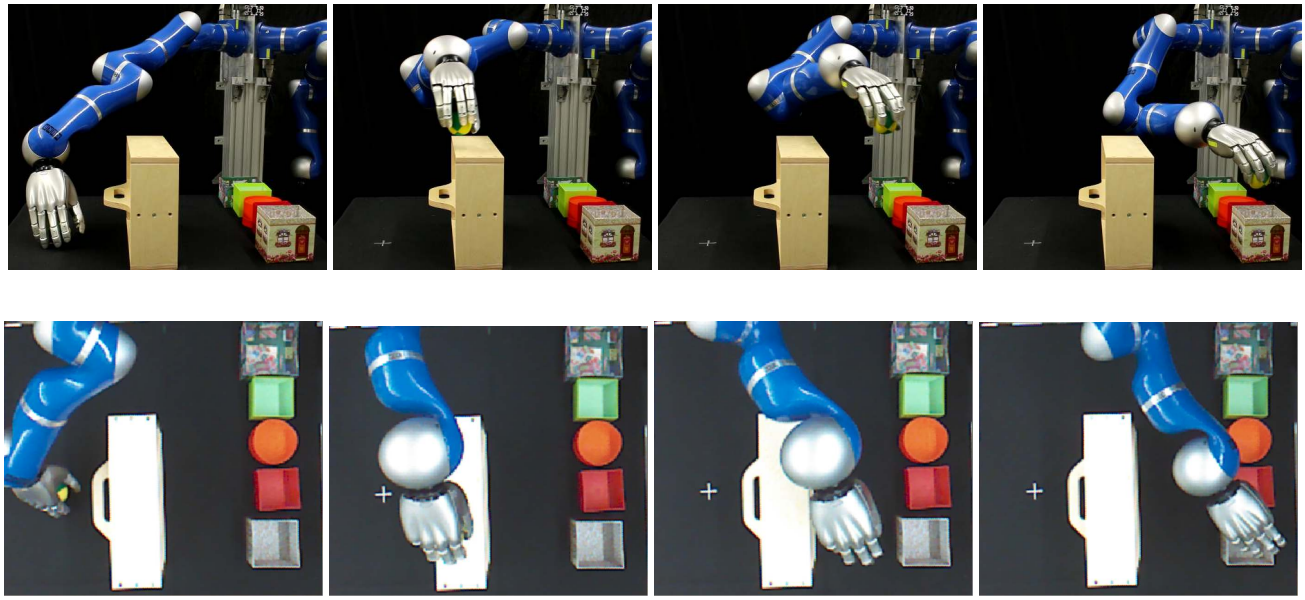


Figure 4.14: In the second setting an obstacle is added between the ball and the boxes. Via conditioning on the optimized ProMP the robot is able to sucessfully deliver the ball to box (5). The resulting motions are shown from frontal and top view.

In particular the frontal view shows how the obstacle is avoided in z-dimension and that the position of the box is successfully reached in the end. Comparing the top view to the demonstrations it also reveals that the movement in x-dimension is not affected by the presence of the obstacle.

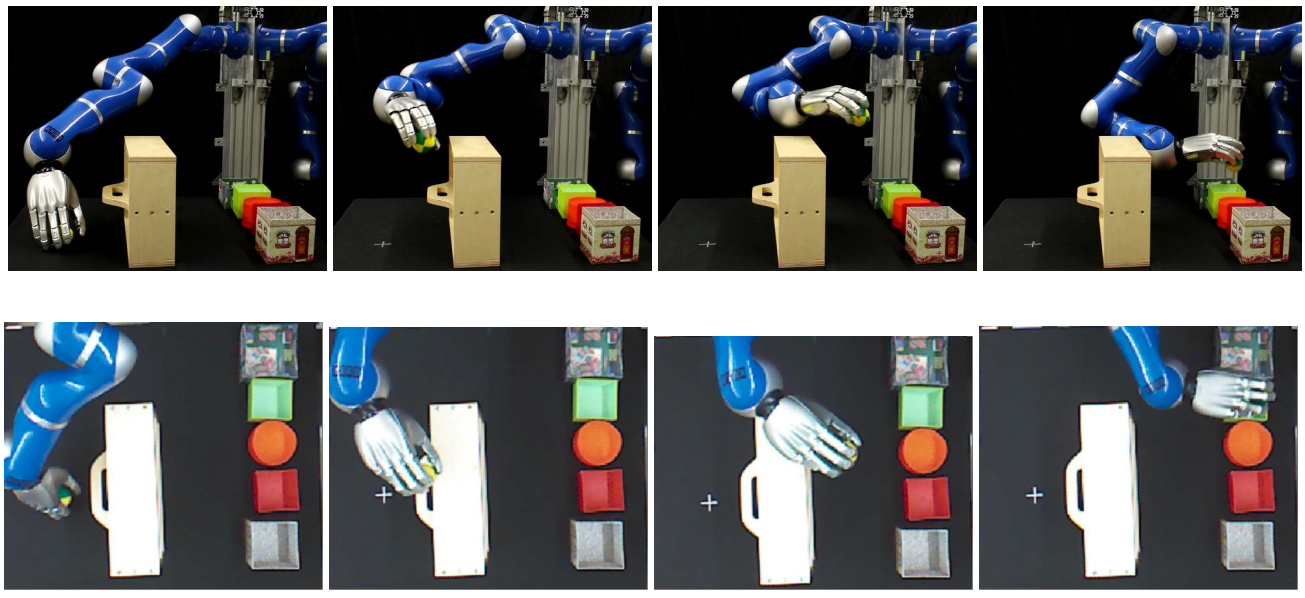


Figure 4.15: Adapting the optimized ProMP enables the robot to deliver the ball to box (2) which was not demonstrated by the human and to avoid the obstacle at the same time. The resulting motions are shown from frontal and top view.

Figure 4.15 shows the results of the robot motions for box 2. It reveals the main advantage of the proposed

algorithm in combination with the ProMP framework: The robot was able to avoid the obstacle and at the same time delivered the ball to a box that was not explicitly shown during the demonstrations.
Similar to experiment 1, a Kinect camera was used to obtain a signed distance field of the workspace. The locations of the ball and the boxes where extracted out of the demonstrations.

Setting C

For setting C the wooden box was placed in the frontal area of the table. It was turned on the side to cover even more space in z-direction. This settings does not constrain deliveries to box 1 and 3 but traverses the demonstrations for box 5. In the experiments the robot delivered the ball successfully to all five boxes. This section presents the results for box 2 and box 5.
Figure 4.16 shows the resulting ProMPs after the trajectory optimization. In opposite to the second experiment the trajectory distribution in x-direction is deformed most by the optimization whereas the one in z converges against the demonstrations. As in the the previous case, the trajectories in y-dimension remain unaffected by the obstacle. Moreover, conditioning was evaluated for all boxes and the results are shown for boxes 2 and 5. As in the experiments of setting B the conditioning of the ProMP framework enables the optimized trajectories to different boxes while avoiding the obstacle. This is demonstrated for boxes shown and not shown during the human demonstrations.

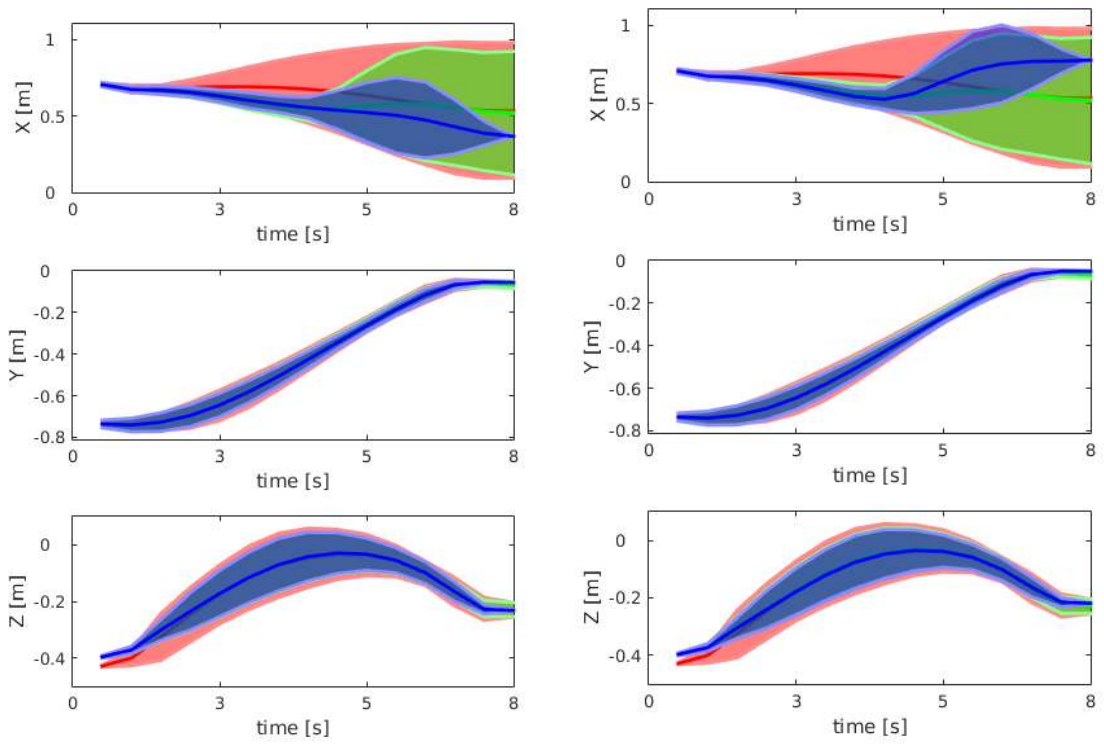


Figure 4.16: The ProMP from the demonstrations (red) is modiefied by the trajectory optimization (green). Using conditioning on the optimized ProMP the motions adapt for different boxes (blue). (a) shows the results from setting C for box (2) and (b) shows the results from setting C for box (5). For y-dimension the distributions overlay as the optimized trajectories converge against the demonstrations.

Figure 4.17 illustrates the resulting movements of the robot for box 2. The motion stays close to the demonstrated trajectories as the obstacle is only blocking the front part of the workspace.

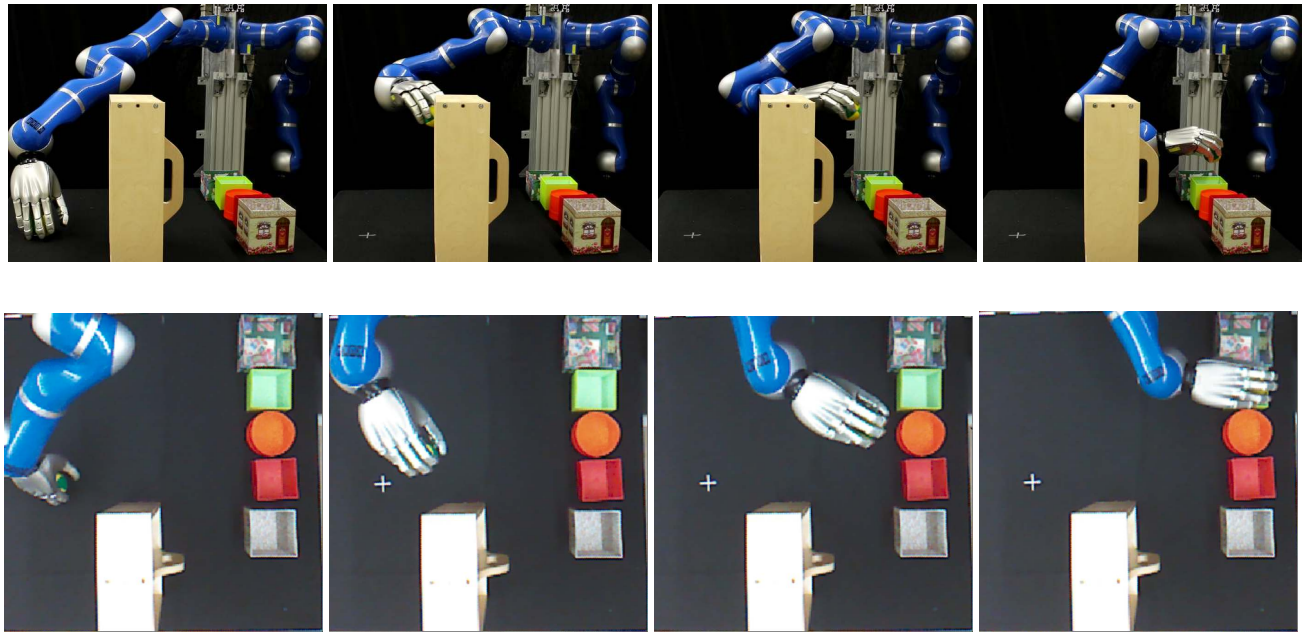


Figure 4.17: In the third setting the demonstrated trajectory to box 5 is traversed by an obstacle. Via conditioning on the optimized ProMP the robot is able to sucessfully deliver the ball to box (2), which was not demonstrated by the human. The resulting trajectory is not constrained by the obstacle and stays close to the demonstrations. The motions are shown from frontal and top view.

In comparison, Figure 4.18 shows the motions for box 5. In setting C the demonstrated trajectories for box 5 are fully traversed by the obstacle. Therefore, the optimized trajectories avoid the obstacle as they circumvent it in x-direction.

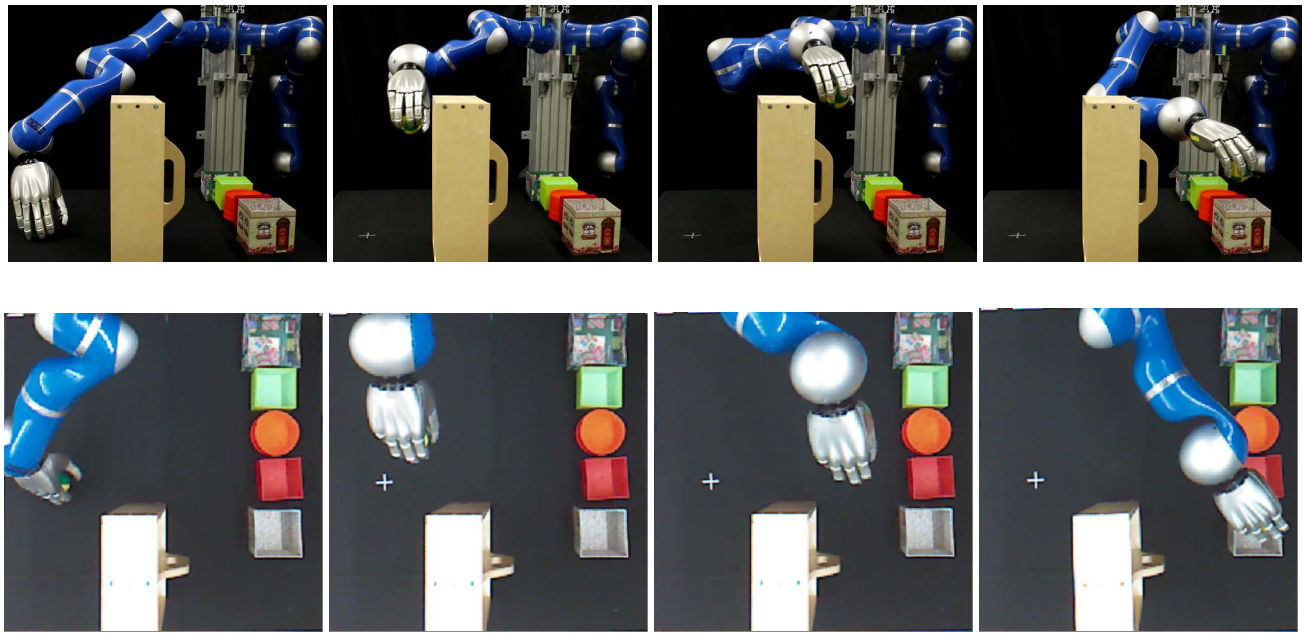


Figure 4.18: Optimizing the ProMp the robot is able to avoid the obstacle and deliver the ball to box (5). The resulting motions are shown from frontal and top view. In particular, the top view illustrates how the robot circumvents the obstacle in x-direction.

The experiments for this third setting indicate that the optimization extracts the dimension which provides obstacle avoidance and allows for minimal deviation from the demonstrated trajectories. Moreover the experiment shows

that, depending on the delivering location, not all trajectories are modified, but only those which are traversed by the obstacle.

Setting D

In the last setting the wooden box is turned to the side and placed in the middle between the ball and the boxes. The obstacle was not placed orthogonal to the y-axis. This way it is not possible for the robot to avoid the obstacle by modifying only one dimension. This section presents the experiments performed for box 1 and box 4.
Figure 4.19 shows the resulting ProMPs after the trajectory optimization. It reveals how mainly x- and z-direction are affected by the presence of the obstacle. In particular in z-dimension the trajectories are modified in the region of the obstacle. The plots also reveal the effect of the window size. The obstacle has high influence in its neighbourhood, but the beginning and the end of the trajectories stay close to the demonstrations.
The results of the subsequent conditioning are shown for box 1 and box 4, as an example.

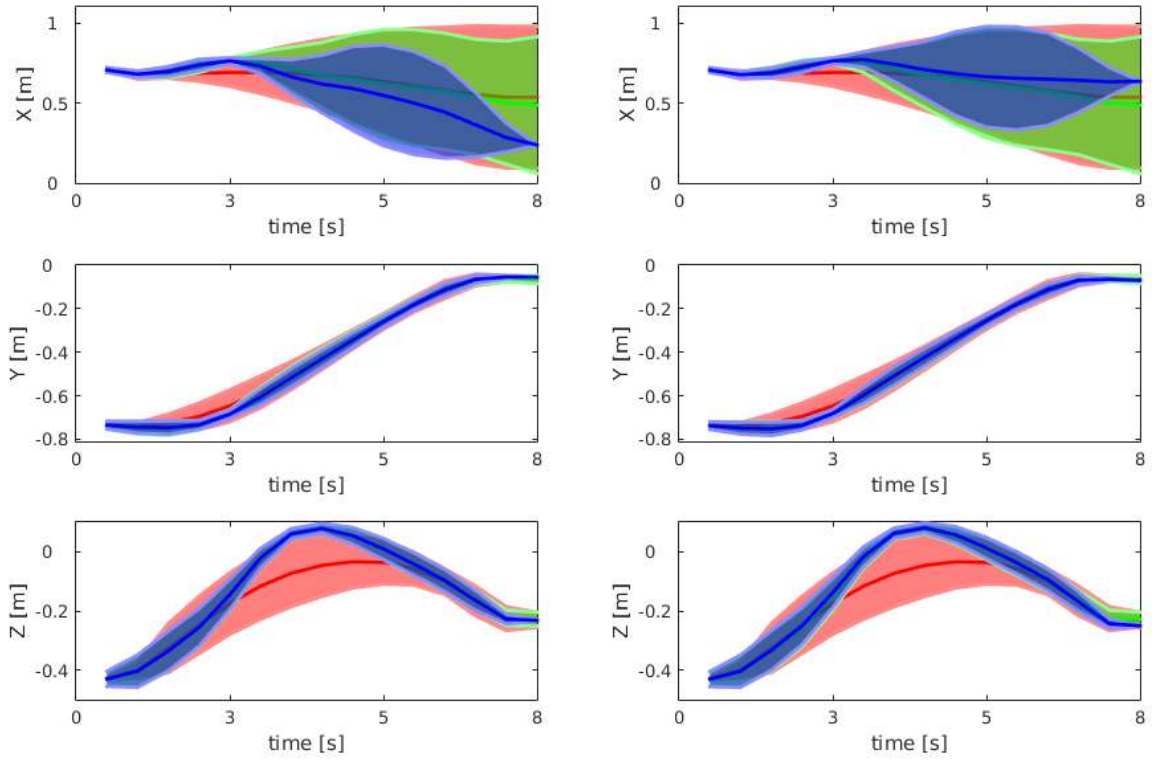


Figure 4.19: The ProMP from the demonstrations (red) is modified by the trajectory optimization (green). Using conditioning on the optimized ProMP the motions adapt for different boxes (blue). (a) shows the results from setting D for box (1) and (b) shows the results from setting D for box (4). For y-dimension the distributions overlay as the optimized trajectories converge towards the demonstrations.

The resulting motions for the robot for box 1 are illustrated by Figure 4.20. It shows how the motions are deformed in x- and z-direction to avoid the obstacle.

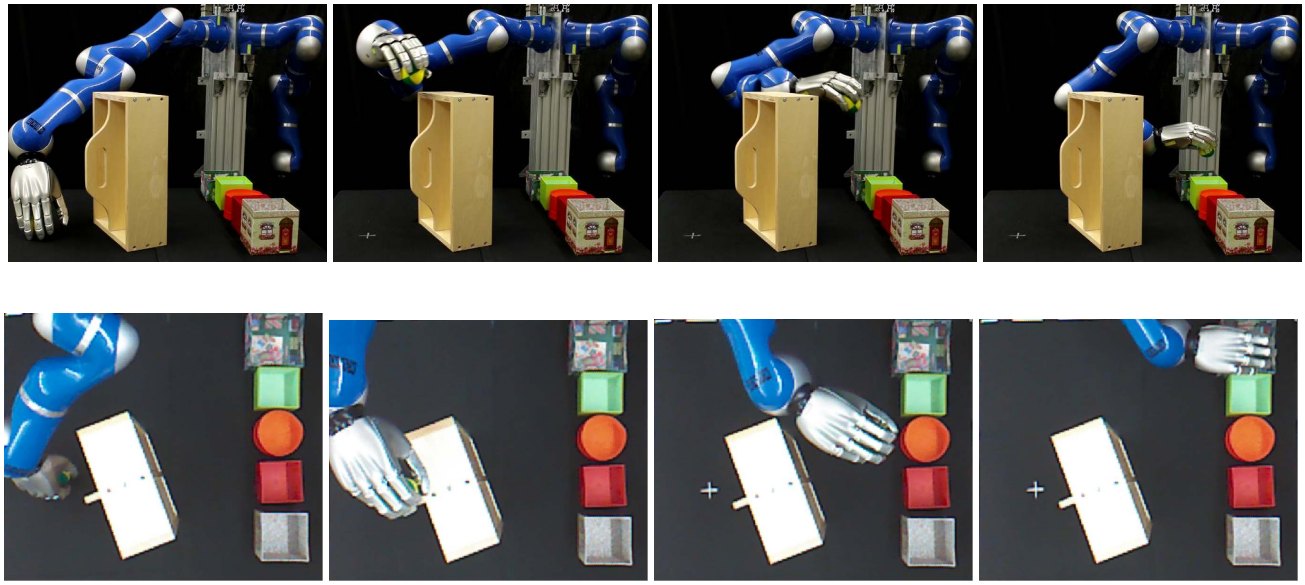


Figure 4.20: In setting D the obstacle is placed in the center of the table. Via conditioning on the optimized ProMP the robot is able to sucessfully deliver the ball to box (1) and avoids the obstacle in both, x- and z-direction. The resulting motions are shown from frontal and top view.

Figure 4.21 illustrates the robot motion for an end point that was not demonstrated by the human. It reveals how the optimized trajectories avoid the obstacle and are able to deliver the ball to box 4. Even though the motion in z-direction needs to be strongly adapted due to the presence of the obstacle the motion still looks human-like, when compared to the demonstrations.

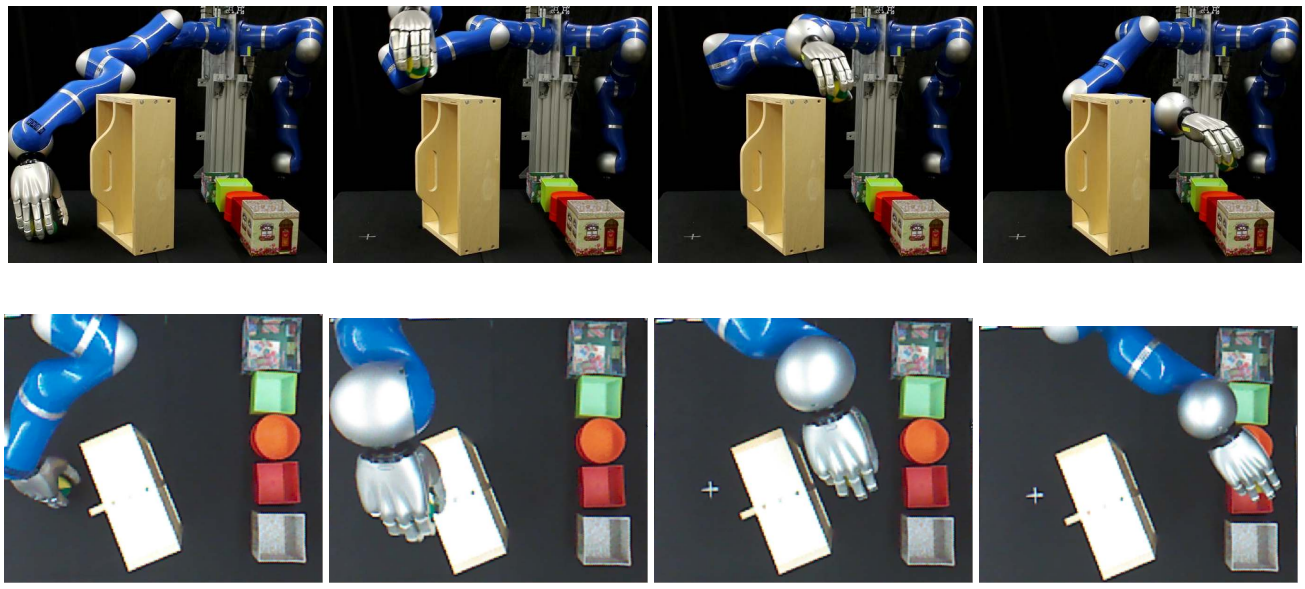
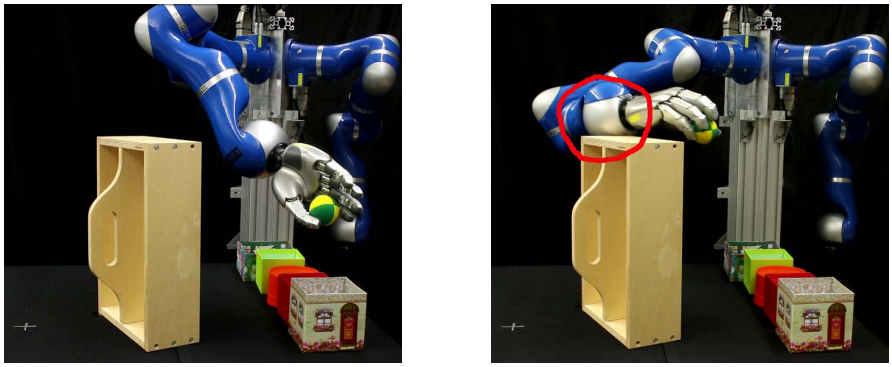


Figure 4.21: Adapting the optimized ProMP enables the robot to deliver the ball to box (4) which was not demonstrated by the human and to avoid the obstacle at the same time. The resulting motions are shown from frontal and top view.

This experiment showed that the trajectory optimization works in settings, where the obstacle can only be avoided in combined deformations for the single dimensions.
The experiments were evaluated for all boxes. In the case of box 5 the motion ended in a weird joint configuration as illustrated by Figure 4.22 (a).



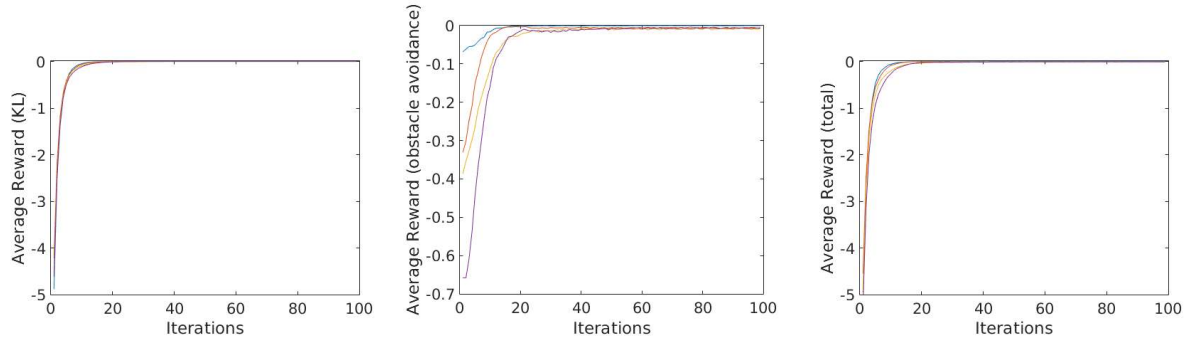
(a) (b)

Figure 4.22: The experiments with setting D revealed two shortcomings. The optimization in task space relies on the use of inverse kinematics. While this works well in the center of the workspace, it can end up with unexpected joint configurations at the workspace limits as shown in (a). Moreover, in this experiment only the end effector was considered for obstacle avoidance. This can lead to collision of other parts of the arm as shown in (b).

This joint configuration occurred due to the fact that the optimization in task space required the use of inverse kinematics to execute the trajectories on the robot. While inverse kinematics provides good solutions in the workspace center it faces limitations at the boarders of the workspace. In future experiments it should be investigated if optimizing the trajectories in joint space or null space optimization can avoid these issues.
For the experiments only the end effector was considered for obstacle avoidance. This can result in situations where the robot avoids the obstacle with the hand but collides with a part of the arm. Figure 4.22 shows such a situation, that occurred one time in the experiments. For future experiments it can be considered to approximate the geometry of the arm with bounding boxes as it was done in STOMP [14].

Summary of the Experiments

To evaluate the algorithm on a real robot application four different workspace settings were compared. A setting with no additional obstacle, one with an obstacle that constrained mainly x-dimension, one that constrained mainly z-dimension and one that constrained x- as well as z-dimension, were evaluated.
Figure 4.23 illustrates the reward function optimization for the four experiments.



(a) (b) (c)

Figure 4.23: The Optimization finds a tradeoff between minimizing the KL divergence to the demonstrated distribution and avoiding the obstacles. (a) shows the average reward for minimizing the deviation to the demonstrations, (b) shows the average reward for the distance to the obstacles and (c) shows the sum of both, which forms the total reward for the optimization.

For all experiments Figure 4.23 (a) - (c) show the rewards for the KL divergence to the demonstrations, the rewards for obstacle avoidance and the total rewards. The optimization converges to zero for all settings. For setting D the reward for obstacle avoidance is higher, as the obstacle is traversing a large range of the workspace at the beginning of the optimization.
For all experiments the trajectories were discretized with equidistant 15 points and the optimization used a 3/15 window. The distribution stays close to the demonstrations in all parts except the ones affected by the obstacles.
Figure 4.24 summarizes the resulting trajectory distributions of the four experiments. The figure shows the results of the obstacle avoidance and how different obstacle positions affect the direction, in which the algorithm tends to avoid the obstacle. In particular this is illustrated by setting B where the trajectories get mainly deformed in z-direction and setting C where they circumvent the obstacle in x-dimension.

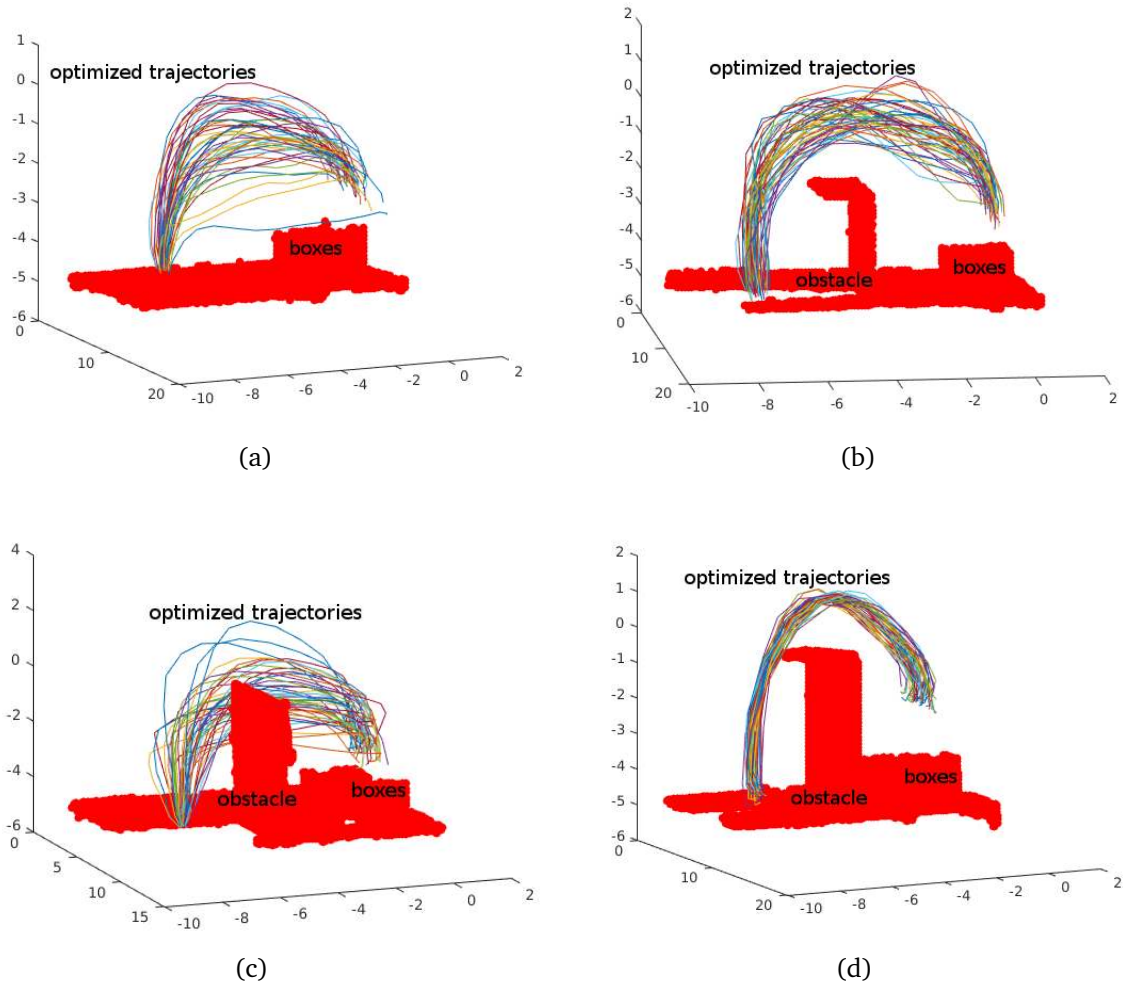


Figure 4.24: The proposed method optimizes the demonstrated trajectories depending on the workspace setting. (a) In the first experiment the distribution matches the demonstrations, as no obstacle is added to the workspace. (b) In the second experiment the trajectories avoid the obstacle in z-dimension. (c) In the third experiment the distribution gets deformed in x-dimension. (d) To avoid the obstacle in the fourth experiment the trajectories get deformed in x- and z-dimension.

The trajectory optimization was computed in an offline learning phase, whereas the conditioning on the different boxes could be performed in real time, once the ProMP was optimized for the workspace.
The algorithm connects a trajectory optimization method to the framework of ProMPs. This enabled the robot to deliver the ball to arbitrary box locations and avoid obstacles, placed in different locations. The experiments validate the method's ability to generalize demonstrated trajectories under unforeseen workspace changes.

5 Conclusion and Future Work

The ability to learn new tasks from human demonstrations and to adapt those demonstrations to different workspaces is a core requirement for future robot coworkers. While the ProMP framework allows for generalization of motions using conditioning, blending and sequencing, ProMPs offer no clearly defined methods for obstacle avoidance and motion planning. To overcome this issue, this thesis proposes to connect trajectory optimization, based on human demonstrations, with ProMPs.
The proposed algorithm optimizes trajectories with respect to two objectives: (1) it minimizes the deviation from a demonstrated distribution using Relative Entropy Policy Search and a KL metric and (2) it allows for motion planning and obstacle avoidance. The use of the KL metric provides the advantage of extracting properties, such as correlation and smoothness, out of the human demonstrations. Moreover, the proposed algorithm preserves the variance of the demonstrations during the optimization process. Evaluation of a number of toy examples indicated the method's capabilities to eliminate the need for prior assumptions on the solution.
Additionally, the thesis introduced a sliding window approach to achieve local modification of the trajectories. By optimizing trajectories for overlapping windows this approach results in a set of locally optimized trajectory distributions, that can subsequently be merged to a global solution. Experiments revealed how tuning the window size affects the solution to avoid an obstacle in its neighbourhood and preserve the demonstrations in parts unaffected by the obstacle.
Experiments on a 7-DoF KUKA lightweight arm showed the algorithm's capability to solve motion planning combined with movement primitive extraction in a real world scenario. For the experiments a Kinect camera was used to obtain a signed distance field. Results show, that the connection of the proposed trajectory optimization to the ProMP framework allows for generalization in terms of via points and obstacle avoidance.

Referring to the results of the experiments, further studies should focus on following objectives:
While the optimization in task space worked reasonably well when near the center of the robot's useful workspace, the use of inverse kinematics resulted in unexpected motions for trajectories close to the workspace limits. Optimization in joint space and null space optimization should be investigated to address these issues.
The experiments revealed an offline computation phase between two and five minutes for different workspaces. While this is applicable for static obstacles it should be a goal of future investigations to decrease this computation time. In particular a faster way to merge the single windows could speed up the algorithm. If it becomes feasible to perform trajectory optimization in real time, the method could be extended for dynamic obstacle avoidance.
Another issue is how to consider obstacle avoidance not only of the end-effector but for the full geometry of the robot arm. Future work should consider collisions of all joints and also self collisions to provide more robust solutions. For example, a bounding box approach, as the method found in [14], could be used.
Moreover, an interesting extension of the proposed method could be to combine human-robot collaboration with the optimization [35]. The experiments revealed how the proposed method was able to generate adaptive trajectories for a robot in a simple pick-and-place scenario. An interesting extension is to consider a collaborative task, in which the robot additionally should optimize its own movements to adapt to a human co-worker. Ben Amor presents in [36] a framework of Interaction Primitives which could form a basis for investigations in this direction. In particular, the proposed way of connecting the trajectory optimization with ProMPs could be extended by a connection to Probabilistic Interaction Primitives [37].
Another interesting aspect is to go beyond trajectory optimization and towards intention learning [38], [39]. Extracting intentions out of human demonstrations could provide a higher level for planning inside the optimization [40]. It should be a goal to transfer already learned primitives to tasks with similar intentions.

Bibliography

[1] Heather Knight. *How humans respond to robots: building public policy through good design*. 2014.

[2] Kate Darling. 'who's johnny?'anthropomorphic framing in human-robot interaction, integration, and policy. *Anthropomorphic Framing in Human-Robot Interaction, Integration, and Policy (March 23, 2015)*, 2015.

[3] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[4] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[5] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[6] James C Spall. Stochastic optimization. In *Handbook of computational statistics*, pages 173–201. Springer, 2012.

[7] Francisco J Solis and Roger J-B Wets. Minimization by random search techniques. *Mathematics of operations research*, 6(1):19–30, 1981.

[8] David Ruppert. Stochastic approximation. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

[9] David E Goldberg et al. *Genetic algorithms in search optimization and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[10] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*. Atlanta, 2010.

[11] Steven M LaValle and James J Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[12] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[13] Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[14] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574. IEEE, 2011.

[15] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[16] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[17] Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151. ACM, 2008.

[18] Pieter Abbeel, Dmitri Dolgov, Andrew Y Ng, and Sebastian Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1083–1090. IEEE, 2008.

[19] Gu Ye and Ron Alterovitz. Demonstration-guided motion planning. In *International symposium on robotics research (ISRR)*, volume 5, 2011.

[20] Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. 2010.

[21] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[22] Peter Englert, Alexandros Paraschos, Marc Peter Deisenroth, and Jan Peters. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21(5):388–403, 2013.

[23] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer, 1997.

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

[25] Ren C Luo, Bo-Han Shih, and Tsung-Wei Lin. Real time human motion imitation of anthropomorphic dual arm robot based on cartesian impedance control. In *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*, pages 25–30. IEEE, 2013.

[26] Dongheui Lee, Christian Ott, and Yoshihiko Nakamura. Mimetic communication model with compliant physical contact in humanâĂŤhumanoid interaction. *The International Journal of Robotics Research*, 29(13):1684–1704, 2010.

[27] Aude Billard and Stefan Schaal. Robust learning of arm trajectories through human demonstration. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 734–739. IEEE, 2001.

[28] Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *ACM transactions on graphics (TOG)*, volume 23, pages 522–531. ACM, 2004.

[29] Michael J Gielniak, C Karen Liu, and Andrea L Thomaz. Task-aware variations in robot motion. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3921–3927. IEEE, 2011.

[30] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. Technical report, 2002.

[31] Jens Kober, Katharina Mülling, Oliver Krömer, Christoph H Lampert, Bernhard Schölkopf, and Jan Peters. Movement templates for learning of hitting and batting. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 853–858. IEEE, 2010.

[32] Freek Stulp, Erhan Oztop, Peter Pastor, Michael Beetz, and Stefan Schaal. Compact models of motor primitive variations for predictable reaching and obstacle avoidance. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 589–595. IEEE, 2009.

[33] Barak Freedman, Alexander Shpunt, Meir Machline, and Yoel Arieli. Depth mapping using projected patterns. In *US Patent*, pages 8,150,142. Google Patents, 2012.

[34] M Kalakrishnan and K Anders. Moveit: Propagation distance field.

[35] Bradley Hayes and Brian Scassellati. Challenges in shared-environment human-robot collaboration. In *learning*, page 9, 2013.

[36] Heni Ben Amor, Gerhard Neumann, Sanket Kamthe, Oliver Kroemer, and Jochen Peters. Interaction primitives for human-robot cooperation tasks. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2831–2837. IEEE, 2014.

[37] Guilherme Maeda, Marco Ewerton, Rudolf Lioutikov, Heni Ben Amor, Jan Peters, and Gerhard Neumann. Learning interaction for collaborative tasks with probabilistic movement primitives. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 527–534. IEEE, 2014.

[38] Muhammad Awais and Dominik Henrich. Online intention learning for human-robot interaction by scene observation. In *Advanced Robotics and its Social Impacts (ARSO), 2012 IEEE Workshop on*, pages 13–18. IEEE, 2012.

[39] Zhikun Wang, Marc Peter Deisenroth, Heni Ben Amor, David Vogt, Bernhard Schölkopf, and Jan Peters. Probabilistic modeling of human movements for intention inference. In *Robotics: Science and Systems*. Sydney, Australia, 2012.

[40] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 299–306. IEEE, 2013.