POLITECNICO DI MILANO
DEPARTMENT OF ELECTRONICS, INFORMATION AND BIOENGINEERING (DEIB)
DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY

TOWARDS ROBOT LEARNING FROM DEMONSTRATION: A VIEW INSPIRED FROM OBSERVATIONAL LEARNING

Doctoral Dissertation of:
Amir Masoud Ghalamzan Esfahani

Supervisor:
Prof. Luca Bascetta

Tutor:
Prof. Maria Prandini

The Chair of the Doctoral Program:
Prof. Carlo Fiorini

2015 – XXVII

To my lovely family

Every moment, the voice of Love is arriving from left and right;
we are departing for the skies–who has a mind for sightseeing?
We were once in heaven, we were friends of the angles;
let us all return thither, for that is our city.
We are even higher than heavens, we are greater than angels;
why should we not transcend both? Our loading-place is Majesty.

Jalal ad-Din Muhammad Rumi–12th century

Acknowledgment

I would like to express my special appreciation and thanks to my advisor Dr. Luca Bascetta for encouraging my research, for allowing me to grow as a research scientist, and for all his kind and precious support and guidance during my PhD study.

Besides, I would like to thank Prof. Paolo Rocco and Dr. Marcello Restelli for their supportive help.

I would also like to express my sincere gratitude to Prof. Gregory Hager from Johns Hopkins University (JHU) for his hospitality and very kind and helpful support and guidance during my research period at JHU-Laboratory for Computational Sensing + Robotics.

Furthermore, I wish to thank Mr. Farshid Alambeighi, Antonio Valsecchi, Chris Paxton, Mazdak Hashempour and all my friends and colleagues both at Politecnico di Milano and Johns Hopkins University who made the period of my Ph.D. study a wonderful experience for me.

Last but not least, I would like to thank Dr. Marin Kobilarov for his helpful comments on the first draft of my thesis.

Milan, May 2, 2015

Abstract

ROBOTS are used to perform many repetitive and precise tasks. However, the programming time and cost of a robot restricts the use of robots, especially in non-production-line uses. Although robot programming by demonstration (PbD), by which a robot can learn to perform a task from demonstrations, has been introduced to tackle this issue, still a major concern is how a robot can generalize task demonstrations across different conditions.

In this regard, many studies have been recently inspired by studies of psychologists and particularly by imitation learning in observational learning, which allows a human to generalize an observation to a new goal point.

Based on the analogy between robot learning from demonstrations and human learning from observation, and according to different types of observational learning, including mimicking, imitation, and emulation, we propose a multilayered approach to robot learning from demonstration. This approach enables a robot to learn a model to perform a task from noisy demonstrations and to generalize it to a new start and goal point as well as to different environments. We demonstrate the usefulness of the approach with a practical example of sweeping.

Riassunto

I Robot vengono oggi utilizzati per l'esecuzione di molti compiti ripetitivi o che richiedono elevata precisione. Tuttavia, la loro programmazione richiede spesso tempi lunghi e, quindi, costi elevati, specialmente nella produzione non in serie.
Sebbene la tecnica di programmazione per dimostrazione, che permette ad un robot di imparare ad eseguire un compito a partire dalle dimostrazioni fornite da un operatore, sia stata introdotta per risolvere questo problema, è ancora poco utilizzata poiché il problema di generalizzare un'operazione a contesti differenti da quello in cui è stata dimostrata non è ancora stato completamente risolto.

Recentemente sono stati effettuati molti studi ispirati dalla teoria psicologica dell'apprendimento per imitazione, secondo cui una persona è in grado di generalizzare quanto appreso attraverso le osservazioni a nuovi contesti.

Sulla base dell'analogia tra le tecniche utilizzate per 'apprendimento dei robot a partire dalle dimostrazioni e l'apprendimento delle persone dall'osservazione, questa tesi propone un approccio gerarchico all'apprendimento da dimostrazioni, suddiviso in differenti livelli. Questo approccio permette ad un robot di imparare un modello del compito che deve eseguire a partire da dimostrazioni e generalizzarlo a differenti condizioni ambientali e differenti stati iniziale/finale.

L'efficacia dell'approccio proposto è dimostrata attraverso l'apprendimento di un compito che consiste nel raccogliere alcuni oggetti in una paletta mediante una piccola scopa.

Summary

ROBOTS have been used for many years in automatic production lines in the industry, because of their capabilities to perform precisely some repetitive tasks without stopping. However, programming time and cost of a robot restrict the use of robots for a new task. Robot programming by demonstration or robot learning from demonstration has been proposed to reduce the time and cost of programming a robot.

This work has three parts. The first part is devoted to the existing approaches of robot Learning from Demonstration (LfD). The existing approaches of robot programming by demonstration are presented and classified according to the mathematical approaches in Chapter 1. Furthermore, in Chapter 2, the connection between robot learning from demonstration and learning by observing in psychology are discussed. We then classify the approaches to robot LfD according to different types of observational learning such as mimicking, imitation and emulation learning.

In the second part of this work, the mathematical model of the proposed robot LfD approach are presented according to the distinguished types of robot LfD in Chapter 2. The mathematical model used in the approach are Gaussian mixture model/Gaussian mixture regression, presented in Chapters 3, Mean-Path algorithm, presented in Chapter 4, dynamic movement primitives, presented in Chapter 3, and inverse optimal control which is presented in Chapter 5.

In the last part and in Chapter 6, the proposed approach of robot LfD is applied to different problems including an example of deburring a workpiece in an industrial context, a surgical robotic task and a sweeping ex-

ample. These examples demonstrate the effectiveness of the proposed approach to obtain a model of a task from demonstrations in different contexts.

Recently, we extended the approach of robot learning from demonstration that allows a robot to replicate the demonstrated task in a dynamic scenario in that the obstacles move during task execution. This result is submitted to IROS 2015; however, we could not include the corresponding part and the results in this thesis.

Although this work tries to illustrate and exploit the connection between studies in psychology and robot LfD, there are still many rooms in this context to explore which may benefit both studies in robotics and observational learning.

Publications

This thesis is based on the following publications and other materials:

1. **A. M., Ghalamzan E.**; C. Paxton, G. D. Hager, L. Bascetta. "Robot Learning from Demonstration:from mimicking to emulation" under preparation for journal publication.
2. **A. M., Ghalamzan E.**; M. Regalia, M. Kobilarov, L. Bascetta. "Robot Learning from Demonstration: from Static to Dynamic Environments" submitted to International Conference on Intelligent Robots and Systems (IROS 2015).
3. **A. M., Ghalamzan E.**; L. Bascetta; M., Restelli; P., Rocco. "Estimating a nominal path from a set of task demonstrations" submitted to Robotics and Computer-Integrated Manufacturing.
4. **A. M, Ghalamzan E.**; C. Paxton, G. D. Hager, L. Bascetta. "An incremental Approach to Learning Generalizable Robot Tasks from Human Demonstration" accepted for publication in the Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA 2015).
5. **A. M., Ghalamzan E.**; L. Bascetta; M., Restelli; P., Rocco. "Estimating a Mean-Path from a set of 2-D curves" accepted for publication in the Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA 2015).
6. **A. M., Ghalamzan E.**; C. Paxton, G. D. Hager, L. Bascetta. "Learning How to Avoid an Obstacle from Demonstration" Robotic Science

and Systems 2014, workshop on Learning Plans with Context from Human Signals (PDF).

7. L. Bascetta, Gianni Ferretti, **A. M., Ghalamzan E.**, G. Magnani, M. Pirotta, P. Rocco, M. Restelli. "Towards an autonomous robotic deburring system" Italian national conference on Motion Control, 2013.

Competitions

- **A. M., Ghalamzan E.**; C. Paxton, G. D. Hager, L. Bascetta. "Robot Learning from Demonstration: from Mimicking to Imitation" accepted for inclusion in the proceedings of the 2015 AAAI video competition (Video).

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Robot Programming

Robots have been used for many years in automatic production lines in the industry, because of their capabilities to perform precisely some repetitive tasks without stopping. They have been used to perform many tasks that had not been planned during robot design procedure. This made them versatile machines; however, to perform every single task an expert must carefully program them. This programming procedure is very time-consuming and costly. So far, many researchers have worked on reducing the time and cost of programming a robot, that results in different robot programming approaches. In 1989, Lozano et al. [82] identified three major categories of robot programming methods, as follows:

1. *Robot-level programming or off-line programming*: special robot programming languages such as Versatile Assembly Languages or general purpose programming languages are used to explicitly control a robot. This method allows a programmer to specify every single movement of the robot. Hence, the method allows a programmer to program the robot in such a way that it can react to external events by using sensors. An expert in sensor-based motion algorithms must

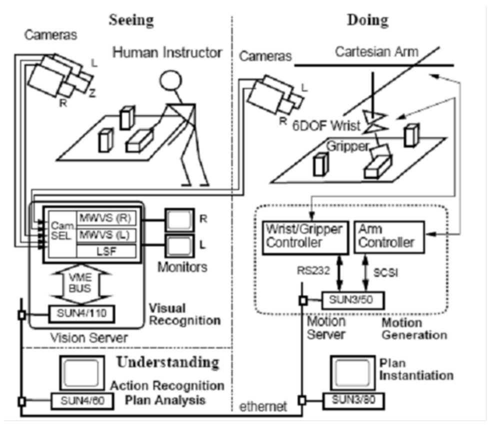


Figure 1.1: *Early approaches to Robot Programming by Demonstration decomposed a task into symbolic units. Temporal dependencies across these units were used to build a hierarchical task plan that reproduce the task [16].*

program the robot, specifying the functionality of the robot based on the sensory information. This enables a robot to cope with uncertainty in the workspace. For example, a robot can be programmed to recognize an object, estimate its position and pick the object from different positions.

2. *Task-level programming or implicit programming*: a programmer only determines the goals and sub-goals of a task. Then, a motion planning system, such as MoveIt [31] or STRIPS[1] [43], is used to generate the necessary movements to carry out the task.

3. *Programming by Guiding (PbG)*: it is also known as teaching by showing, teach-in, play-back, walk-through and lead-through. In programming by guiding, an operator moves the robot's end effector to perform a corresponding task. During these movements, all relevant positions are explicitly recorded[2]. The sequence of recorded poses is then used to carry out the demonstrated task. The full motion is then obtained by interpolation [83] between the set of key points or a planning system can be used to reach the key points.

An operator may only know implicitly how to perform a task, but he may not have either skills or the time to program the robot to perform the task. PbG allows the operator to teach the robot how to perform a task by

[1]STRIPS is abbreviation of Stanford Research Institute Problem Solver, is an automated planner that finds an optimal plan at the symbolic level to perform a task.

[2]An operator may use a teach pendant to move the robot's end effector or he may tele-operate the robot.

showing a robot an example execution of the task, and without programming it at task- or robot-level. Programming by guiding approach has been effectively used to reduce the programming time of a robot and the corresponding cost in a production line. Nonetheless, it is only effective in the case where a robot has to repeatedly execute the same movements. For example, it has been used successfully to program an industrial robot for spot welding or painting [6].

Despite the advantage of using programming by guiding, some limitations arise where the robots have to repeat the task in a slightly different working condition. For example, although PbG enables a robot to accomplish a task if the conditions during all task reproductions are identical, namely in the painting and the welding examples, it cannot be used to enable a robot to pick an object from slightly different positions.

In order to cope with these limitations of PbG, and to generalize a demonstration to a slightly different condition, Munch et al. [91] proposed an approach called *Robot Programming by Demonstration (RPD)*. This approach decomposes a demonstration into some symbolic units. Then, a planning approach is used to determine temporal dependencies of units to reproduce the task in different conditions (Figure 1.1). RPD incorporates the sensory information into a programming procedure to generalize the task demonstrations across different situations. For instance, consider the task of picking an object from a conveyor belt in a production line and putting it in a box, where the position of the object relative to the robot changes at each repetition. Our goal is to teach a robot how to pick objects from different given positions.

RPD has been initially inspired from *programming by demonstration* in software development [11, 50]. Over the past couple of decades, many RPD approaches have been developed at different levels of task abstraction [44]. Many machine learning approaches have been adopted to solve the generalization problem of RPD, such as Gaussian mixture model/ Gaussian mixture regression [22], artificial neural networks [79], fuzzy logic [37], radial-basis function networks [61], hidden Markov models [72] and inverse reinforcement learning [13]. Later in 1990s, due to analogies between human learning by observing and robot programming by demonstration[3], *observational learning* in psychology study became a main source of inspiration to develop new corresponding approaches of RPD [110]. Hence, robot programming by demonstration and robot learning from demonstration have been interchangeably used.

[3]Hence, here we use robot learning from demonstration as an identical term to robot programming by demonstration.

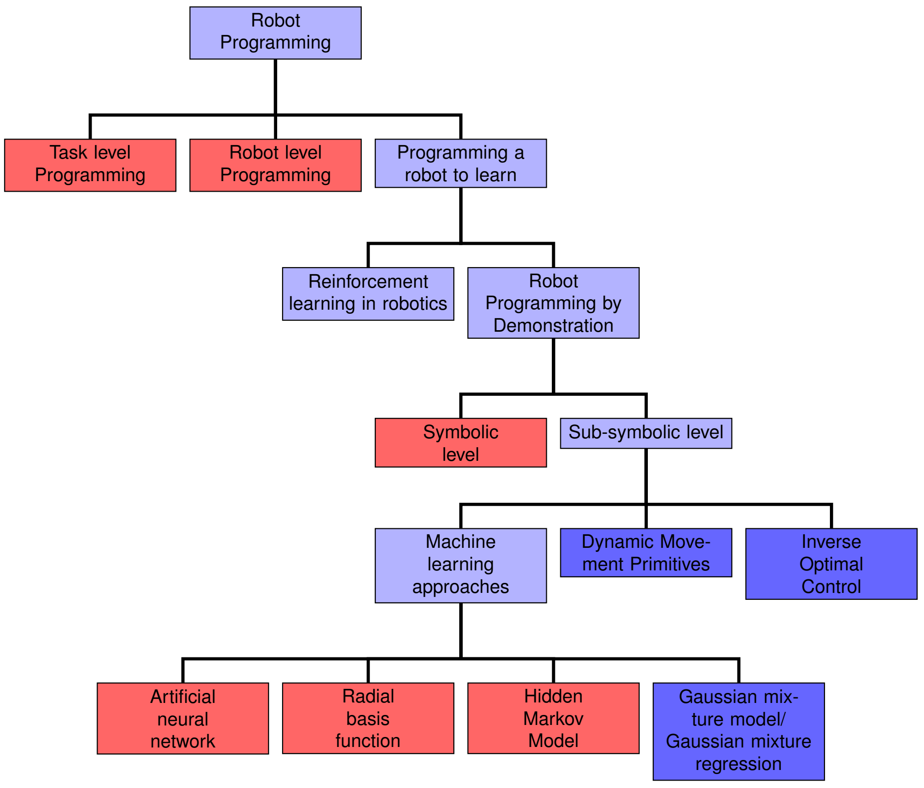

Robot Programming
Task level Programming
Robot level Programming
Programming a robot to learn
Reinforcement learning in robotics
Robot Programming by Demonstration
Symbolic level
Sub-symbolic level
Machine learning approaches
Dynamic Movement Primitives
Inverse Optimal Control
Artificial neural network
Radial basis function
Hidden Markov Model
Gaussian mixture model/ Gaussian mixture regression

Figure 1.2: *Schema of robot programming categories distinguished in this thesis.*

So far, four different types of observational learning have been distinguished in psychology studies [123, 135], including *stimulus enhancement*, *mimicking*, *imitation*, and *emulation*. In observational learning apart from stimulus enhancement, a human learns how to perform a task from observing others performing either the same task or another task. In fact, every type of observational learning represents a level of generalization of the observed task to a new scenario. Accordingly, in RPD context, we recognize three levels of generalization of task demonstrations such as RPD through *mimicking*, *imitation* and *emulation*. Then, we propose a workflow of RPD allowing a robot to learn a model of a task from a set of noisy demonstrations and to generalize it across different situations. In the context of robot learning from demonstration, we are interested to find simpler modular models that can be easily transferred across different tasks. The workflow of RPD consisting of three proposed types of RPD decomposes a model of a complex task to be learned into simpler components that can be used across different tasks.

In the rest of this chapter, we present an overview of the related studies that extended robot programming by guiding to robot programming by demonstration. We distinguish two levels of task model abstraction of RPD, i.e. symbolic level and sub-symbolic level (Figure 1.2). State-of-the-art of three main lines of studies of robot programming by demonstration such as statistical approaches, Dynamic Movement Primitives and inverse reinforcement learning (Figure 1.2), as well as their limitations and advantages are discussed in Section 1.4.1. The application of the existing methods of RPD in an industrial problem is presented in Section 1.4.1, and their limitations are discussed. Finally, in Section 1.5 we present the contribution of this thesis. In Figure 1.2, the works related to the content of this thesis are shown with dark blue blocks. The rest are branches of studies that are out of the scope of this thesis, but are here considered in order to draw a complete picture of RPD.

1.2 Programming by Demonstration

In software developing, programming by example or Programming by Demonstration is an approach for faster software development in which computer learns a new behavior of a user by recording user's input through a user interface. A computer can later exploit the learned behavior to help the user by reducing the amount of required inputs that the user must give to the computer/machine to carry out a user's desired procedure (Fig. 1.3).

In 1950, Turing [128] proposed the question " Can machines think?". This initiated many studies in different machine learning contexts. In the late 70s, Bauer [11] proposed an algorithm to synthesize a procedure from a set of example computations. This work was based upon the work of Biermann and Krishnaswamy [12], in which formation of procedures from sequences of instructions are discussed. In the mid-80s, Halbert [50] presented a practical system for programming by example. He discussed further how that system could be incorporated into a richly functional, commercial software system. In the 80s and early 90s, programming by example and RPD had been mutually inspired [50]. Eventually in the late 1990s, many researchers started to cope with this problem from the viewpoint of learning [110].

1.3 Robot programming by demonstration

"*Industrial robots have long been programmed by example [122]. The programmer of the robot can either physically move the robot's*

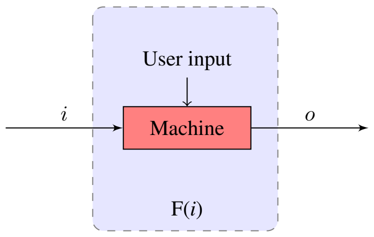


Figure 1.3: *Schema of computing machine, initial state* i*, final state* o *and user input. Programming by demonstration aims at learning function* F.

"arms"(leadthrough) or else command the robot from a control panel (walkthrough), and have the robot record the sequence of movements and actions. The machine moves the workpiece, its drill bits, cutting tools, spray guns, arc welders, and so forth as it follows the program."

(Halbert [50] 1984)

In the early 1980s, programming by guiding began attracting attention of the people working in the domain of industrial robotics [122]. Eventually, it has become a useful and practical approach of programming an industrial robot to perform a task, which is less time-consuming and costly than the conventional programming methods, namely robot- and task-level programming. Still, a main advantage of conventional robot-level programming was that it has capabilities to cope with some uncertainties in the workspace. This allows a robot to adaptively perform a task in different situations. In order to extend robot programming by guiding such that the robot can perform a demonstrated task in different environments, *generalization* became a crucial point [5]. However, the generalization of a demonstrated task in robotics is a much more challenging problem than generalization in software development because task reproduction involves some hard physical constraints to be satisfied. Furthermore, the operator and the robot may not necessarily share the same embodiment and affordances.

After the appearance of non-industrial robots, such as humanoid and mobile robots, the generalization issue became even a more crucial point, since the robot's workspace was no longer fixed and under control. Thus, instead of simply copying a single demonstration, a robot had to generalize the demonstrated movements to a new situation and across a set of demonstrations. In the domain of humanoid robotics, RPD resembles observational learning[4] in human [135]. Accordingly, many studies were in-

[4]Observational learning is a basic way of learning in human. In this way, a person learns to perform a task by watching another person doing that task.

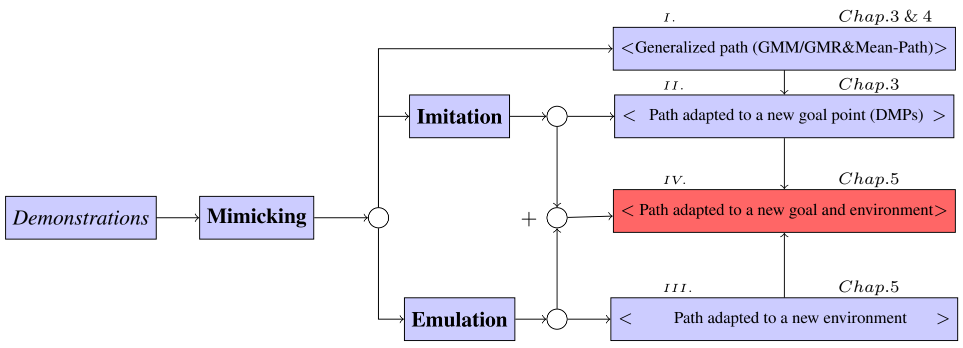

I. Chap.3 & 4
<Generalized path (GMM/GMR&Mean-Path)>
II. Chap.3
< Path adapted to a new goal point (DMPs) >
IV. Chap.5
< Path adapted to a new goal and environment>
III. Chap.5
< Path adapted to a new environment >
Imitation
Emulation
Mimicking
Demonstrations
+

Figure 1.4: *Schema of Robot Learning from Demonstration workflow. In the proposed workflow, first a generalized path is computed from a set of given task demonstrations (level I). The obtained path is used as a baseline to learn Dynamic Movement Primitives (level II) as well as learning a cost function encoding a response of demonstrator to environment (level III). Finally a cost function combines the path with a new goal point and response of the demonstrator to environment (level IV).*

spired from observational learning and specifically from imitation learning[5] [94, 110, 23].

In the following sections, we briefly review the works related to the problem of generalization of task demonstrations to a new environment. In Chapter 2, we present the corresponding types of observational learning, and then we discuss how each level of RPD gives rise to a different aspect of generalizing task demonstrations to different situations.

1.4 Problem of generalization

The first attempt at generalizing a demonstrated skill was mainly based on the user's feedback to give the robot explicit knowledge about the user's intentions [53]. Friedrich et al. [44] proposed a dialog based method in which a demonstration of the task was stored in a trace and transformed into the symbolic level using a given STRIPS-like planning language. Then, they used a planning language to find the optimal sequence of basic operations. Moreover, the user's intention was used to analyze a demonstration and to detect and eliminate the superfluous subparts. Therefore, generalization occurred based on the user's intention and optimization of the planning parameters. Friedrich et al. [44] proposed two levels of task model abstractions to resolve the generalization issue, as follows (Figure 1.2):

[5]Imitation learning is a type of observational learning in which a person copies the form of the demonstrated movements to achieve a goal. For example, a toddler may watch a person pouring water into a cup and he may copy the form of the demonstrated movements to repeat the same task where the positions of the cup changed.

1. symbolic level
2. sub-symbolic level

RPD at the symbolic level decomposes demonstrations into a sequence of subtasks including some perception-action units as addressed by Ekvall and Kragic [42], Friedrich et al. [44], Nicolescu and Mataric [97] and Pardowitz et al. [102]. An induction process is then used to determine the sequence of subtasks to be performed in a new condition. At the sub-symbolic level, instead, RPD determines a nonlinear mappings between sensory and motor information. Hence, a task at the sub-symbolic level is described by continuous signals representing different configurations of the robot during the execution time [7, 112, 129, 56, 21].

After Turing [128] proposed the question " Can machines think?', many machine learning techniques were developed that could resolve the generalization problem in the RPD. The developed approaches attracted the attention of robotic researchers [124] when generalization became a major issue in the RPD. Muench et al. [91] used machine learning techniques to identify elementary operators from a demonstration. They extracted the dependencies of each elementary operation at a symbolic level, based on the user feedback. A sequence of discrete basic motor skills was then determined by a planner to use the obtained skill of performing a task in a new condition. Many other works explored further the symbolic reasoning in the context of RPD [42, 109, 120].

The symbolic level of RPD determines the sequence of symbolic representation of the subtasks by induction. Hence, a robot cannot only use them to execute the corresponding task. In order to perform the task, a robot must interpret the high-level symbolic information and then generate the necessary low-level information, such as motor current over the execution time. A planner, such as MoveIt and STRIPS, can be used for example to generate the motor current based on the symbolic representations of the corresponding task. On the other hand, the sub-symbolic level of RPD generates a path or trajectory to be followed during task execution. The sub-symbolic level methods of RPD are thus lower level of abstraction than symbolic level methods.

A few years later, different levels of abstraction at the sub-symbolic level have been also considered. For instance, data corresponding to a demonstration can be collected as the end effector poses or as the motor currents. In this regard, Alissandrakis et al. [5] defined three different levels of the sub-symbolic method. In the following, these levels are presented in successively increasing levels of resolution, called 'granularity'.

- end-point level: a robot tries to reach the overall destination. Hence, the behavior of the robot between the starting point and endpoint may be qualitatively different from the one shown by the demonstrator.
- trajectory level: a robot has to visit a sequence of some important points to be reached specified by the demonstrator.
- path level: a robot tries to exactly visit all the sequence of points that demonstrator followed during task demonstration.

We assume that a complex demonstrated task can be decomposed into many atomic movements, called sub-tasks. Hence, generalization of demonstrated task may need to simultaneously occur at the symbolic and sub-symbolic levels. In fact, a robot learns from task demonstrations how to determine a sequence of sub-task at the symbolic level as well as how to generate appropriate trajectories to accomplish each atomic movement in an unseen environment. Dillmann et al. [38] combined RPD methods at the symbolic and trajectory levels to benefit from generalization at both levels. Nonetheless, in this thesis we focus only on generalization of task demonstrations at the sub-symbolic level. In the following section we briefly review the studies on generalization problem at the sub-symbolic level.

Besides the studies on generalization issue, many studies focused on developing new interfaces to log the necessary data of a task demonstration, such as kinesthetic teaching method [60], data gloves [127], laser range finder [58] and vision sensors [62].

1.4.1 Related works

To solve the problem of generalization at the sub-symbolic level, many approaches of machine learning has been adopted, such as artificial neural networks [79], fuzzy logic [37], radial-basis function networks [61] and HMM [72]. In the following, we review three main lines of research related to each level of the proposed workflow of RPD.

1. Machine learning approaches
2. Dynamic Movement Primitive
3. Inverse reinforcement learning

RPD using machine learning approaches

The use of statistical techniques has been investigated in many studies to encode, recognize and reproduce a set of movements. HMM has been used to encode a temporal, as well as a spatial variation of a complex movement. For instance, Hovland et al. [55] modeled a skill at task level by a set of discrete event controller. The structure of the discrete event is then modeled by HMM whose parameters are obtained by training on the sensory data of human demonstration. Tso and Liu [126] used an HMM model to represent human demonstrations implying an appropriate model for human performance modeling that is then used to retrieve the Cartesian trajectory.

In Computer Graphics, Brand and Aaron [19] suggested using HMMs to identify common elements in a motion and synthesize new motions by combining and blending these elements. In many works, an averaging approach has been used to retrieve human motion sequences from HMM [59]. Lee and Nakamura [73] used HMM to encode a set of trajectories captured by a monocular camera. The multiple HMM is then employed to retrieve a generalized movement. Calinon et al. [25] encoded trajectories into HMM models by decomposing them into a set of relevant key points and employed spline fitting to retrieve and generalize the continuous trajectories. The main problem of using HMM to encode the demonstrated trajectories is that this model reproduce discontinuous trajectory. Billard et al. [15] used interpolation between a set of computed key points to resolve this problem.

Other studies, instead of encoding the demonstration into HMM, encoded a set of trajectories into Gaussian components of an HMM. Then, they used Gaussian Mixture Regression to obtain a smooth generalized trajectory [27]. The proposed method inherently solved the problem of smoothness of the generated trajectory. For example, Calinon [22] used HMM (Figure 1.5) as a first approach. He then further explored the joint use of GMM/GMR, see Figure 1.6.

Calinon et al. [27] showed that the first principal advantage of GMM/GMR over HMM is that task constraints can be represented continuously along the trajectory and GMM/GMR model reproduces smoother trajectory than HMM. To combine several constraints across different task executions, they used GMM/GMR. They also showed how GMM/GMR can be incrementally trained. Cho and Jo [32] used the GMM/GMR method to encode the demonstrated behavior in the components of Gaussian parameters. They proposed a method that automatically determines the number of Gaussian components during learning, when a new teaching trial is available.

In order to satisfy the constraints in both joint and task spaces, Calinon and Billard [24] used the inverse kinematics information to modify the covariance matrix of the Gaussian component during reproduction. Muhlig et al. [92] used GMM to encode the dataset. Then, they defined a cost function

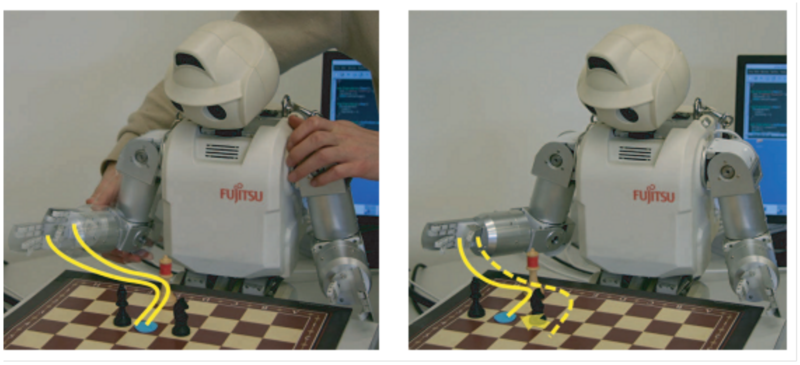


Figure 1.5: *Left: A robot learns how to make a chess move by generalizing across different demonstrations of the task performed in slightly different starting positions of the hand. Right: The robot reproduces the skill for slightly different initial position of the chess piece by computing appropriate path by HMM [22].*

rating similarity between the produced movement and the demonstrated one to reproduce the movement in a new condition.

RPD has been originated in the domain of industrial robots, as mentioned at the beginning of this chapter. Nonetheless, the developed statistical approaches focused on non-industrial problems. The mentioned statistical approach aimed at using variability across different demonstrations in order to capture the most relevant trajectory in different situations. Accordingly, they provided some approaches for online adaptation and learning that are useful for humanoid robots [131].

When the precision is not a major issue in a robotic task, the developed statistical approaches of RPD may be successfully used to reproduce the demonstrated motion in a different situation. For example, in a production line in the presence of perturbation and/or spatial and temporal constraints, the task of picking an object from different positions on a conveyer belt and putting it in a box needs the precise start and goal point, but may not need a precise trajectory to be followed. Hence, the precision of the trajectory generation phase may not be a major issue.

In Chapter 4, we present how RPD can be used in a problem with a high precision requirement. For example, we describe how RPD can be used in a deburring problem to automatically generate a nominal profile of the workpiece from a set of demonstrated profiles generated by an expert. The main goal in deburring is to remove the burrs from final products, see Figure 1.7. In this problem, the robot must automatically generate a nominal profile of a new product and learn the feed rate of deburring tool from a set of human demonstrations to perform the deburring operation. The product may have

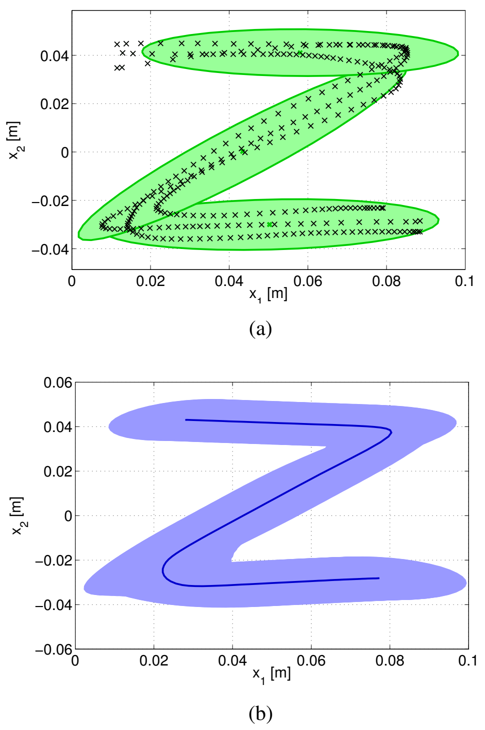


Figure 1.6: *(a) A set of three sample 2-D paths and the computed Gaussian components in the plane; (b) The computed GMR model based on the computed GMM.*

different nonlinear profiles. Since human performance may not be optimal, deburring residuals may exist on the final product at random positions after an expert carries out the deburring operation.

Deburring problem In-contact tasks, such as deburring, are the most common robotic tasks in manufacturing [139]. RPD is useful especially in the case of frequent changes in production in medium and small size enterprises. In order to program the robot to perform the deburring task, different sources of information, e.g. human input, sensor data and a model of the work-piece, are used [36]. The goal of RPD, however, is to reduce the programming time and cost by decreasing the amount of information provided by an expert that a robot requires to perform a task [110].

Shimokura and Liu [118] stored the tool feed rate in accordance with varying burr characteristic in an associative memory, given a set of demonstrations of an expert performing the task of deburring. The learned associative memory was then represented as a neural network to produce the

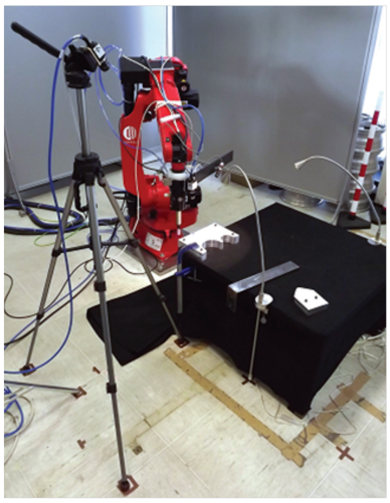


Figure 1.7: *A setup prepared to collect data of work piece profiles and perform deburring operation [10].*

feed rate for a new burr characteristic. In this work, the burr size was measured by laser sensor based on an available model of the work-piece. Ziliani et al. [142] used a hybrid force/velocity control method along with a new design of the deburring tool with two ball bearings to avoid penetration of the tool in the work-piece. Therefore, a profile model was not required since a mechanical constraint, namely two ball bearings, align the tool with the burr free surface of the work-piece. Aertbelien and Van Brussel [3] modeled the expert actions as an impedance controller with adaptive parameters in accordance with burr characteristics. A neural network was then employed to learn the nonlinear relation between the parameters and burr characteristics.

Although the problem of learning the force and feed rate of the tool for a deburring task has been studied in prior works, the problem of building a profile model from a set of suboptimal and noisy human demonstrations has not been taken into account in those works. Building a profile model from a dataset of observations is especially important for medium and small size companies where a precise CAD model of each final product may not be available.

As mentioned in the last prior section, in the last two decades a growing body of works on RPD has been devoted to non-industrial robotic problems. For example, multi demonstrations are employed to find a subspace representation of a demonstrated task [51, 68, 26]. Calinon et al. [27] linearly projected a dataset of human demonstrations onto a subspace of lower dimensionality using Principal Component Analysis (PCA). GMM/GMR method is then used to compute a nonlinear generalized trajectory from a subspace representation of the demonstrations.

We assume that the profile model of a work-piece may have different nonlinear shapes. Thus, in analogy with standard PCA, we can use Principal curves [39] and nonlinear PCA [115]. These methods have been proposed to capture and remove nonlinear correlation between data points where data points lie on a nonlinear manifold. These methods are subsets of a framework developed to compute a nonlinear manifold of the data points, such as principal curves [52, 63, 100, 117], kernel PCA [119, 78] and nonlinear PCA [85, 80, 140, 114]. In a dataset of deburring task demonstrations, we assume that every single demonstration is strongly correlated with the task profile. Therefore, a nonlinear principal curve of a demonstrated dataset represents a noise free profile model of the corresponding work-piece.

1.4.2 Dynamical system

In another part of this thesis, we consider the problem of learning a model of a demonstrated task that allows a robot to reproduce a path/trajectory to a new goal point. Although the use of HMM and GMM/GMR addressed some aspects of RPD in robotics, they could not provide the flexibility, that is usually needed for generalizing the demonstrated task to a new goal point.

In another line of research, a dynamical system approach has been adopted to deal with perturbations as well as to reproduce a trajectory with a new goal point. For example, Ito et al. [60] proposed modeling the interaction between a human user and a humanoid robot by using a Recurrent Neural Network. This model learns the dynamics of the motion and allows to switch between different motions through a dynamic interaction. Ijspeert et al. [56, 57] proposed a motion planner called Dynamic Movement Primitives (DMP), which uses an autonomous set of nonlinear differential equations that form a control policy to generate a trajectory with a new goal point, see Figure 3.4. They demonstrated that their approach is robust to external perturbations and it can be modified on-line by additional perceptual variables. The stability of evolution of a produced trajectory from an initial point to an attractor goal point is further studied by Hoffmann et al.[54], Ijspeert et al. [57] Lim et al. [76] and Schaal et al. [112]. This approach is appropriate for many manipulative tasks, in which the start position and goal position are of utmost importance to reproduce the task. Calinon et

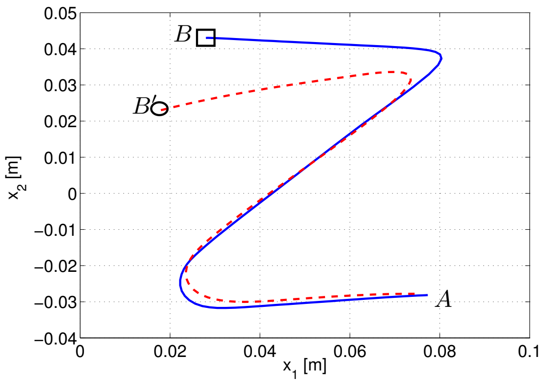


Figure 1.8: *The computed path by GMM/GMR (blue line) with start point A and goal point B. The produced path by the learned Dynamic Movement Primitives with start point A and the new goal point B'.*

al. [30] explored further the use of GMM to encode a set of virtual spring-damper systems connected to a set of candidate coordinate frames based on a set of demonstrations. The desired trajectory is then generated by use of GMR. This allowed them to combine the statistical method and dynamical system. Therefore, they benefited from important features of the task characterized by a statistical approach, as well as the capability of dynamical systems to cope with perturbations in real-time and generating a trajectory with a new goal point. In the mentioned methods, a dynamic model is trained using a dataset of task demonstrations, and it is then used to reproduce a trajectory with a new goal point.

During reproduction of a trajectory in the presence of an obstacle, avoiding an obstacle is a major issue and serious concern in robotics. A range of methods has been developed to cope with obstacles in the environment. Some studies combined the obstacle avoidance policy with policy obtained by robot learning from demonstration methods. Calinon et al. [28] used a modified version of dynamic movement primitives by considering a set of virtual spring along the demonstrated trajectories. They used the variability of different demonstrations along the movement to estimate stiffness matrices of the virtual springs. A risk indicator modulating repulsive force was then defined to enable a robot to safely avoid collisions with a human. In another work by Guenter et al. [48], a model of a task was developed using a dynamical system modulated by a GMM. This was combined with reinforcement learning to enable a robot to learn a new way of accomplishing a task in a constrained environment. Kormushev et al. [67] used a model

based on a dynamic system learned by a demonstrated trajectory. They then used reinforcement learning to compute the optimal parameter values of the model for a new environment.
Park et al. [103] employed a dynamic potential field that depends on the relative distance and velocity between end-effector and obstacle combined with a dynamic model to solve the problem of obstacle avoidance. In this work, the gradient of the potential field was added to the acceleration term of the differential equation of dynamic movement primitives. Hoffmann et al. [54] proposed a method based on the use of a relative angle between robot end-effector velocity and a vector connecting the position of the end-effector to the position of the obstacle. This relative angle is then modified based on the relative position and velocity of the end-effector and obstacle. A term of perturbation, which is a function of the relative angle, is then added to the Dynamic Movement Primitives formulation to guarantee that the generated trajectory does not collide with the obstacle. Lin and Lai [77] encoded the learned skill of reaching an object in GMM components. They used reinforcement learning to modify the component parameters and adjust the end-effector trajectory near the obstacle.

The main challenge addressed in these works was to solve the problem of the obstacle avoidance in combination with the learned model, such as dynamic movement primitives, for robot learning from demonstration. In these works, the parameters of obstacle avoidance model were fixed explicitly in the formulation. Although the goal of RPD is to reduce the programming time and cost by enabling a robot to learn a new task from task demonstrations, the methods mentioned in this section do not provide the flexibility needed to teach the robot the desired responses to different obstacles.

Based on the workflow proposed in this thesis, we will present a multilayer approach, which builds upon the Dynamic Movement Primitives a utility function encoding the response of a demonstrator to a given obstacle. To recover the corresponding utility function of a task demonstration we use inverse optimal control, also know as inverse reinforcement learning approach. This method enables a robot to learn obstacle avoidance from a set of task demonstrations.

1.4.3 Inverse reinforcement learning

Inverse reinforcement learning is an approach to recover a utility function underlying a set of task demonstrations, attracting many researchers of different domains including robotics. Ng and Russell [96] addressed the prob-

Table 1.1: *Methods and studies of robot programming by demonstration based on the classification discussed in this chapter including sub-symbolic level approaches, such as inverse reinforcement learning (IRL) and Dynamic Movement Primitives (DMPs).*

Robot Programming by Demonstration			
Symbolic level	**Sub-symbolic level**		
	IRL	**Statistical approach**	**Dynamical System**
		Hidden Markov Model:	
Muench et al. [91], 1994	NG et al. [96], 2000	Hovland et al. [55], 1996	Calinon et al. [28], 2010
Kuniyoshi et al. [70], 1994	Abbeel et al. [1], 2004	Tso and Liu [126], 1996	Calinon et al. [30], 2012
Wallner et al. [133], 1994	Abbeel et al. [2], 2007	Brand et al. [19], 2000	**DMPs:**
Dillmann et al. [37], 1995	Ziebart et al. [141], 2008	Billard et al. [15], 2006	Ijspeert et al. [56], 2002
Kaiser et al. [61], 1996	Ratliff et al. [106], 2009	Inamura et al. [59], 2006	Schaal et al. [112], 2005
Friedrich et al. [44], 1996	Mombaur et al. [90], 2010	Calinon et al. [27], 2007	Lim et al. [76], 2005
Nicolescu et al. [97], 2003	Dvijotham et al. [40], 2010	Calinon et al. [29], 2010	Ito et al. [60], 2006
		GMM/GMR:	Hoffmann et al. [54], 2009
Ekvall et al. [42], 2006	Boularias et al. [18], 2011	Calinon et al.[27], 2007	Bitzer et al.[17], 2009
Pardowitz et al. [102], 2007	Levine et al. [75], 2011	Calinon et al. [24], 2008	Ude et al. [129], 2010
Dillmann et al. [38], 2010	Levine et al. [74], 2012	Muhlig et al. [92], 2009	Tamosiunaite et al. [121], 2011
Ek et al. [41], 2010	Billard et al. [13], 2013	Calinon [22], 2009	Nemec et al. [95], 2012
	Billard et al. [13], 2013	Calinon et al. [29], 2010	Ijspeert et al. [57], 2013
		Cho and Jo [32], 2013	**Obstacle avoidance:**
		Other methods:	Guenter et al. [48], 2007
		Liu and Asada [79], 1993	Park et al. [103], 2008
		Shimokura and Liu [118], 1994	Pastor et al. [104], 2009
		Atkeson et al. [7], 1997	Calinon et al. [28], 2010
		Miyamoto et al. [89], 1988	Kormushev et al. [67], 2010
		Aertbelien et al. [3], 1999	Kulvicius et al. [69], 2012
		Schaal [110], 1999	Lin and Lai [77], 2012

lem of extracting a reward function from a given observed optimal behavior in the context of Markov decision processes. In large state space, they approximated the reward function by a linear combination of some predefined basis functions. Abbeel and Ng [1] proposed an approach, called apprenticeship learning, to learn a reward function from expert demonstrations. They assumed that the reward function is a linear function of predefined features. In this approach, an empirical estimate of the features is computed for a set of demonstrations. They proposed an algorithm that finds parameters of the reward function by minimizing a distance between the estimated empirical feature of the demonstrations and an estimated empirical feature of a reproduced trajectory. Nonetheless, if the true reward function is not a linear function of the features, the problem became an ill-posed problem. Ziebart et al. [141] employed the principle of maximum entropy to address this issue. They used it to determine a distribution over decisions to recover the reward function of the demonstrated behavior for deterministic Markov decision processes.

Dvijotham and Todorov [40] proposed a method called inverse optimal control with linearly solvable Markov decision processes to recover a policy and a cost function of a set of demonstrations. A major advantage of this method is that it does not solve the optimal control problem iteratively to recover the cost function, i.e. it does not compute the optimal solution to the estimated cost function. They showed that the proposed algorithm outperformed the previous inverse reinforcement learning methods. Levine et al. [75] presented an approach of inverse reinforcement learning by employing Gaussian processes to recover a nonlinear reward function. They demonstrated that the method is able to select the relevant features to the task among a set of candidate features. Levine and Koltun [74] explored further a probabilistic inverse optimal control algorithm to recover a continuous reward function. They used the derivative of the proposed reward features to learn the reward function. Therefore, in their approach the reward features need to be differentiable.

Although inverse reinforcement learning has demonstrated promising results, there are few studies that use inverse reinforcement learning in the context of robot learning from demonstration due to its computational complexity. For example, Boularias et al. [18] proposed a model free inverse reinforcement learning algorithm employing the relative entropy between state-action distribution under a learned and a true reward function. They validated the result of their method on a robotic simulation example. Boularias et al. [13] used inverse reinforcement learning to learn a reward function, a linear combination of some predetermined known features, from a set of demonstrations in which some experts performed the same task. The proposed approach was then validated by a simulated robot example.

Inverse reinforcement learning allows for computing a reward function underlying a set of task demonstrations, however learning a full reward function of a task is computationally expensive. This complexity increases by increasing the dimensions of the state and action space. Our proposed method, that simplifies the learning process by decomposing it into different components, reduces the complexity of the problem of computing a utility function underlying the demonstrations. This results in a utility function with lower dimension of state and action space. We benefit from the developed methods of RPD, GMM/GMR, DMP and inverse optimal control, to compute different components.

In Table 1.1, some of the related works to the corresponding workflow of RPD are listed. These works correspond to the blocks shown in Figure 1.2 with dark blue.

1.5 Contribution and outline of the thesis

In this thesis, we assume that a robot must learn how to perform a task from a set of sub-optimal and noisy task demonstrations. We decompose a complex processes of robot learning from demonstration into different components. These components provide different levels of generalization of the demonstrations, resulting in a workflow of RPD. This workflow allows a robot to obtain a task model from a set of noisy demonstrations and allows the robot to generalize the task model to a new start and goal point as well as to a new environment in the presence of obstacles.

In Chapter 2, the observational learning and its different types are briefly presented. We discuss how the studies on psychology inspired us to decompose a learning from demonstration processes into three layers: mimicking, imitation and emulation.

In Chapter 3, we present the problem of mimicking, computing a generalized path across a set of sub-optimal task demonstrations. We emphasize the precision of the computed generalized path by defining a precision metric in Chapter 4. Accordingly, we propose an algorithm inspired from expectation-maximization to compute the generalized path from a set of demonstrated paths. The precisions of the generalized path computed by this algorithm, by GMM/GMR and by NLPCA are compared for different datasets. The comparison illustrates that the proposed algorithm outperforms both GMM/GMR and nonlinear PCA in terms of precision. In the deburring example, the obtained profile model can be later used as a baseline for other learning methods, e.g. neural network [3] or reinforcement learning, to learn a control policy to compute the required temporal information, such as feed rate of the tool, and force. These values, namely the force and the feed rate, determine the removed amount of burrs and must be adapted during every single execution, as the tool sharpness may vary from one demonstration to another one.

The problem of imitation is also studied in Chapter 3. Dynamic movement primitive is used to generalize a desired demonstration to a new start and goal point.

In Chapter 5, the problem of emulation is presented. In this chapter, we show how a robot can learn specific reaction to a stimulus during task reproduction. In particular, we focus on the problem of obstacle avoidance through RPD. To do so, we build a utility function whose parameters are learned from task demonstrations. Our approach incorporates mimicking, imitation and emulation into a utility function, allowing us to generate a necessary path to perform a task with a new goal point in an environment with an unseen configuration of obstacles.

In Chapter 6, we use our approach to obtain a model of a robotic example designed for surgeon training. Furthermore, we applied the approach to a problem in which an industrial robot learns from noisy task demonstrations how to sweep a rubbish into a dustpan while reacting specifically to different non-rubbish object.

CHAPTER 2

Robot learning from demonstration (LfD)

2.1 Outline of the chapter

This chapter points to the connection between robot programming by demonstration and observational learning in psychology[1] [9]. Hereinafter, due to this connection we use Robot Learning from Demonstration (RLfD) for the framework in which a robot learns how to perform a task from a set of task demonstrations.

RLfD is a subset of robot learning (see Figure 2.1). Corresponding to human learning, robot learning allows a robot to obtain the ability to perform a task by different means, such as reinforcement learning in robotics, knowledge transfer and experimental learning.

In this chapter, we aim at using the existing taxonomy of observational learning proposed in psychology by Thompson and Russell [123] and Whiten et al. [135]. We adapt this taxonomy to specify the corresponding processes of robot learning from demonstration. This helps us to recognize and decompose different processes of robot learning from demonstration that eventually results in relatively simpler models.

[1] Observational learning is a learning process that occurs through observing the behavior of others.

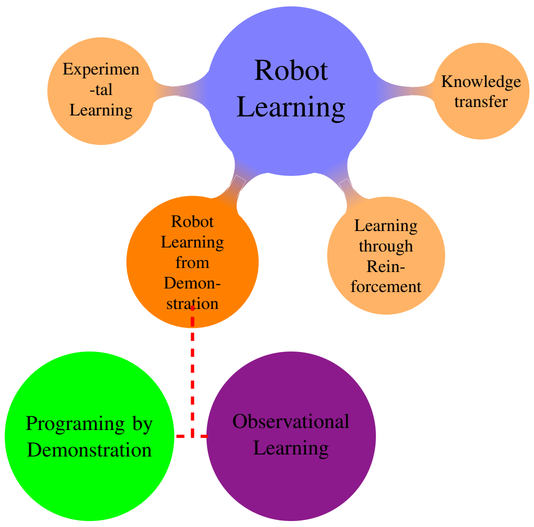

Robot Learning
Experimen -tal Learning
Knowledge transfer
Robot Learning from Demon- stration
Learning through Rein- forcement
Programing by Demonstration
Observational Learning

Figure 2.1: *Robot learning includes several types, including robot learning from demonstration and reinforcement learning. Robot learning from demonstration has strong connection with observational learning and programming by demonstration.*

2.2 From RPD to RLfD

The first approach of robot learning from demonstration, as mentioned in Chapter 1, appeared in industrial robotics and was called programming by guiding, teaching by showing, teach-in, play-back, walkthrough and lead-through. The redundant terminologies of robot programming by demonstration methods exist due to the different types of data collection of demonstrations, due to the different level of generalizing the demonstration to a new situation and due to the different abstraction levels of the model. For example, Friedrich et al. [44] distinguished the generalization of the demonstrated task at the symbolic and sub-symbolic levels. A robot collects the trajectory data set when a robot programmer physically moves the robot's arm in the *leadthrouth* method or the programmer moves the robot by tele-operating in *walkthrough* method.

As people started to use robot programming by demonstration in domains different from industrial robotics, e.g. humanoid or mobile robotics, robot learning from demonstration became an identical term to robot programming by demonstration [8]. This seemed natural due to the analogy between robot programming by demonstration and observational learning in psychology [135]. Eventually, robot learning from demonstration has become the most used term in robotics for the framework in which a robot acquires the ability to perform a task from a set of task demonstrations. Nonetheless, a variety of terms has been used interchangeably for this framework in the robotics literature, such as robot learning/programming by/from demonstration/watching, programming by guiding, imitation learning, teaching a new skill to a robot and apprenticeship learning.

In 1999, Schaal [110] discussed the analogy between robot learning from demonstration and imitation learning[2] from the viewpoint of behavioral and cognitive science [88] as well as imitation from the viewpoint of neuroscience and cognitive neuroscience [98]. After that, many studies focused on imitation learning in robotics [120, 5, 14].

In the mid 90s, Tomasello [125] proposed a new process of learning from demonstration in observational learning, called emulation learning. Accordingly, in 2004 Thompson and Russell [123] proposed a more solid classification of different processes of observational learning. However, this classification did not completely clarify the complex processes of observational learning. After that, in 2009 Whiten et al. [135] described the classification with more details. This is still an ongoing research in psychology and cognitive neuroscience as well as behavioral and cognitive science.

2.3 Observational learning

Observational learning is a means of learning through which humans learn basic skills from observing others. In this way, a person learns how to perform a task by watching another person doing that task. In the following, we present some useful definitions of different types of observational learning (Figure 2.2) relevant to robot learning from demonstration. These definitions are discussed here because a classification of the existing methods of robot learning from demonstration can be built based on these types of observational learning. The existing methods of robot learning from demonstration are then categorized based on the defined classification.

As mentioned earlier, observational learning is a very basic type of human learning through which a toddler obtains very basic skills of performing a task by watching others doing that task. For example, a toddler learns by observational learning to manipulate an object. In the early 1990s, imitation learning was widely thought as the main means of learning from demonstration in the psychology community. After that, Whiten et al. [135], Thompson and Russell [123] established a more detailed classification of observational learning, distinguishing imitation learning from

[2]Imitation learning is a subset of observational learning.

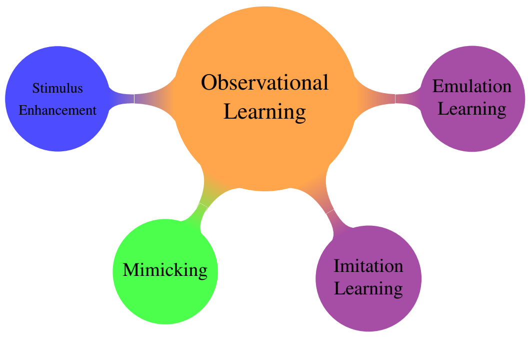


Figure 2.2: *Schema of different types of observational learning proposed by Thompson and Russell [123].*

other types of observational learning. These different types, defined in the following, are useful to better understand different methods of robot learning from demonstration. In the following definitions, models are people who perform a task when a person, called observer, watches them. Four main learning categories of observational learning have been distinguished by Thompson and Russell [123] and Whiten et al. [135], as follows:

1. In stimulus enhancement, the model's action does no more than draw the observer's attention to the object or part of the apparatus that has to be manipulated. During this process observer does not notice what has to be done and does not take an action [123].

2. *Mimicking* is the copying of a model's bodily movements. Thompson and Russell [123] mentioned that mimicking must involve no conceptualization on the part of the observer of the purpose of the action. Hence, the observer may or may not perform the task.

3. Whiten and Ham [134] defined *imitation learning* as goal oriented copying the form of an observed action. In imitation learning an observer is assumed to recognize what the form of the model's movements is bringing about and use that to carry out the task. Nonetheless, affordance learning may not be involved at all [123].

4. In *emulation learning*, the observer replicates, as a means to an end, not the bodily movements of the model, but the dynamic result of the model's action, having noted how the manipulated element caused the desired result. What is conceptualized here is both the causal power of the affected element and the kind of actions that the element affords [123]. In 1989, emulation learning was originally distinguished

from imitation learning by Wood [136]. In 1996, Tomasello [125] recognized emulation learning as a method of observational learning by which the observer learns the environmental results of the model's action. In 2009, Whiten et al. [135] recognized and classified different types of emulation learning, such as end-state emulation, affordance learning and object movement re-enactment. In 2013, Gibson [46] defined emulation as follows. In emulation, the learner matches the achievement of a conspecific by focusing not on the conspecific's behavioral strategy, but on the environmental outcomes caused by its actions, and does so by perceiving the dynamic affordances and learning a model of environmental outcome based on the taken actions.

These definitions may seem not so clear because the corresponding processes are complex. Furthermore, they are defined by psychologists. Hence, in the following sections according to the presented definitions, we define the corresponding terms in robotics. Although the proposed definitions of the corresponding terms may not be complete, we try to formalize a simple definition that distinguishes different processes of learning from demonstration. There are still ongoing research studies on better understanding the difference between imitation and emulation, and separating imitation and emulation learning is not easy in some cases.

The correspondence between robot learning from demonstration and observational learning in psychology as a consequence of the appearance of humanoid robots initiated many studies in robotics. People, working in robotic, were initially inspired from imitation learning [8] to develop necessary methods of robot learning from demonstration, because at that time imitation learning was the only definite types of observational learning allowing an observer to generalize an observed behavior to a new situation.

After that in the late 90s, psychologist recognized emulation learning as a key component of observational learning that is different from imitation learning [125]. Hence, it is necessary to use the updated classification of observational learning in robotics to develop necessary method for the proposed emulation learning.

In this chapter we try to take the most solid definitions of the different observational learning processes consistent with the different types of robot learning from demonstration. Consequently, in the following sections, we present the corresponding processes of robot learning from demonstration to the processes of observational learning resulting in a multi-layered robot learning from demonstration, as mentioned in Chapter 1.

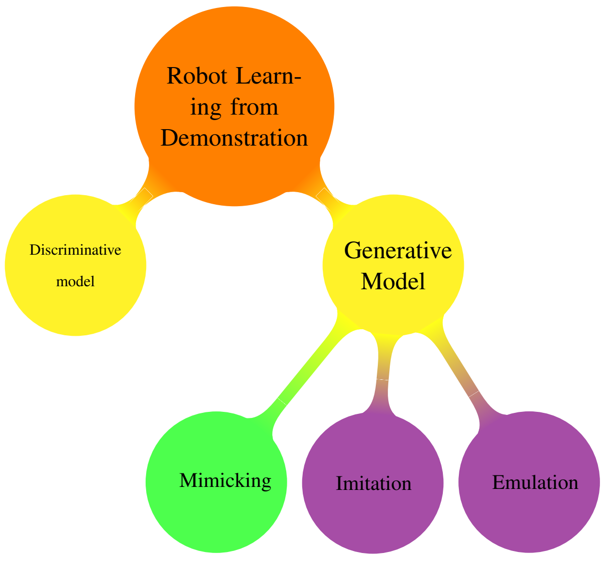


Figure 2.3: *Schema of Robot Learning from Demonstration and corresponding system identification approach, correspondence are depicted by hollow bars.*

2.4 Towards robot learning from demonstration

For designing a framework in which a robot learns how to perform a task from a set of task demonstrations, Alissandrakis et al. [5] proposed five central questions to be answered: When to imitate? Who to imitate? What to imitate? How to imitate? How to evaluate a successful imitation?

In this section, we benefit from these five questions to describe a taxonomy of robot learning from demonstration. In particular, we translate these questions about imitation into five corresponding questions of robot learning from demonstration:

1. What to learn from demonstration?
2. How to learn from demonstration?
3. What is the model?
4. When to learn from demonstration?
5. How to evaluate a successful reproduction?

2.4.1 What and how to learn from demonstration?

The question of 'What and how to learn from demonstration?' can be discussed based on the different types of observational learning in psychology.

Four different types of robot learning from demonstration are thus recognized in this section, as follows.

Robot learns a model to discriminate

In the first type of robot learning from demonstration shown in Figure. 2.3, a robot learns a model to classify or cluster a data set. The learned model may be used to recognize a pattern or structure of a given data set [3]. For instance, we identify the methods of object or activity recognition, enabling a robot to recognize an object or an activity of a person, as robot learning a model to discriminate from a set of given demonstrations [132, 65]. This type of robot learning from demonstration corresponds to stimulus enhancement in observational learning.

Mimicking

The second type of robot learning from demonstration called mimicking enables a robot to learn how to perform a task where the reproduction and demonstration conditions are identical. In this way, robot plays back exactly the recorded data. This corresponds to programming by guiding, discussed in Chapter 1. For example, in the framework of robot learning from demonstration, one may record some important key points of a task and robot uses a planning algorithm to reach these key points, or one may record the motor current of each robot joint motor during a demonstration and robot plays back the recorded motor current to repeat the same trajectory. So, a robot builds a model of a demonstrated task to replicate the same demonstrated performance in mimicking. This replication brings the task goal about if the environmental condition is invariant.
In this type of robot learning from demonstration, the robot has neither the ability to recognize the goal of the movement nor the ability to generalize the demonstrated task to slightly different conditions. For example, consider a task of picking an object at A and placing it at B. In this example, if a programer tele-operates the robot to pick the object while the robot is recording the followed trajectory, the task can be successfully repeated as long as the position of the object during the demonstration and reproduction are identical. Otherwise, the robot is not able to pick the object. People used successfully robot learning through mimicking in industry to program a robot to repeat exactly the same trajectory. As mentioned in Chapter 1, this was called programming by guiding, *walk through* and *lead through*.

[3]In Figure. 2.3 and Figure. 2.2 corresponding components are represented with the same color.

If demonstrations are noisy, robot has to build a model from demonstrations that produces a smooth trajectory or path. In this regard, Calinon [22] used Gaussian Mixture Model/Gaussian Mixture regression to compute a generalized trajectory across a set of demonstrations.

Imitation learning

The third approach of robot learning from demonstration, called imitation learning, enables a robot to learn how to perform a task and generalize it to a slightly different situation. In this approach, the robot is capable of recognizing the goal of the movement. Hence, the robot builds a model based on the demonstrations to generate the sufficient and necessary actions, trajectory or path to reach the goal of the demonstrated task. In the example of picking and placing an object, the robot recognizes a new position of the object A' and position of the goal B' and it generates the necessary path to be followed based on the demonstrated path.

Among the methods of robot learning from demonstration at the subsymbolic-level, Dynamic Movement Primitives and Gaussian mixture model/Gaussian mixture regression has been largely studied in this context. In particular, Schaal [111] studied Dynamic Movement Primitives to generate a trajectory with a new start and goal points based on a single demonstrated trajectory. Calinon et al. [27] studied Gaussian mixture model/-Gaussian mixture regression to account for temporal and spatial constraints across different demonstrations by combining Gaussian mixture models of different tasks. The obtained model is then used to produce a generalized trajectory across a set of demonstrated trajectories. This method encodes invariant constraints into a model across different tasks.

Emulation learning

Before the early 90s, imitation learning was recognized as the highest level of observational learning in psychology community[4]. However, it does not describe the complex process of observational learning in the situations in which a learner does not copy the form of the model's movements. For example, a person may learn from demonstration how to open a door by rotating a door handle and use the learned model to push a door handle to open a different door. In this example, the person learns affordance of the door handle by emulation, and generalize the learned model across different

[4]There are common processes of observational learning between human and apes [125]. Psychologists believe that humans use more imitation learning where apes use emulation learning to learn by watching. That is why they mention that imitation learning is the highest level of observational learning that make human more successful in developing skills and cultures [20].

Table 2.1: *Studies that are classified based on the classification of robot learning from demonstration discussed in this chapter, see Figure 2.3.*

Robot learning from demonstration			
Model to discriminate	**Models that generate action, path or trajectory**		
	Mimicking	**Imitation**	**Emulation**
Viola et al. [132], 2001	Calinon et al.[27], 2007	Ijspeert et al. [56], 2002	Abbeel et al. [1], 2004
Vail et al. [130], 2007	Calinon et al. [24], 2008	Schaal et al. [112], 2005	Ziebart et al. [141], 2008
Sethi et al. [116], 2009	Muhlig et al. [92], 2009	Ito et al. [60], 2006	Ratliff et al. [106], 2009
Murphy et al. [93], 2009	Calinon [22], 2009	Ude et al. [129], 2010	Dvijotham et al. [40], 2010
Yang et al. [138], 2011	Calinon et al. [29], 2010	Nemec et al. [95], 2012	Levine et al. [75], 2011
Kitani et al. [65], 2012	Cho and Jo [32], 2013	Ijspeert et al. [57], 2013	Levine et al. [74], 2012

types of door handles to open different door types. Lopes et al. [81] studied affordance learning of different objects in the context of robot learning from demonstration.

The fourth approach of robot learning from demonstration, called emulation learning, enables a robot to learn how to respond to different environmental features while performing a task to achieve the goal of demonstrated task. In 1996, Tomasello [125] described a process of observational learning, called emulation learning, and studied the difference between imitation and emulation learning. For example, one of the major concerns in robotics is the problem of obstacle avoidance. In the example of picking and placing an object, the robot can learn by emulation how to respond to the measured distance from different objects.

In many robotic tasks, robot must learn a model that is task dependent as well as environment dependent. For example, robot learns from demonstration how to assemble some production parts in an unstructured environment that entails performing a learned task of assembly and avoiding obstacles. Imitation learning enables a robot to learn a task dependent model where emulation enables the robot to learn an environment dependent model of a demonstrated task.

In order to clarify the difference between the last three types of robot learning from demonstration in Section 2.4.1, consider the following example. Assume that a person (the trainee) wants to put a spoon of sugar from cup A in cup B, where *A* and *B* are on a table. If the trainee was to learn by *mimicking*, it would follow exactly the same path as the model did, and would be successful so long as neither cup A nor cup B are moved. Given some noisy demonstrations, Calinon [22] studied Gaussian Mixture Model/Gaussian Mixture regression to compute a generalized trajectory across the demonstrations. This method can be considered as mimicking given a set of trajectories.
On the other hand, if the trainee was to use *imitation learning*, s/he would

be able to successfully carry out the task no matter where the cups were. In this case, the trainee is able to extract a task model given a set of task demonstrations, recognize the state of the goal, which are positions of the cups *A* and *B*, and map the observed behavior to the new states of the goal to do the job. Ijspeert et al. [56] proposed Dynamic Movement Primitives model allowing a robot to replicate a demonstrated trajectory or path with a new target point. In a scene with an obstacle, the nominal model learned by Dynamic Movement Primitives may no longer be feasible, so the trainee uses *emulation learning* to infer from features of the scene how and when to deviate from the nominal trajectory. Inverse reinforcement learning can be employed to encode the responses of a model to environmental features into a reward function. In Table 2.1 some of the related works to each distinguished robot LfD in this chapter are listed.

2.4.2 When to learn and what is the model?

The question of 'when and who to imitate' corresponds to the data set of demonstrations. An observer determines the data set of demonstrations by answering to these two questions. Nowadays, these questions are already answered by giving a dataset of demonstrations to a robot. A main issue arising about data set of task demonstration both in robotics [5] and cognitive neuroscience [108] is the problem of transformation of the demonstrations from model coordinate frame into the coordinate frame of the observer or robot.

In robotics, demonstrations may be presented either in self-frame (robot base frame) or in demonstrated-frame. For example, demonstrations are available in the self-frame in walkthrough and lead-through methods where the data are collected through joint angles. Moreover, due to development of required sensors, e.g. data gloves [127], laser range finder [58] and vision sensor [62], demonstrated-frame became available. The demonstrated data collected with sensors has to be transformed from demonstrated-frame into the robot base frame whereas robot does not need to transform the demonstrations collected in the self-frame. Di Pellegrino et al. [35] has also studied the hypothesis of transformation in observational learning.

2.4.3 How to evaluate a successful reproduction?

The question of 'how to evaluate a successful reproduction?' entails a feedback on robot performance. Either this feedback may be provided by an external evaluator, which is called teacher, or a robot can generate a signal of feedback based on comparing the results of his performance and the results of the demonstrations. In either case the evaluation entails employing learning approaches different from robot learning from demonstration, e.g. reinforcement learning.

2.5 Conclusion

In this chapter we discussed the connection between robot learning from demonstration and observational learning in psychology. Corresponding to the developed classification in observational learning, we defined a classification of robot learning from demonstration. This classification describes different processes of robot learning from demonstration. Although the main focus in this thesis is on robot learning at the subsymbolic-level, the defined classification can be extended to robot learning at the symbolic-level. In order to better describe the processes of robot learning from demonstration, we answered five central questions originally proposed in the context of imitation learning in robotics by Alissandrakis et al. [5]. The existing methods of robot learning from demonstration have been then categorized based on the defined classification. As robots are expected to perform some physical task, such as picking and placing an object, the processes of robot learning from demonstration that generate path or trajectories are the main focus of this thesis. These processes are learning from demonstration through mimicking, imitation and emulation.
In the next chapter, Gaussian mixture model/Gaussian mixture regression is presented as a method of robot learning from demonstration through mimicking with multiple demonstrations. Then, Dynamic movement Primitives is presented as a method of robot imitation learning with a single demonstration.

CHAPTER 3

Robot LfD: from mimicking to imitation Learning

3.1 Outline of the chapter

This chapter presents two methods of robot learning from demonstration through mimicking and imitation at the subsymbolic-level. We present Gaussian mixture model/Gaussian mixture regression as a method to compute a generalized paths given a set of noisy demonstrations for mimicking and Dynamic Movement Primitives for imitation learning model.

In robotics, many tasks are highly constrained in some parts of the corresponding paths or trajectory where the other parts are less constrained, resulting in different ways of task execution. Moreover, the collected data might be noisy due to measurement noise. In order to tackle these problems and compute an invariant basis of the demonstrated task performance, we present Gaussian mixture model/Gaussian mixture regression as a method that computes a generalized path given a set of demonstrated paths that can be used for robot learning from demonstration through mimicking.

Dynamic Movement Primitives, inspired by biological systems, has been introduced by Ijspeert et al. [56]. We use DMP as an imitation learning model. This model encodes an observed behavior represented by a trajectory into a dynamical system model with an attractor, which is a goal point. The main advantage of Dynamic Movement Primitives is that it generates a trajectory with a new determined goal position. The stability of the generated trajectory under external perturbation has been also discussed by Ijspeert et al. [56]. This model provides a stable trajectory generation with attractive landscape or an attractor point. Accordingly, the learned model from a demonstrated trajectory by Dynamic Movement Primitives is used to generate a corresponding trajectory to a new goal point.

This chapter is organized as follows. In Section 3.2 the joint use of Gaussian mixture model and Gaussian mixture regression is presented in order to compute a generalized trajectory from a set of demonstrated trajectories. Next, in Section 3.3 Dynamic Movement Primitives, a method to encode a demonstrated trajectory into a dynamical system model, is discussed. This model is then used to generate a trajectory with a new goal point. An example of human walking task is presented in Section 3.4. In this example, a data set of positions of humans during walking is collected. Gaussian mixture model and Gaussian mixture regression are then used to compute an invariant basis of the set of collected data set. To do so, first the data set is encoded into a set of Gaussian model components. Gaussian mixture regression is then used to computed a generalized path that is invariant basis of the demonstrated task. The computed generalized path is used to learn a Dynamic Movement Primitives model that is used to generate a path to a new goal point.

3.2 Gaussian mixture model/Gaussian mixture regression

The theory of Gaussian mixture model and Gaussian mixture regression has been largely studied in machine learning community by McLachlan and Peel [87, 86], Dasgupta and Schulman [33], Ghahramani and Jordan [45]. The application of this method in the robot learning from demonstration domain has been then studied by Calinon [22] to encode an observed behavior characterized by a set of demonstrated trajectories into a model. In this method, an invariant basis of a set of demonstrated trajectories is encoded into a set of Gaussian components. Gaussian mixture regression is then used to compute a generalized trajectory across the set of demonstrated trajectories. This computed trajectory may be considered as an optimal way to carry out the task based on the demonstrations. The obtained generalized trajectory captures the invariant constraints across all demonstrated trajectories.

As there is no generalization of demonstrated trajectories to a new goal point, we classify this method as robot learning from demonstration through mimicking given multiple demonstrations. Nonetheless, this method enables a robot to compute a generalized trajectory across multiple demonstrated trajectories. At the lowest level of mimicking, a robot plays back only a recorded trajectory that may be noisy or may not be the optimal trajectory. GMM/GMR enables a robot to compute a trajectory that captures the invariant information across all the demonstrated trajectories.

3.2.1 Gaussian mixture model (GMM)

Consider n observed demonstrations of a task performed by a human, consisting of n observations of time dependent sensory data. Each demonstration is rescaled to the fixed number of observed data points T. The total number of collected datapoint is then given by $N = nT$. Therefore, observed data set is $\zeta = \{\zeta_j\}_{j=1}^{N}$ where $\zeta_j \in \mathbb{R}^D$ and D is the dimensionality of each datapoint. The subscript notation t and s is used to define temporal or spatial variables of each datapoint $\zeta_j = \{\zeta_s, \zeta_t\}$ consisting of a temporal value $\zeta_t \in \mathbb{R}$ and a spatial vector $\zeta_s \in \mathbb{R}^{D-1}$.

The dataset ζ is modeled by a mixture of K components, defined by the probability density function:

$$p(\zeta_j) = \sum_{k=1}^{K} p(k)p(\zeta_j|k) \tag{3.1}$$

where $p(k)$ is a prior and $p(\zeta_j|k)$ is a conditional probability density function. For a mixture of K Gaussian distributions of dimensionality D, the parameters in equation (3.1) are defined by

$$\begin{aligned} p(k) &= \pi_k \\ p(\zeta_j|k) &= \mathcal{N}(\zeta_j; \mu_k, \Sigma_k) = \\ \frac{1}{\sqrt{(2\pi)^D \mid \Sigma_k \mid}} & e^{\frac{\left((\zeta_j-\mu_k)^T \Sigma_k^{-1} (\zeta_j-\mu_k)\right)}{2}} \end{aligned} \tag{3.2}$$

where $\mathcal{N}(\zeta_j; \mu_k, \Sigma_k)$ represents the probability of a datapoint ζ with respect to the normal distribution $\mathcal{N}(\mu, \Sigma)$. The Gaussian mixture model parameters, including prior probabilities, mean vectors and covariance matrices, are described by $\{\pi_k, \mu_k, \Sigma_k\}$.

3.2.2 Gaussian mixture regression (GMR)

After encoding the temporal and spatial values of a set of trajectories in a set of Gaussian models, Gaussian mixture regression is introduced to retrieve a smooth trajectory from the observed trajectories. To do so, the observed data is first modeled by a joint probability distribution $p(\zeta_s \mid \zeta_t)$. A generalized trajectory is then computed by estimating $\mathbb{E}[p(\zeta_s \mid \zeta_t)]$, and $cov(p(\zeta_s \mid \zeta_t))$ is used to estimate the constraints of performing the task, where $\mathbb{E}$ and cov are expectation and covariance, respectively.

In regression problem, a set of input variables $X \in \mathbb{R}^p$ and response variable $Y \in \mathbb{R}^q$ are given, where p and q are the dimensionality of model input and output. The regression methods aim at estimating the conditional expectation of Y given X based on a set of observations $\{X, Y\}$. We consider a set of observation $\{\zeta_t, \zeta_s\}$ where ζ_s is a vector of positions at time ζ_t. In the context of Gaussian mixture regression, regression aims at maximizing the conditional expectation of ζ_s given ζ_t. The generalized trajectory is obtained by computing conditional expectation of ζ_s at each time step. In this way, a joint density of the set of trajectories is first estimated by a Gaussian mixture model and then a regression approach is used to compute a generalized trajectory. Consider spatial and temporal values of a Gaussian component as follows:

$$\mu_k = \{\mu_{t,k}, \mu_{s,k}\}\ ,\ \Sigma_k = \begin{pmatrix} \Sigma_{tt,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{ss,k} \end{pmatrix}.$$

For each component k, the expected distribution of $\zeta_{s,k}$ given the temporal value ζ_t is given by theorem of Gaussian conditioning based on linear combination property of Gaussian distributions, as follows.

$$\begin{aligned} p(\zeta_{s,k}|\zeta_t, k) &= \mathcal{N}(\zeta_{s,k}; \hat{\zeta}_{s,k}, \hat{\Sigma}_{ss,k}) \\ \hat{\zeta}_{s,k} &= \mu_{s,k} - \Sigma_{st,k}\left(\Sigma_{tt,k}\right)^{-1}(\zeta_t - \mu_{t,k}), \\ \hat{\Sigma}_{ss,k} &= \Sigma_{ss,k} - \Sigma_{st,k}(\Sigma_{tt,k})^{-1}\Sigma_{ts,k}, \end{aligned}$$

The K Gaussian distributions $\mathcal{N}(\hat{\zeta}_{s,k}, \hat{\Sigma}_{ss,k})$ are mixed according to prior β_k

$$p(\zeta_s \mid \zeta_t) = \sum_{k=1}^{K} \beta_k \mathcal{N}(\zeta_s; \hat{\zeta}_{s,k}, \hat{\Sigma}_{ss,k}) \tag{3.3}$$

where $\beta_k = p(k \mid \zeta_t)$ is determined by the probability of the component k to be responsible for ζ_t

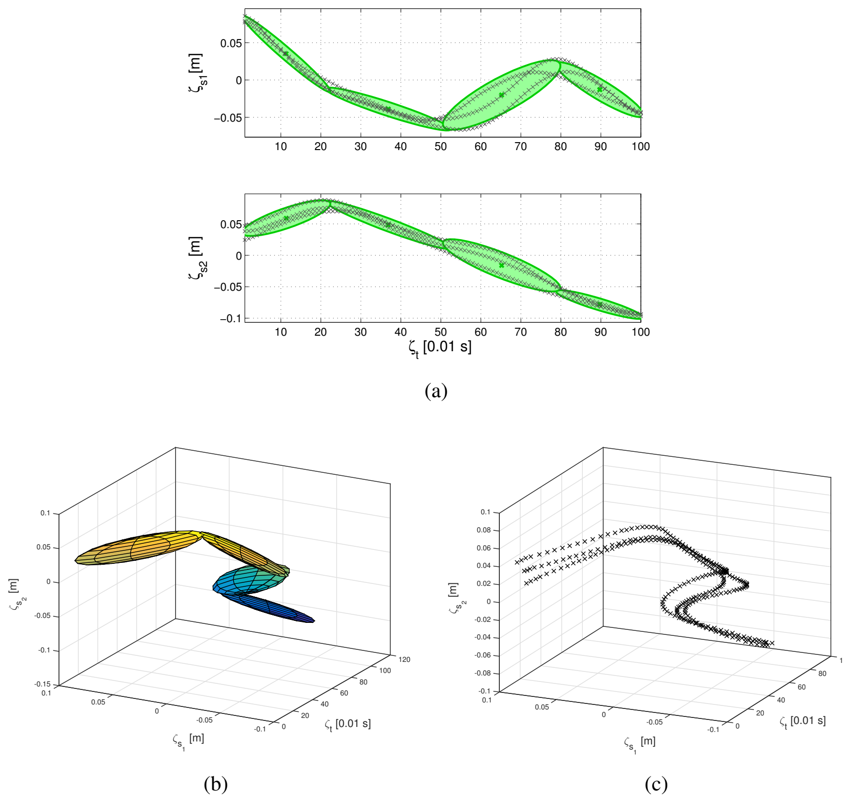


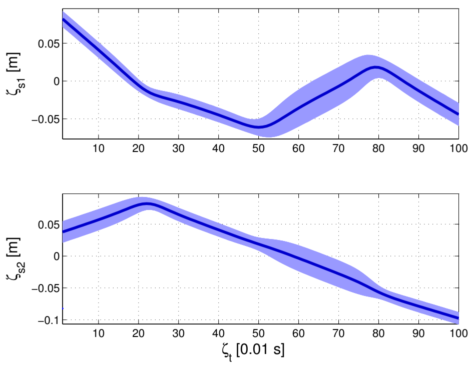
Figure 3.1: *(a) A set of three sample 2-D trajectories and the computed Gaussian components for ζ_{s1} and ζ_{s2}; (c) A set of three sample 2-D paths and the computed Gaussian components in the plane.*

$$\beta_k = \frac{p(k)p(\zeta_t|k)}{\sum_{i=1}^{K} p(i)p(\zeta_t \mid i)} = \frac{\pi_k \mathcal{N}(\zeta_t; \mu_{t,k}, \Sigma_{tt,k})}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\zeta_t; \mu_{t,i}, \Sigma_{tt,i})}$$

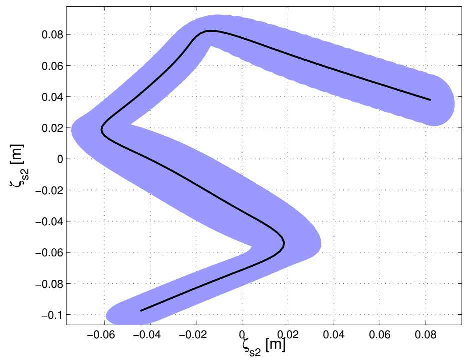
An estimation of conditional expectation of ζ_s given ζ_t in equation (3.3) is computed by using the linear combination property of Gaussian distribution by $p(\zeta_s|\zeta_t) \sim \mathcal{N}(\hat{\zeta}_s, \hat{\Sigma}_{ss})$, where parameters of the Gaussian distribution are defined by

$$\hat{\zeta}_s = \sum_{k=1}^{K} \beta_k \hat{\zeta}_{s,k} \quad , \quad \hat{\Sigma}_{ss} = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{ss,k}. \tag{3.4}$$

A trajectory and associated covariance matrix $\hat{\Sigma}_{ss}$ is then reproduced by evaluating $\{\hat{\zeta}_s, \hat{\Sigma}_{ss}\}$ at different ζ_t.



(a)



(b)

Figure 3.2: *(a) The computed trajectory by Gaussian mixture regression based on the computed Gaussian mixture model for ζ_{s1} (top) and ζ_{s2} (bottom) versus time ζ_t; (b) The corresponding computed Gaussian mixture regression in 2-D. The blue shaded area represents the computed covariance matrix of the Gaussian mixture regression model.*

3.2.3 Learning of GMM parameters

Expectation-Maximization algorithm [34] has been proposed to compute the parameters of the Gaussian mixture model. Consider equation (3.2) and equation (3.1), $p_{k,j}$ can be defined as posterior probability $p(k \mid \zeta_j)$ computed using the Bayes theorem $p(k \mid \zeta_j) = \frac{p(k)p(\zeta_j)}{\sum_{i=1}^{K} p(i)p(\zeta_j|i)}$. An initial estimate of the parameters used for iterative algorithm $\theta = \{\pi_k, \mu_k, \Sigma_k, E_k\}$, where E_k is the sum of the posterior probabilities, is computed by $k-means$ segmentation [84]. The parameters are then iteratively computed

$E - step$ $p_{k,j}^{(r+1)} = \frac{\pi_k^{(r)} \mathcal{N}(\zeta_j; \mu_k^{(r)}, \Sigma_k^{(r)})}{\sum_{i=1}^{K} \pi_k^{(r)} \mathcal{N}(\zeta_j; \mu_i^{(r)}, \Sigma_i^{(r)})}$ $E_k^{(r+1)} = \sum_{j=1}^{N} p_{k,j}^{(r+1)}$
$Maximization - step$ $\pi_k^{(r+1)} = \frac{E_k^{(r+1)}}{N}$ $\mu_k^{(r+1)} = \frac{\sum_{j=1}^{N} p_{k,j}^{(r+1)} \zeta_j}{E_k^{(r+1)}}$ $\Sigma_k^{(r+1)} = \frac{\sum_{j=1}^{N} p_{k,j}^{(r+1)} (\zeta_j - \mu_k^{(r+1)})(\zeta_j - \mu_k^{(r+1)})^T}{E_k^{(r+1)}}$

Table 3.1: *Expectation maximization algorithm that is used in [22] to learn the GMM parameters.*

until convergence by expectation-maximization algorithm[1] reported in Table 3.1.

The iterations stop when the improvement in log-likelihood is less than a threshold.

$$\frac{\mathcal{L}^{(r+1)}}{\mathcal{L}^{(r)}} < C$$

where $\mathcal{L}^{(r)}(\zeta, \theta) = \sum_{j=1}^{N} log\left(p\left(\zeta_j \mid \theta^{(r)}\right)\right)$.

3.3 Dynamic movement primitives

Dynamic movement primitives (DMPs) are a method of learning from demonstration through imitation at the subsymbolic-level [57]. Inspiring from the biological concept of motor primitives [47], Ijspeert et al. [56] introduced DMPs.

In the domain of cognitive science, it is proposed that a skill consists of many movement primitives [64]. Hence, one may adapt an obtained skill by combining the movement primitives in a new situation.

The processes of learning dynamic movement primitives were inspired from movement primitive. DMPs aim at finding some movement primitives based on a demonstration that can produce a new trajectory to a new goal point. A stable linear dynamical system that has an attractor is used to build dynamic movement primitives. In order to encode a trajectory, an external

[1]For further details of the expectation maximization algorithm refer to [22].

input added to the linear dynamical system. This input is generated by a nonlinear function learned from the demonstration and is fixed during all reproductions of the demonstrated task to different goal points. Dynamic movement primitives has been proposed to encode rhythmic as well as non-rhythmic behaviors. In the following, we present a discrete Dynamic movement primitives formulation [104], i.e. DMPs which encode non-rhythmic or episodic trajectories.

DMPs are composed of two differential equations, a first order differential equation, called canonical system, and a second order differential equation. The canonical system is independent where the second order system depends on the output of the canonical system. Consider a linear dynamical system represented by a second order differential equation as follows:

$$\tau\ddot{y}(t) = k\left(g - y(t)\right) - c\dot{y}(t) \tag{3.5}$$

where τ, k and c are mass, spring constant and damping factor of a simple mass-spring system, respectively. Moreover, g is a goal point of a new reproduction of the demonstrated task. The damping factor c, spring constant k and τ are chosen such that the equation (3.7) becomes a critically damped dynamical system. We add an external input u to the dynamical system.

$$\tau\ddot{y}(t) = k\left(g - y(t)\right) - c\dot{y}(t) + \left(g - y(0)\right)u \tag{3.6}$$

where $y(0)$ is the initial position. We consider $x_1(t) = y(t)$ and $x_2(t) = \dot{y}(t)$, hence, the state space representation of equation (3.6) is given by

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-k}{\tau} & \frac{-c}{\tau} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k}{\tau} \end{bmatrix} g + \begin{bmatrix} 0 \\ \frac{g-y(0)}{\tau} \end{bmatrix} u(t) \tag{3.7}$$

The first order differential equation is used to compute the external input u in combination with a set of basis functions, as follows:

$$\tau\dot{Z} = -\alpha Z \tag{3.8}$$

where Z and α are state of the differential equation and a predefined constant, respectively. α is selected such that s monotonically goes from initial stat $Z(0) = 1$ to $Z(T_1) = 0$ during trajectory production ($[x_1, x_2]^T$) (see Figure 3.3).

3.3.1 Learning of the external input

The external input in equation (3.7) is selected to be nonlinear in the state Z of the differential equation in (3.8) and it transforms the simple dynamics of

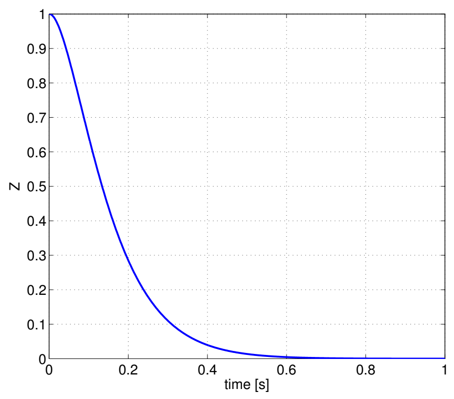


Figure 3.3: *Evolution of the canonical state Z during the trajectory reproduction.*

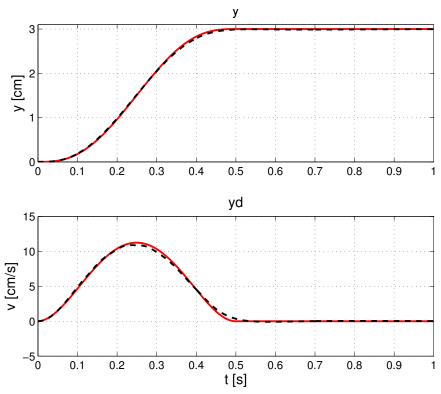
the unforced system in (3.5) into a forced system with a desired trajectory (equation (3.6)). The external input u is active in a finite time window (T_1) leading to a dynamical system with an attractor (g), i.e. $u(t) = 0 \; \forall \; t > T_1$. In order to learn a new skill represented by a trajectory, the data of a desired movement $x_{1,d}$ is collected and its velocities $x_{2,d}$ and acceleration $\dot{x}_{2,d}$ are computed at each time step $t = 0, ..., T$. Second, the canonical system is computed by using equation (3.8) with the selected temporal scaling factor τ. The target external input u_d of the desired movement is then computed (see Figure 3.5) based on equation (3.7) as follows:

$$\begin{bmatrix} 0 \\ \frac{g_d - x_{1,d}(0)}{\tau} \end{bmatrix} u_d = \begin{bmatrix} \dot{x}_{1,d} \\ \dot{x}_{2,d} \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ \frac{-c}{\tau} & \frac{-k}{\tau} \end{bmatrix} \begin{bmatrix} x_{1,d} \\ x_{2,d} \end{bmatrix} - \begin{bmatrix} 0 \\ \frac{k}{\tau} \end{bmatrix} g \quad (3.9)$$

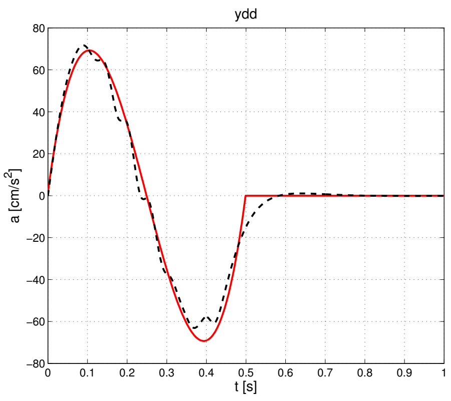
where $x_{1,d}$ and $x_{2,d}$ are position and velocity of the demonstrated trajectory. Equation (3.9) can be also written as follows:

$$\begin{aligned} u_d(t) &= \frac{\tau \dot{x}_{2,d}(t) - k(g_d - x_{1,d}(t)) + c x_{2,d}(t)}{(g - x_{1,d}(0))} \\ u_d(t) &= \frac{\tau \ddot{y}_d(t) - k(g - y_d(t)) + c \dot{y}_d(t)}{(g - y_d(0))} \end{aligned} \quad (3.10)$$

where $x_{1,d}(0)$ and g_d are initial and goal positions of the demonstrated trajectory. In order to encode different trajectories into dynamic movement primitives model, the external input is defined as a function of the state of the canonical system, as follows.



(a)

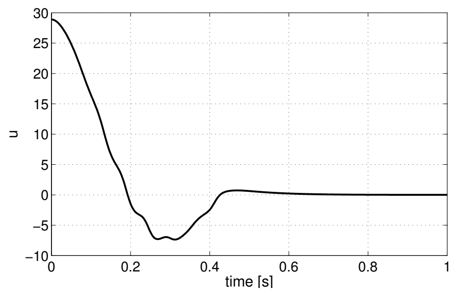


(b)

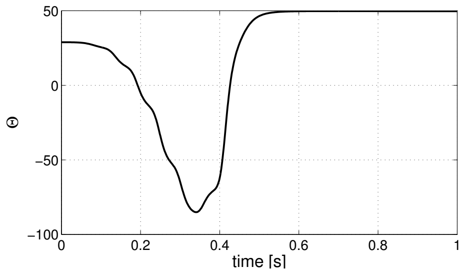
Figure 3.4: *(a) A demonstrated minimum jerk trajectory (red line) and the reproduced trajectory by the learned DMPs model (black dashed line) (top), the corresponding velocities of demonstrated trajectory (red line) and the reproduced trajectory (black dashed line) (bottom); (b) corresponding acceleration of the demonstrated trajectory (red line) and the reproduced trajectory (black dashed line).*

$$\begin{aligned} u(t) &= \Theta(Z)Z(t) \\ \Theta(Z) &= \frac{\sum_i w_i \psi_i(Z)}{\sum_i \psi_i(Z)} \end{aligned} \tag{3.11}$$

where $\psi_i(Z) = e^{(-h_i(Z-c_i)^2)}$ is a fixed Gaussian basis function, with fixed centers c_is and widths h_is, and w_is are adjustable weights. c_is and h_is are design parameters and should be selected such that the basis functions spans the space of Z to generate the required external input u. The external input depends on the state of the canonical system (equation (3.11)) where



(a)



(b)

Figure 3.5: *(a) The external input that generates the demonstrated trajectory in Figure 3.4; (b) the learned Θ corresponding to the obtained external input u.*

it evolves autonomously based on the differential equation in (3.8). The complete state space representation is obtained from equation (3.7), (3.8) and (3.11) as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-c}{\tau} & \frac{-k}{\tau} & (\frac{g-x_1(0)}{\tau}) \times \Theta(Z) \\ 0 & 0 & \frac{-\tau}{\alpha} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k}{\tau} \\ 0 \end{bmatrix} g \tag{3.12}$$

Ijspeert et al. [56] discussed the stability of the solution of DMP and the convergence of the solution to the goal point. The weights w_i can be learned to generate every desired nonlinear trajectory. The model can be used to generate a trajectory with different goal position g similar to the demonstrated trajectory in the training phase without changing the weights w_i. The weights w_is are thus computed by minimizing the error criterion in equation (3.13).

$$J = \sum_t (u_d(t) - u(t))^2 \tag{3.13}$$

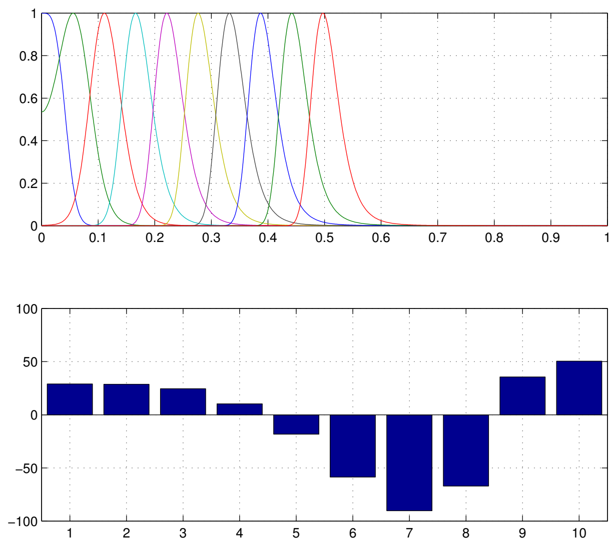


Figure 3.6: *10 considered Gaussian basis function for computing the Dynamic Movement Primitives model (top). A set of 10 weights of the considered Gaussian basis functions computed for the given trajectory in Fig, 3.4.*

The weights w_is can be computed in a closed form as follows.

$$F(t) = \tau \ddot{y}_d - k(g - y_d) + c\dot{y}_d$$
$$\mathbf{F} = \begin{bmatrix} F(t_1) \\ \vdots \\ F(t_T) \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \tag{3.14}$$

where $\ddot{y}_d$, $\dot{y}_d$ and y_d are acceleration, velocity and position of the desired or demonstrated trajectory. By using equation (3.14), equation (3.11) can be written in the following matrix form [129]:

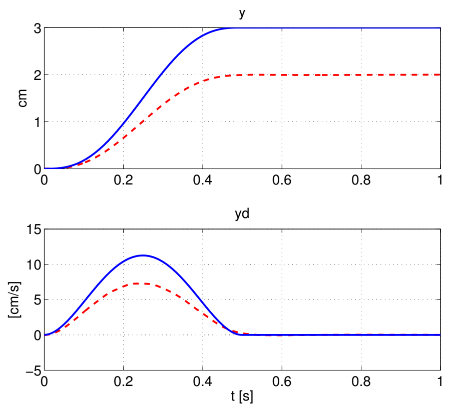
$$X\mathbf{W} = \mathbf{F}$$

where

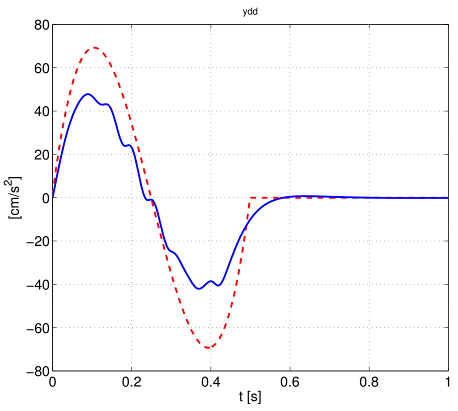
$$X = \begin{bmatrix} \frac{\psi_1(Z_1)}{\sum_{i=1}^{N} \psi_i(Z_1)} Z_1 & \dots & \frac{\psi_N(Z_1)}{\sum_{i=1}^{N} \psi_i(Z_1)} Z_1 \\ \dots & \dots & \dots \\ \frac{\psi_1(Z_T)}{\sum_{i=1}^{N} \psi_i(Z_T)} Z_T & \dots & \frac{\psi_N(Z_T)}{\sum_{i=1}^{N} \psi_i(Z_T)} Z_T \end{bmatrix}$$

Hence, the corresponding weights of the external input that minimizes the cost in equation (3.13) can be computed as follows.

$$\mathbf{W} = (\mathbf{X^T X})^{-1}\mathbf{XF}$$



(a)



(b)

Figure 3.7: *(a) Position (top) and velocity (bottom) of the demonstrated trajectory (red dashed line) and reproduced trajectory (blue line) with a new goal point generated by Dynamic Movement Primitives model; (b) corresponding acceleration of the demonstrated trajectory (red dashed line) and reproduced trajectory (blue line) with new point by Dynamic Movement Primitives model.*

In Figure 3.6 the considered basis functions and the obtained weights for a desired trajectory are shown. Furthermore, in Figure 3.4 the desired trajectory, its velocity and acceleration are shown. The target external input required to generate the desired trajectory is shown in Figure 3.5. A new trajectory with a new goal point can be generated using equation (3.12), where g equals to the new goal position. A major advantage of the proposed formulation is the robustness of the generated movements against perturbation, studied by Ijspeert et al. [56, 57]. Dynamic Movements Primitives has been also used to encode rhythmic movements by learning the weights of the model for an attractor landscape rather than point attractor [56].

To show how Dynamic Movement Primitives reproduce an observed trajectory with a new goal point a minimum jerk 1-D trajectory shown in Figure 3.4 is taken as a demonstrated trajectory. Dynamic Movement Primitives model is learned from the demonstrated trajectory. where the weights learned from the desired trajectory are shown in Figure 3.6. Moreover, the reproduced trajectory by the learned Dynamic Movement Primitives is shown in Figure 3.4. The obtained model is then used to generate a trajectory with a new goal point shown in Figure 3.7. This example illustrates how the leaned model of Dynamic Movement Primitives generates a smooth trajectory to a new goal point.

3.4 A case study

In this section, we take a set of demonstrated paths of a walking experiment into account. In this experiment, 5 volunteers were asked to walk from a point within circle A, in a room, to enter a corridor and reach a point within a circle B (see Figure 3.8). While we were collecting the positions of each volunteer, they repeated the task six times. Hence, a set of 30 paths constitutes the walking experiment data set. In this example, the main objective is to compute a path from a set of noisy demonstrations that a mobile robot must follow to reach a desired goal point.

Based on the proposed workflow for robot LfD (see Figure 1.4), we first use GMM/GMR to compute a generalized path across the set of demonstrated paths. For the sake of simplicity, we assume that all the experiments are completed in the same time $T = 20[s]$. We resampled every path to get 100 number of data points. The obtained mean value of the Gaussian components computed by the expectation-maximization algorithm reported in Table 3.1 are:

$$\mu_1 = \{13.3584, 1.4776, -0.3013\}$$

$$\mu_2 = \{7.0778, 3.0349, -0.0252\}$$

$$\mu_3 = \{2.1845, 4.2990, -0.0590\}$$

$$\mu_4 = \{18.1842, 0.4296, -0.8792\}$$

where $\mu_k = \{\mu_{t,k}, \mu_{s,k}\}, k = 1, ..., 4$, and the corresponding obtained covariance matrices are as follows:

$$\Sigma_1 = \begin{bmatrix} 4.6572 & -1.0930 & -0.3902 \\ -1.0930 & 0.2634 & 0.0863 \\ -0.3902 & 0.0863 & 0.0403 \end{bmatrix},$$

$$\Sigma_2 = \begin{bmatrix} 5.0746 & -1.2776 & -0.0249 \\ -1.2776 & 0.3244 & 0.0049 \\ -0.0249 & 0.0049 & 0.0043 \end{bmatrix},$$

$$\Sigma_3 = \begin{bmatrix} 1.7950 & -0.4545 & -0.0163 \\ -0.4545 & 0.1159 & 0.0039 \\ -0.0163 & 0.0039 & 0.0063 \end{bmatrix},$$

$$\Sigma_4 = \begin{bmatrix} 1.6186 & -0.1976 & -0.3317 \\ -0.1976 & 0.0294 & 0.0394 \\ -0.3317 & 0.0394 & 0.0700 \end{bmatrix}.$$

where

$$\Sigma_k = \begin{pmatrix} \Sigma_{tt,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{ss,k} \end{pmatrix}, k = 1, ..., 4.$$

Then a continuous generalized path is obtained by using equation (3.4) (Fig. 3.9).
In the case that a set of noisy task demonstrations is provided, performing the task by following the generalized path corresponds to the mimicking level of robot LfD. This generalized path can be used to move a mobile robot from point A to point B. We assume that the optimal time component of a demonstrated trajectory can be determined by considering minimum-jerk for the robot that performs the task [105, 71].

The considered Gaussian model components, as well as the computed path, are shown in Figure 3.9. The shaded area in the bottom figure in Figure 3.9 represents the variability of the demonstrated path computed by covariance matrix along the generalized path. The covariance matrices along the generalized path computed by equation (3.4) represent the constraints imposed by the environment during task performance. That is, the part of generalized path with low covariance matrix is more constrained than the part with higher covariance. This information allows to modify the generalized path where demonstrated paths have higher variation. In this example, the constraints are imposed by corridor walls shown by thick red lines in

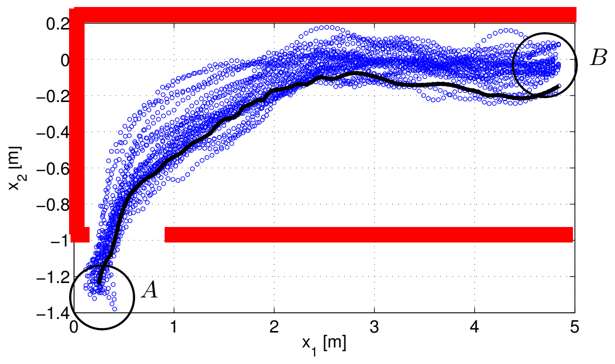


Figure 3.8: *Walking experiment data set, start and goal circle A and B and the constraints of the environment represented with thick lines. One demonstrated path is shown by black line and the others are shown by the corresponding data points.*

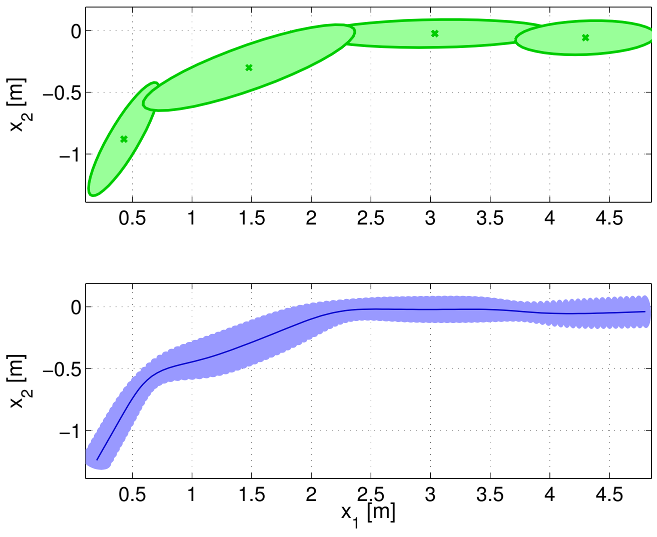


Figure 3.9: *The Gaussian mixture component computed based on the walking example dataset in Fig, 3.8 (top), the corresponding computed Gaussian mixture regression.*

Figure 3.8. Nonetheless, this generalized path is only useful as long as the goal point is invariant across all performances.

The robot learns a Dynamic Movement Primitives model using the computed generalized path, and uses that model to generate a new path with a new desired goal point. This corresponds to imitation component of robot LfD.

To do so, the path computed by GMM/GMR is used to learn Dynamic Movement Primitives model. First, each degree of freedom of the obtained

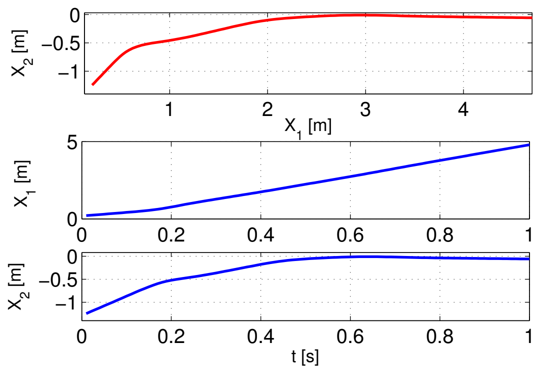


Figure 3.10: *The generalized path obtained by GMM/GMR (top figure), and the presentation of each DOF versus time (two bottom figures).*

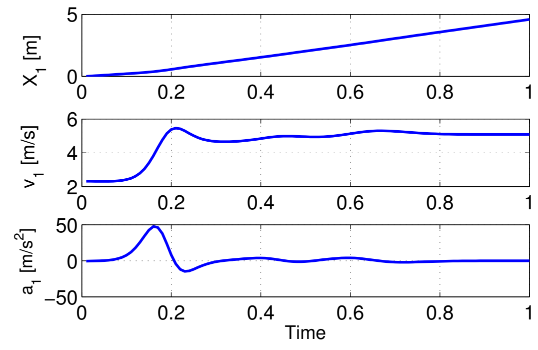


Figure 3.11: X_1 *versus time (top), its corresponding velocity (middle) and acceleration (bottom).*

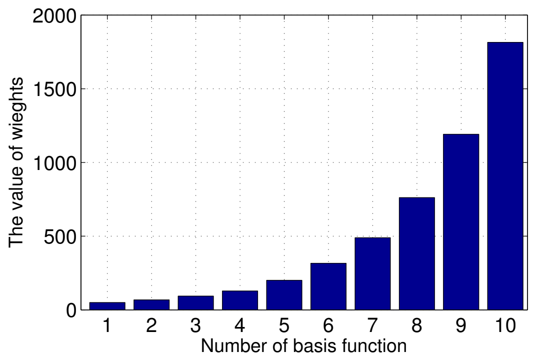


Figure 3.12: *The weights of the basis functions obtained from* X_1.

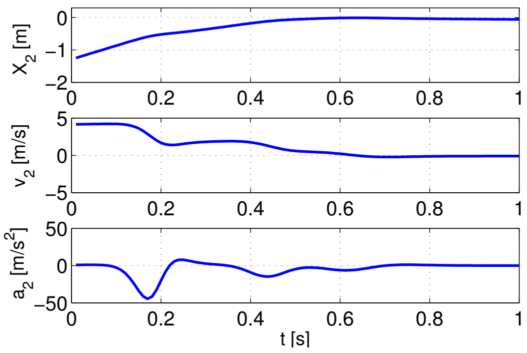


Figure 3.13: *X_2 versus time (top), its corresponding velocity (middle) and acceleration (bottom).*

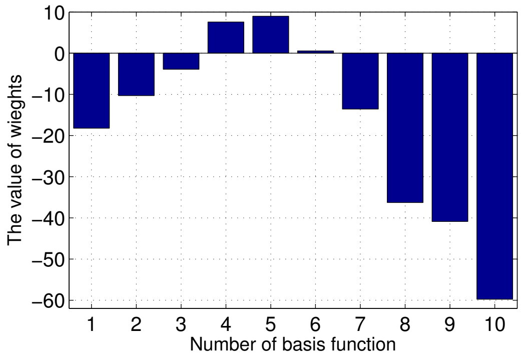
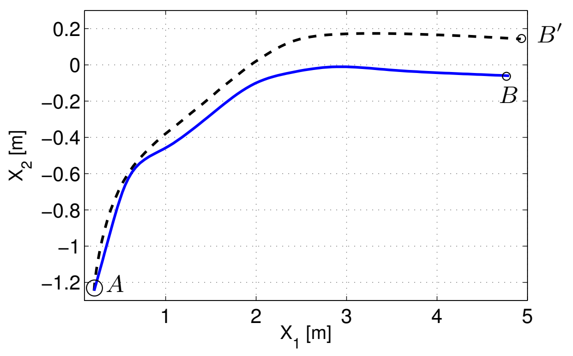


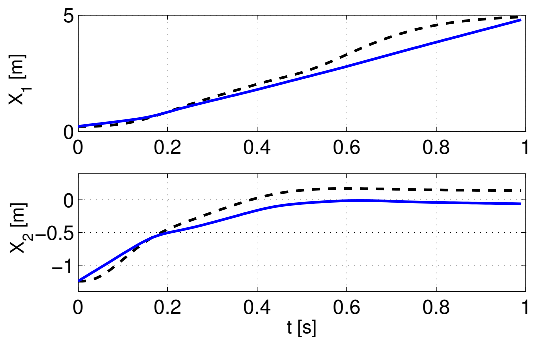
Figure 3.14: *The weights of the basis functions obtained from X_2.*

generalized path is taken into account (Fig. 3.10). We assign a fixed time step to every sample point of this path (Figs. 3.11 and 3.13), i.e. we assign every sample point P_i to time $i\Delta t, i = 1, ..., m$ and $T = m\Delta t$. Although we consider $T = 1$, the total duration considered to perform the task, it can be scaled to any desired value. The velocity and acceleration of each degree of freedom is then computed, shown in Figs. 3.11 and 3.13. We trained an MDP model for every degree of freedom where the obtained wights of the corresponding basis functions are shown in Figs. 3.12 and 3.14. The damping factor $c = 25$, spring constant $k = 156.25$ and $\tau = 1$ are chosen so that the equation (3.7) represents a critically damped system. The basis functions and the canonical system shown in Figures 3.3 and 3.6 are also used for this example.

The learned model is then used to generate a smooth path that reaches a new imposed goal point. Figure 3.15 shows the produced path by the



(a)



(b)

Figure 3.15: *(a) The generalized path obtained by GMM/GMR (blue solid line) starting from A and terminating at B, the path obtained by the learned DMP model to a new goal point B'; (b) The generated trajectory by the learned DMP model for X_1 (top) and X_2 (bottom).*

learned Dynamic Movement Primitives that satisfies the constraints of environment and reaches the new imposed goal point.

This example illustrates how two levels of workflow shown in Figure 1.4 are used to produce a generalized path to a new desired goal point from a set of noisy demonstrations. This is highly important, because human demonstrations are not always optimal. Hence, this workflow provides us with a practical tool to tackle the problem of noisy demonstrations and produces a path or trajectory to a new goal point.

3.5 Conclusion

In this chapter, we presented two methods of robot LfD corresponding to two level of the proposed workflow (Figure 1.4). In the first level, namely mimicking, a generalized path or trajectory is computed from a set of demonstrations. In the second level, namely imitation, the generalized path is used to produce a path to a new goal point.

We discussed Gaussian mixture model/Gaussian mixture regression as a method of robot learning from demonstration through mimicking. This method was developed in machine learning community and its application in robot learning from demonstration has been studied [22] to encode an observed behavior and produce a generalized trajectory. The main advantage of using this method is that invariant basis of a set of demonstrated trajectories can be encoded into the Gaussian components. These components are then used to compute a generalized trajectory by Gaussian mixture regression across all the demonstrated trajectories represented by covariance matrix along the generalized trajectory. The obtained generalized trajectory captures the invariant constraints across all demonstrated trajectories.

We then presented Dynamic Movement Primitives as a method of robot LfD corresponding to second level of the workflow. This model encodes an observed behavior, represented by a trajectory, into a dynamical system model. The main advantage of Dynamic Movement Primitives is that it generates a trajectory with an attractor, which is a goal point. The stability of the generated solution by DMPs, whose parameters are selected such that the main differential equation becomes critically damped, against external perturbation was studied by Ijspeert et al. [56, 57]. The learned model by Dynamic Movement Primitives can be used to generate necessary trajectory to accomplish a demonstrated task with a different goal point. This is mainly useful in the context of imitation learning proposed in Chapter 2.

We use these two methods of robot LfD to compute the necessary path from the experimental data set of the walking example. The path obtained by these methods can be then used to move a mobile robot from point A to either B or B'. This example illustrates that robot can learn a generalized path to a new goal point from a set of noisy demonstrations. For example, in picking and placing an object, a generalized path to place objects at different goal positions can be computed from a set of noisy demonstrations and based on sensory information about the goal position.

CHAPTER 4

Estimating a Mean-Path as a mimicking component

4.1 Outline of the chapter

In the previous chapter, Gaussian mixture model/ Gaussian mixture regression was presented as a means of computing a generalized path from a set of task demonstrations. We use this method as the mimicking component of the proposed workflow of robot LfD. In this chapter, we stress the precision of the computed generalized path (Figure 2.3), and we formalize a measure that allows us to evaluate the precision of the existing nonlinear regression methods, such as GMM/GMR. The main motivation of this chapter arose from industrial applications in which the precision of the computed generalized path is of utmost importance.

In the context of robot learning from demonstration, variability across multiple demonstrations has been studied by Calinon [22]. However, it is mainly focused on computing a generalized path across different tasks to encode invariant constraints across them for a humanoid robot. In this sort of applications, precision is not critical, but in industrial applications, such as deburring, the precision is critical. In order to compute the precision of

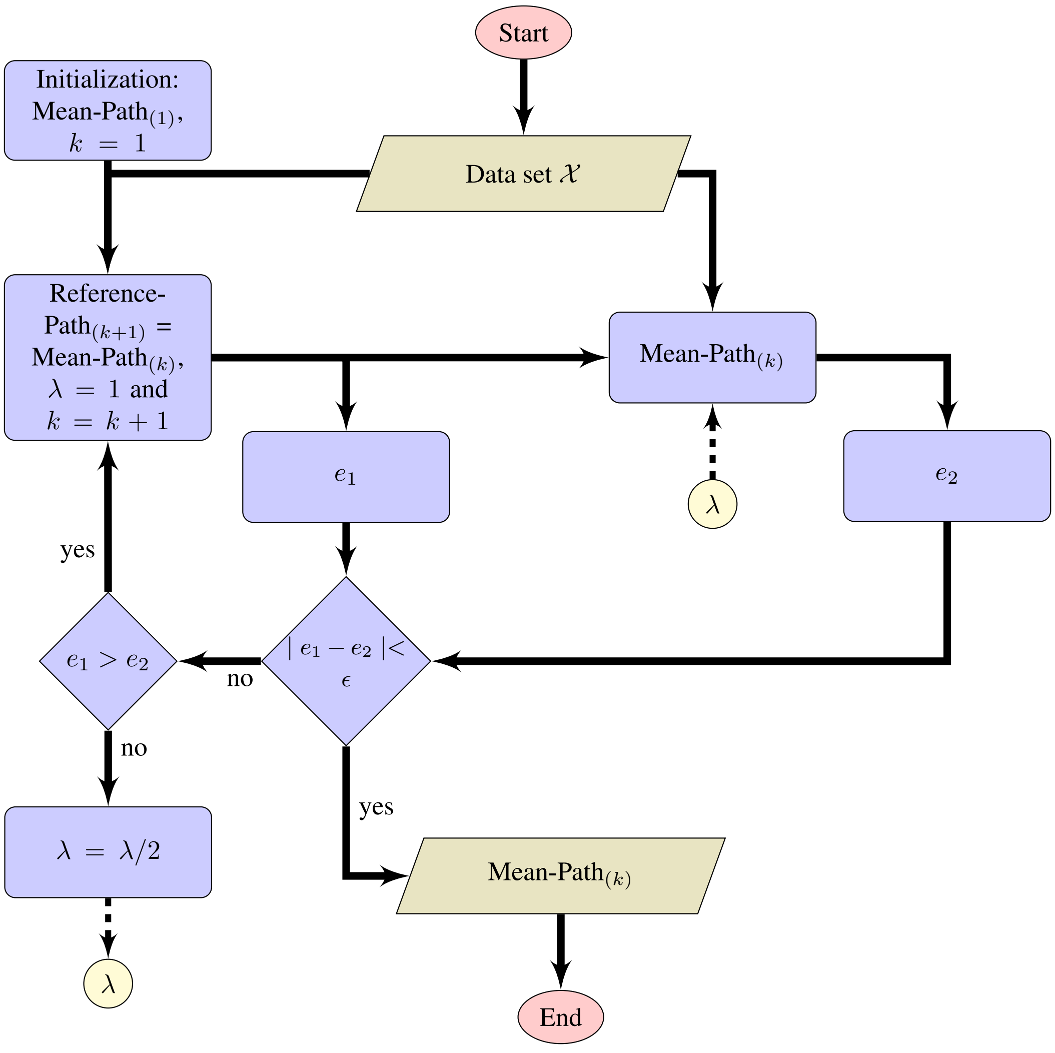

Start
Initialization: Mean-Path$_{(1)}$, $k = 1$
Data set $\mathcal{X}$
Reference-Path$_{(k+1)}$ = Mean-Path$_{(k)}$, $\lambda = 1$ and $k = k + 1$
Mean-Path$_{(k)}$
e_1
λ
e_2
yes
$e_1 > e_2$
no
$| e_1 - e_2 | < \epsilon$
no
yes
$\lambda = \lambda/2$
Mean-Path$_{(k)}$
λ
End

Figure 4.1: *This flow chart shows the Mean-Path algorithm. A set of paths of a task performances constitutes our data set (see the text for more information). The algorithm terminates when the difference between error of Reference-Path and estimated Mean-Path ($| e_1 - e_2 |$) is less than a threshold ϵ.*

the generalized path, we propose an error metric in Section 4.4.

Based on the proposed error metric, we present an algorithm for computing a generalized path, called Mean-Path, from a set of demonstrated paths. This method computes a generalized path that minimizes the proposed error metric.

A flow chart of the proposed method for computing a generalized path

Table 4.1: *Glossary*

MP	: Mean-Path, ζ^{mp}	PL	: Perpendicular-Line
RPa	: Reference-Path,	DS	: Distance-Space, $\mathcal{Z}$
RPo	: Reference-Point	OS	: Original-Space, $\mathcal{X}$
OP	: Objective-Path	Φ	: map from OS into DS
Ψ	: map from DS into OS		

from a set of paths is shown in Figure 4.1. According to this algorithm, one of the path within the set is taken as initial path for Mean-Path computation, called Reference-Path. Distances of all path from the Reference-Path are computed at each point of Reference-Path along a line normal to the Reference-Path. This distances build a new representation of paths based on the considered Reference-Path, called Distance-Space representation of paths.

An expected Mean-Path is then computed based on the corresponding Distance-Space. Every point of the expected Mean-Path is computed by finding the mean value of the distances of all paths from the point of Reference-path. The covariance matrices of Distance-Space representations of paths based on the Reference-Path and Mean-Path are computed. Then, the trace of the computed covariance matrices is taken as an evaluation metric and it is used to check the convergence of the algorithm. A regularization parameter is also used to guarantee the convergence of the algorithm (see Appendix 7.1). A sequence of the computed points is taken as a new Reference-Path and this procedure is iterated till the proposed evaluation metric is improving. This results in a path that has the minimum evaluation metric value.

To evaluate the result of the proposed Mean-Path algorithm, we use Mean-Path algorithm, GMM/GMR and nonlinear PCA, which is a method to identify and remove nonlinear correlation between data points, to compute a generalized path of different data sets, such as an artificially generated data set, the walking experiment data set and a data set of a deburring example. The results obtained by the Mean-Path algorithm, GMM/GMR and nonlinear PCA illustrate the superiority of the Mean-Path algorithm in terms of precision. In order to increase the readability of this chapter, in Table 4.1 and 4.2 we provide a glossary of abbreviations and nomenclature, respectively.

4.2 Problem definition

Consider a set of n continuous 2-D trajectories $x^j(t)$ (Fig. 4.2)

$$\boldsymbol{X} = \left\{x^1(t), x^2(t), ..., x^N(t)\right\}, \tag{4.1}$$

where t denotes the time. Although trajectories are continuous (as expressed in eq. (4.1)), they are, usually, available only through a set of collected sample points. Hereinafter, we call a sequence of sample points, which are collected while a continuous trajectory is followed, a path ζ. We assume that all the paths ($\zeta_j \in \mathcal{X}, j = 1, 2, ..., N$) are represented by M sample points,

$$\zeta_j = \left\{x_i^j \in \mathbb{R}^2, \forall\, i = 1, ..., M\right\}, \tag{4.2}$$

where $x_i^j = x^j(t_i)$, $\forall j = 1, ..., N$ and t_i denotes the time corresponding to i^{th} collected points of x^j. In this work, we use the term path for a sequence of points unless it is stated.

We assume that a set of task demonstrations is generated by a nonlinear function $\mathbf{F}(t_i)$ perturbed with noise e, as follows:

$$x_i^j = \mathbf{F}(t_i) + e$$

where $e = \mathcal{N}(\mathbf{0}_{n,1}, \sigma^2 I_{n,n})$, $\mathbf{0}_{n,1} = [0, ..., 0]^T$ and n is the number of paths in $\mathcal{X}$. We are interested in a nonlinear path, called Mean-Path ζ^{mp}, that captures major variation of a set of task demonstrations. We assume that a set of task demonstrations is available through a set of points $x_i^j \in \mathcal{X}$. Hence, similar to first principal component in standard PCA, we minimize the following error:

$$\zeta^{mp} = \underset{\mathbf{F}}{\operatorname{argmin}} \sum_{j=1}^{N} \sum_{i=1}^{M} \left(x_i^j - \mathbf{F}(t_i)\right)^2$$
$$x_i^j \in \zeta_j\ \forall\, j = 1, ..., M. \tag{4.3}$$

ζ^{mp} removes the variability of human demonstrations from data. This solution provides an invariant basis of the demonstrated path.

In order to formalize a definition of a generalized path, called Mean-Path, an *Euclidean* distance between a point on a path, called Reference-Path, and another path is introduced in the following (Fig. 4.2).

A list of symbols used in this chapter is presented in Table 4.2.

Table 4.2: *Nomenclature*

$\boldsymbol{X}$	a set of continuous paths, $\boldsymbol{X} = \{x^1(t), ..., x^N(t)\}$	$x^j(t)$	a continuous path
ζ_j	a set of sample points of $x^j(t)$, $\zeta_j = \{x^j_1, ..., x^j_M\}$	$\mathcal{X}$	$\{\zeta_1, ..., \zeta_N\}$
M	number of collected points for each path	N	number of path
q	number of dimensions of the collected data points	t	each computation step
x^j_i	i^{th} sample point of ζ_j and $x^j_i \in \mathbb{R}^q$	$\overline{\delta}^j_i$	$\overline{x^j_{i-1}, x^j_i}$
Δ_{ζ_j}	:$\{\delta^j_2, ..., \delta^j_M\}$	ζ^r	Reference-Path
x^r_i	: Reference-Point belonging to i^{th} Reference-Path (ζ^r)	L^r_i	$\perp \overline{x^r_i, x^r_{i+1}}$ at x^r_i
$p_i(*, .)$	$L^*_i \cap \Delta_. , p^{r,j}_i = p_i(\zeta_j, \zeta^r_k)$	$V^{r,j}_i$	$x^r_i - p^{r,j}_i$
$\mathcal{Z}$	Distance-Space of $\mathcal{X}$; $\mathcal{Z} = \{\zeta^d_1, ..., \zeta^d_N\}$	ζ^d_j	$\{\Phi(V^{r,j}_1), ..., \Phi(V^{r,j}_M)\}$
Φ	mapping from Original-Space into Distance-Space; $\mathcal{Z} = \Phi(\mathcal{X}, \zeta^r)$, $\zeta^d_j = \Phi(\zeta_j, \zeta^r)$	$\overrightarrow{I}_i$	innovation at x^r_i
Ψ	mapping from Distance-Space into Original-Space; $\mathcal{X} = \Psi(\mathcal{Z}, \zeta^r)$, $\zeta_j = \Psi(\zeta^d_j, \zeta^r)$	$tr(\mathbf{x})$	trace of a square matrix **x**
e^1	the first evaluation metric; $e^1 = tr(\mathcal{Z}^T\mathcal{Z})$	ζ^n	a priori known Mean-Path
e^2	the second evaluation metric, $e^2_k := \sum_{i=1}^M \left\lVert z^{mp}_i - P^N_i \right\rVert$ and $z^{mp}_i = L^N_i \cap \Delta_{\zeta^{mp}}$	ζ^{mp}	estimated Mean-Path
e^3	the third evaluation metric, $e^3_k := \sum_{i=1}^M \left\lvert \overrightarrow{I}_i \right\rvert$	x^{mp}_i	a point of ζ^{mp}

4.2.1 Distance between a Reference-Point and another path

Let $\zeta^r = \{x^r_1, ..., x^r_M\}$ be a 2-D path taken as a reference for distance computation, called Reference-Path, and x^r_is be the Reference-Points. A set of line segments, denoted by Δ_{ζ_j}, connecting two consecutive points of a path, are defined as follows:

$$\begin{aligned} \Delta_{\zeta_j} &= \left\{\overline{\delta}^j_1, \overline{\delta}^j_2, ..., \overline{\delta}^j_{M-1}\right\}, \\ \overline{\delta}^j_i &= \overline{x^j_{i-1} x^j_i}, \end{aligned} \tag{4.4}$$

where $\overline{\delta}^j_i$ is a line segment connecting an initial point x^j_{i-1} and a terminal point x^j_i. As the information of a continuous path $x^j(t)$ may not be always available, we assume that $x^j(t)$ is piecewise linear. Hence, $x^j(t)$ is approximated by some line segments connecting consecutive collected points, Δ_{ζ_j}, wherever information of the corresponding continuous path is needed. For example, an orthogonal subspace to a Reference-Path at point x^r_i is approximated by a subspace orthogonal to each line segment of the

Reference-Path, $\overline{\delta}_i^r$, which is denoted by $\overleftrightarrow{L_i^r}$[1] $\in \mathbb{R}^{q-1}$. In the 2-D case, this subspace is a line, called Perpendicular-Line, where the inner product between $\overline{\delta}_i^r$ and the unit vector of L_i^r is zero (see Fig. 4.2), i.e. $L_i^r \perp \overline{\delta}_i^r$.

A similarity measure between a Reference-Path and another path in the set, denoted by $V_i^{r,j}$, at each Reference-Point, $x_i^r \; \forall \, i = 1, ..., M$, is a vector whose absolute value is a distance between x_i^r and Δ_{ζ_j}, $\forall \, j = 1, ..., N$, as follows:

$$\begin{gathered} V_i^{r,j} := \overrightarrow{x_i^r p_i^{r,j}}, \\ \left| V_i^{r,j} \right| = \left\| \, x_i^r - p_i^{r,j} \right\|, \end{gathered} \tag{4.5}$$

where $p_i^{r,j}$ is an intersection of Δ_{ζ_j} and L_i^r (see Fig. 4.2),

$$p_i^{r,j} = \Delta_{\zeta_j} \cap L_i^r. \tag{4.6}$$

Intuitively, a local observer, moving along the Reference-Path, measures a distance from its location to Δ_{ζ_j} at each x_i^r along L_i^r. In the case of more than one intersection of L_i^r and Δ_{ζ_j}, $V_i^{r,j}$ corresponding to the minimum distance is chosen.

It is assumed that for a set of paths $\mathcal{X}$ and a considered ζ^r there always exists an intersection of L_i^r and each Δ_{ζ_j}, otherwise distance between Δ_{ζ^r} and Δ_{ζ_j} along L_i^r is assigned to be infinite.

In Fig. 4.2, a set of continuous paths, $\boldsymbol{X} = \{x^1(t), x^2(t), x^3(t)\}$, their collected data points, $\mathcal{X} = \{\zeta_1, \zeta_2, \zeta_3\}$, and the proposed similarity measure, $V_i^{r,2}$, for a point of the Reference-Path are shown.

It is worth mentioning that the proposed definition of similarity measure between Δ_{ζ^r} and Δ_{ζ_j} at each point of Δ_{ζ^r} is not symmetric. This definition first determines a point on Δ_{ζ_j}s, namely $p_i^{r,j}$, that corresponds to a point of ζ^r, namely x_i^r. Then, the distance between these two points is taken as a similarity measure, computed by eq. 7.1. However, if we take Δ_{ζ_j} as a Reference-Path, x_i^r does not necessarily correspond with $p_i^{r,j}$ because the subspaces orthogonal to Δ_{ζ^r} and Δ_{ζ_j} are not necessarily the same.

The similarity measure here presented can be easily extended to higher dimensions by considering an intersection between a path and a subspace orthogonal to a reference path.

In the following section, a new representation of the paths, that is called Distance-Space, and its use for the Mean-Path computation algorithm is presented.

[1]For the sake of simplicity we use L_i^* instead of $\overleftrightarrow{L_i^*}$.

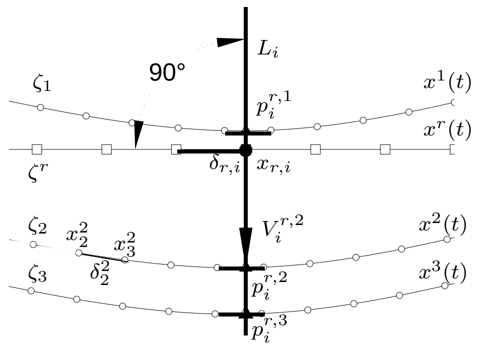


Figure 4.2: *A set of four 2-D curves (continuous ones $x^i,\ i = 1, 2, 3$, and their collected data points ζ, shown by circle signs), Reference-Path x^r and its data points ζ^r shown with square signs. The line segment $\delta_2^2 = \overline{x_2^2, x_3^2}$ linearly approximates the continuous path x^2 between x_2^2 and x_3^2. The line L_i is perpendicular to $\delta_{r,i}$. Intersections of L_i and each Δ_{ζ_j}, denoted by $p_i^{r,j}, \forall\, j = 1, 2, 3$ (marked with triangle sings). Vector $V_i^{r,2}$ determines the position of $p_i^{r,2}$ relative to x_i^r.*

4.3 The Distance-Space

In this section, the similarity measure in eq. (4.5) is used to represent a path relative to a Reference-Path, in such a way that the information concerning the relative position of the considered path with respect to the Reference-Path is kept invariant.

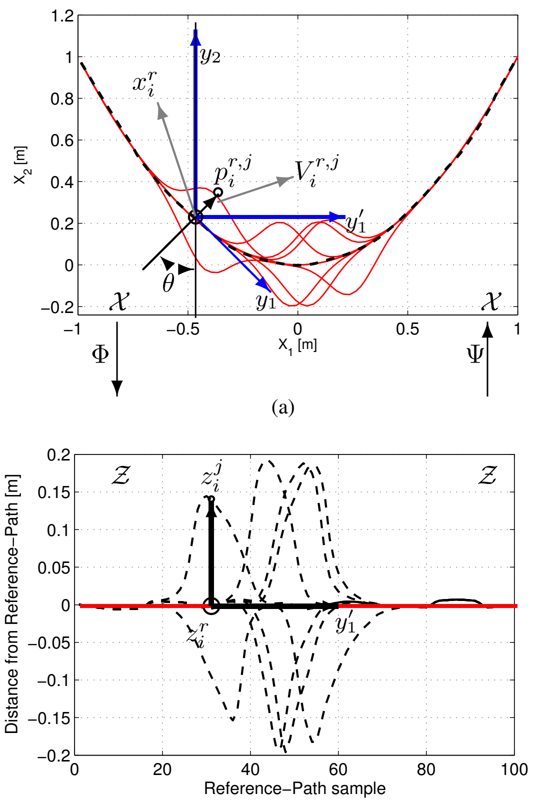
A new representation of all the paths, $\mathcal{X}$, is then obtained by computing the similarity measure proposed in eq. 7.1, based on a chosen Reference-Path, and expressing the paths in a common coordinate frame, called Distance-Space coordinate frame, as follows:

$$
\begin{aligned}
\begin{pmatrix} z_i^j \\ 0 \end{pmatrix} &= R_i^j V_i^{r,j},\ i = 1, ..., M \\
\zeta_j^d &= \{z_1^j, ..., z_i^M\} \\
\mathcal{Z} &= \{\zeta_1^d, ..., \zeta_N^d\}
\end{aligned}
\tag{4.7}
$$

where $\mathcal{Z} \in \mathbb{R}^{N\times M}$, $z_i^j \in \mathbb{R}^1$ and $R_i^j \in \mathbb{R}^{2\times 2},\ i = 1, ..., M$ are rotation matrices

$$
\begin{pmatrix} z_i^j \\ 0 \end{pmatrix} = \begin{pmatrix} cos(\theta_i^j) & -sin(\theta_i^j) \\ sin(\theta_i^j) & cos(\theta_i^j) \end{pmatrix} V_i^{r,j},\ \forall i = 1, ..., M,\ \forall j = 1, ..., N.
$$

These rotation matrices align a local coordinate system fixed at every reference point, e.g. y_1 in Fig. 4.3(a), with the coordinate system of Distance-



(b)

Figure 4.3: *A point $p_i^{r,j}$ in original data space $\mathcal{X}$ is transformed into a point in the Distance-Space $\mathcal{Z}$ based on the considered Reference-Path (black dashed line) using Φ, and the corresponding point in the Distance-Space, namely z_i^j, is transformed back to original data space using inverse mapping Ψ. The Reference-Points, x_i^rs, and $p_i^{r,j}$s are marked with bold dots in both spaces: (a) A set of 2-D paths (red lines), a chosen Reference-Path ζ^r (black dashed line), computed similarity metric $d_{j,i}$, and the local coordinate frame y_1 fixed at a Reference-Point tangent to $\delta_{r,i}$ (blue arrow); (b) the corresponding paths in Distance-Space, $\mathcal{Z}$, and the corresponding local coordinate frame y_1 at the corresponding Reference-Point (blue arrow). For better view, continuous representation of points are shown.*

Space, e.g. y_1' in Fig. 4.3(b).

The vectors computed by eq. (4.7) constitute a bijective mapping, namely Φ[2], from points of the data set in the Original-Space into a set of points in

[2]Notice that the mapping is invertible since it is bijective.

the Distance-Space (Fig. 4.3), as follows:

$$\Phi(\mathcal{X}, \zeta_r) : \mathcal{X} \xrightarrow{L_i^r \perp \zeta^r \; at \; x_i^r} \mathcal{Z},$$

where, $\mathcal{X} \in \mathbb{R}^{n \times m \times q}$ and $\mathcal{Z} \in \mathbb{R}^{n \times m \times q-1}$. Consequently, an inverse mapping, denoted by Ψ, from the Distance-Space, shown in Fig. 4.3(b), to the Original-Space, shown in Fig. 4.3(a), is also defined using the same Reference-Path, which is a reference for mapping.
Based on a Reference-Path, ζ^r, a map from original data space into Distance-Space, Φ, and its inverse, Ψ, are symmetric resulting in a one-by-one correspondence of each pair of points in $\mathcal{X}$ and $\mathcal{Z}$ (see Fig. 4.3). Φ transforms ζ^r into the horizontal axis of Distance-Space coordinate system (see Fig. 4.3(b)), and Ψ transforms the horizontal axis of Distance-Space coordinate system into ζ^r (see Fig. 4.3(a)). Hence, we express the map from the Original-Space into the Distance-Space in eq. 4.8 and vice versa in eq. 4.9 by addition and subtraction of the Reference-Path with and from the corresponding data set, as follows:

$$\begin{aligned} z_i^j &= R_i^j (p_i^{r,j} - x_i^r) \\ \mathcal{Z} &= \Phi(\mathcal{X}, \zeta^r) \end{aligned} \tag{4.8}$$

and

$$\begin{aligned} p_i^{r,j} &= \left(R_i^j\right)^{-1} z_i^j + x_i^r \\ \mathcal{X} &= \Psi(\mathcal{Z}, \zeta^r), \end{aligned} \tag{4.9}$$

where $p_i^{r,j} \in \Delta_{\zeta_j}$, Δ_{ζ_j} is obtained from $\zeta_j \in \mathcal{X}$ (see eq. (7.1)) and $z_i^j \in \mathcal{Z}$. Φ and Ψ maps the data set from Original-Sapce into Distance-Space , in accordance with eq. (4.7).

4.4 Mean-Path

In this section, we use the Distance-Space representation of a set of paths, $\mathcal{X}$, to define a generalized path, called Mean-Path.

First of all, we introduce some definitions.

We call residual of the data set, based on a Reference-Path ζ^r, the variations of the paths ζ_j, $j = 1, ..., M$, with respect to ζ^r. The residual can be obtained by computing the trace of the data covariance in the Distance-Space based on ζ^r, i.e. $tr(\mathcal{Z}^T \mathcal{Z})$, where $\mathcal{Z}$ is computed using eq. (4.8).

In this chapter, a smooth nonlinear curve that captures the maximum variation of the data set, $\mathcal{X}$, is called Mean-Path of the data set. The Mean-

Path is denoted by ζ^{mp} and can be computed as follows:

$$\begin{aligned} \zeta^{mp} = \quad & \underset{\zeta^r \in \mathbb{R}^{m \times q}}{\arg\min}\, tr(\Sigma) \\ & \text{s.t.} \quad \Sigma = \mathcal{Z}^T \mathcal{Z} \\ & \qquad\quad \mathcal{Z} = \Phi(\mathcal{X}, \zeta^r) \end{aligned} \tag{4.10}$$

A distance-Space representation of the data set, based on the computed Mean-Path, corresponds to the representation of a linear data set in a principal component coordinate system, in which the first principal component is aligned with the first main axis.
This is exemplified in Fig. 4.7. In Figs. 4.7(a) and 4.7(b), a data set that lies on a nonlinear manifold and its corresponding Distance-Space, based on the computed Mean-Path, are shown. A 2-D Gaussian distribution and its corresponding representation in a principal component coordinate system are also shown in Fig. 4.7(c). Figs. 4.7(b) and 4.7(c) (bottom) illustrates the difference between the underlying assumptions of the standard principal component analysis and the Mean-Path: in standard principal component analysis it is assumed that a data set is normally distributed around the first principal component which is linear, whereas in the Mean-Path computation we assume that the data set is normally distributed around a non-linear curve.

4.4.1 Mean-Path algorithm

In this section, we introduce an iterative algorithm to compute a solution to eq. (4.10) for a set of paths $\mathcal{X}$, whose elements are supposed to be highly correlated with an unknown nonlinear path called nominal path and denoted by ζ^n. The aforementioned algorithm is mathematically formalized in Section 4.5. The algorithm results in the minimum variance of the data set, $\zeta_j \in \mathcal{X}\ \forall j = 1, ..., N$, relative to the obtained path ζ^{mp}, as per eq. (4.10).

First, an arbitrary path is taken from the data set as the Reference-Path for the first iteration k of the Mean-Path computation algorithm, e.g. $\zeta^r_k \in \mathcal{X}, k = 1$.

We consider a coordinate system fixed at each point of the Reference-Path such that its first axis is aligned with the corresponding $\overline{\delta}_{r,i}$. Hence, the data set is transformed into the Distance-Space using eq. (4.8) (Fig. 4.3) and the residual of the data set is computed using $tr(\mathcal{Z}^T \mathcal{Z})$.

Assuming that for each subspace in Distance-Space, $L_{i,k}\ \forall i = 1, ..., M$, the points of the data set, $z^j_{i,k}$, $\forall j = 1, ..., N$, are normally distributed, the expected value of $z^j_{i,k}$, $z^c_{i,k}$, is estimated by maximizing the likelihood of the corresponding sample points. The computed expected value at each subspace is taken as a candidate point of a new Reference-Path, as it minimizes the variance of a set of normally distributed data points $z^j_{i,k}$.

For each point of the Reference-Path, an innovation is defined as $\overrightarrow{I}_{i,k}$[3] $= x^r_{i,k} - z^c_{i,k}$. This operation can be interpreted as an estimation of a gradient descent direction of the objective function in eq. (4.10). The obtained innovations can be used to update the origin of the coordinate systems by $x^r_{i,k+1} = x^r_{i,k} + \lambda I_{i,k}$ where $\lambda \in [0,1]$.

The sequence of points $x^r_{i,k+1}$ generates a Reference-Path for the next iteration $\zeta^r_{k+1} = \{x^r_{1,k+1}, ..., x^r_{M,k+1}\}$. The regularization parameter λ can be used, as per eq. (4.17), to guarantee the convergence of the algorithm at each iteration.

This procedure is repeated with the new Reference-Path, ζ^r_{k+1}, until the algorithm converges to a solution. A flow chart of the algorithm shown in Figure 7.1 describes the processes of Mean-Path computation.

4.5 Formulation of the Mean-Path and of the iterative algorithm

In this section, a mathematical formulation of the algorithm introduced in Section 4.4.1 is presented. A pseudocode of the iterative algorithm is reported in Algorithm 1.

We consider a set of paths, $\boldsymbol{X}$, whose components $x^j(t)$ can be represented as the superposition of a nominal path $x^n(t)$ and a random perturbation $\eta(t)$ directed along the normal to $x^n(t)$ as follows:

$$\begin{aligned} \boldsymbol{X} = \Big\{ x^j(t) \in \mathbb{R}^q : x^j(t) = x^n(t) + \eta_j(t)\ \mathbb{1}^T \hat{L}_j \Big\} \\ j = 1, ..., N \\ N(0, \bar{\sigma}(t)) = [\eta_1(t), ..., \eta_N(t)] \end{aligned} \tag{4.11}$$

where $N(0, \bar{\sigma}(t))$ is a multivariate random generator, $\eta_j \in \mathbb{R}^{(q-1)\times 1}$, $\mathbb{1} \in \mathbb{R}^{(q-1)\times 1}$ and all its elements equals to one, and $\hat{L}_j \in \mathbb{R}^{(q-1)\times(q-1)}$ is a matrix of unit vectors of the corresponding subspace. For example, $N(0, \bar{\sigma}(t))$ for a 2-D problem $x^j \in \mathbb{R}^2$, generates n random numbers at each time t, $\eta_j(t) \in \mathbb{R}^1$, with zero mean, where the variance $\bar{\sigma}$ at every s may be different[4]. Here, we do not enforce any assumption of the correlation between

[3]For the sake of simplicity we use $I_{i,k}$ instead of $\overrightarrow{I}_{i,k}$

[4]Although we derive the formulation making reference to a set of normally distributed paths, the same argument can be applied for a set of uniform distributed paths obtaining the same results.

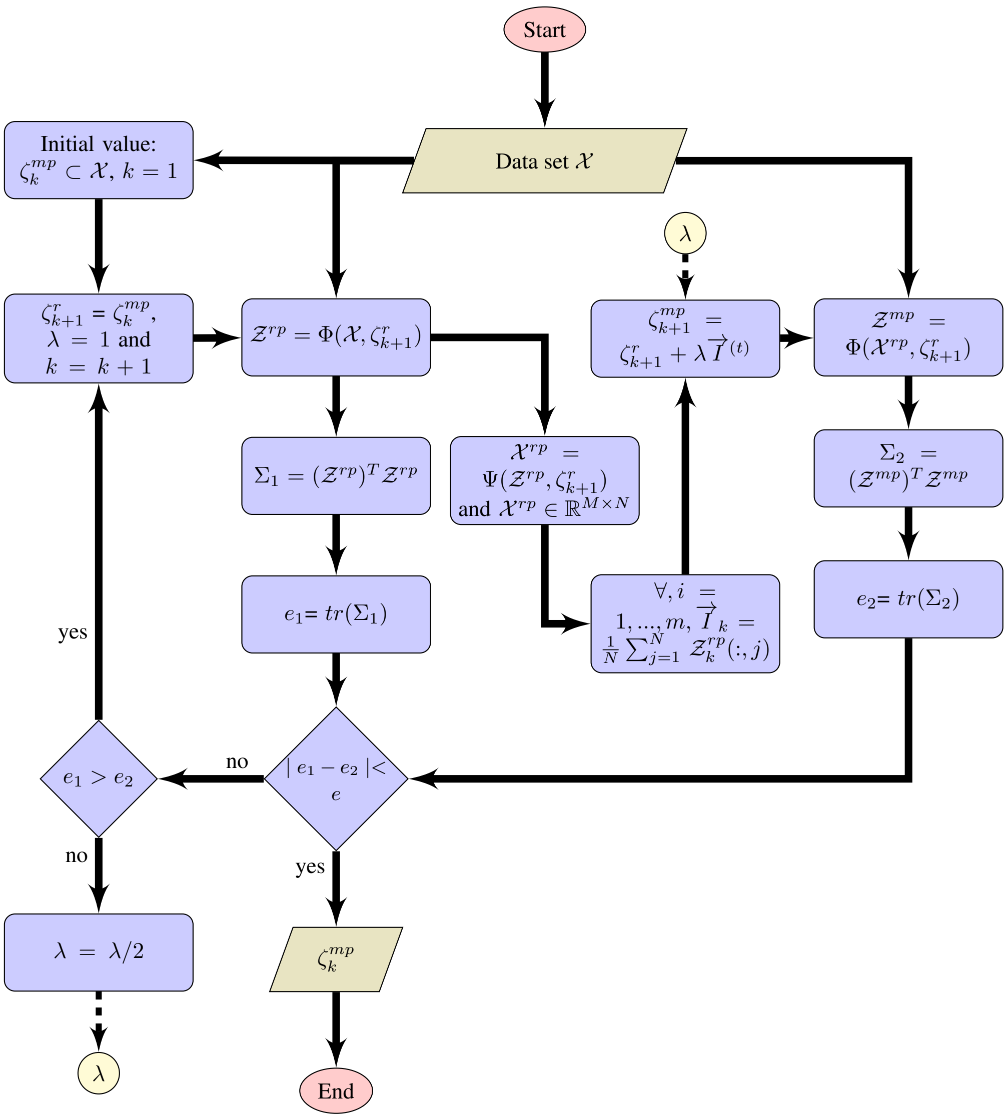

Start
Data set $\mathcal{X}$
Initial value: $\zeta_k^{mp} \subset \mathcal{X}$, $k = 1$
$\zeta_{k+1}^{r} = \zeta_k^{mp}$, $\lambda = 1$ and $k = k+1$
$\mathcal{Z}^{rp} = \Phi(\mathcal{X}, \zeta_{k+1}^{r})$
λ
$\zeta_{k+1}^{mp} = \zeta_{k+1}^{r} + \lambda \overrightarrow{I}^{(t)}$
$\mathcal{Z}^{mp} = \Phi(\mathcal{X}^{rp}, \zeta_{k+1}^{r})$
$\Sigma_1 = (\mathcal{Z}^{rp})^T \mathcal{Z}^{rp}$
$\mathcal{X}^{rp} = \Psi(\mathcal{Z}^{rp}, \zeta_{k+1}^{r})$ and $\mathcal{X}^{rp} \in \mathbb{R}^{M \times N}$
$\Sigma_2 = (\mathcal{Z}^{mp})^T \mathcal{Z}^{mp}$
$e_1 = tr(\Sigma_1)$
$\forall, i = 1, ..., m$, $\overrightarrow{I}_k = \frac{1}{N} \sum_{j=1}^{N} \mathcal{Z}_k^{rp}(:, j)$
$e_2 = tr(\Sigma_2)$
yes
$e_1 > e_2$
no
$| e_1 - e_2 | < e$
no
yes
$\lambda = \lambda/2$
ζ_k^{mp}
λ
End

Figure 4.4: *This flow chart shows the Mean-Path algorithm. A set of paths of a task performances constitutes our data set. The iterative algorithm at each computation step updates the Reference-Path along a set of lines normal to the Reference-Path.*

two consecutive points of a path. However, to ensure the convergence of the algorithm, we assume that every path, x^j, is *Lipschitz* continuous.

Each path in X is represented by a set of collected data, $\zeta_j, \forall\, j = 1, ..., M$, of points[5] $z_i^{n,j} \in \zeta_j$, $j = 1, ..., N$ that are normally distributed

[5]The superscript ^{n} denotes the values computed based on ζ^n taken as the Reference-Path.

around $x_i^n \in \zeta^n$ and along L_i^n, i.e.

$$z_i^{n,j} \sim N(x_i^n, \bar{\sigma}_i^2),\ j = 1, ..., N$$

where σ_i is the standard deviation of the path distribution along each $L_{i,k}$. As a consequence, the probability density function of these points along each $L_{i,k}$ can be written as follows:

$$f(z_i^{n,j}) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{\left(z_i^{n,j} - x_i^n\right)^2}{2\sigma_i^2}}$$

Accordingly, the likelihood function is

$$\mathcal{L}\left(\mu_i, \sigma_i^2 \middle| z_i^{n,1}, ..., z_i^{n,M}\right) = \prod_{j=1}^{N} f\left(z_i^{n,j}; \mu_i, \sigma_i^2\right).$$

In order to find these parameters, one can maximize the log-likelihood function, i.e.

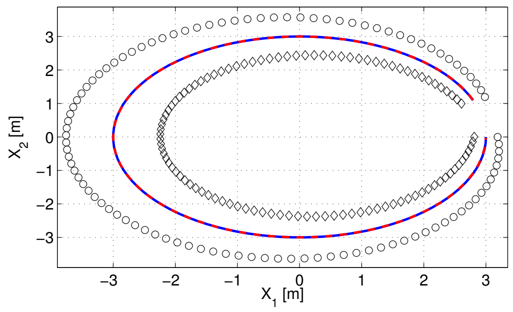
$$\ln \mathcal{L}\left(\mu_i, \sigma_i^2\right) = \sum_{j=1}^{N} \ln f\left(z_i^{n,j} | \mu_i, \sigma_i^2\right) =$$

$$-\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\sigma_i^2 - \frac{1}{2\sigma_i^2}\sum_{j=1}^{N}\left\|\left(z_i^{n,j} - \mu_i\right)\right\|^2, \tag{4.12}$$

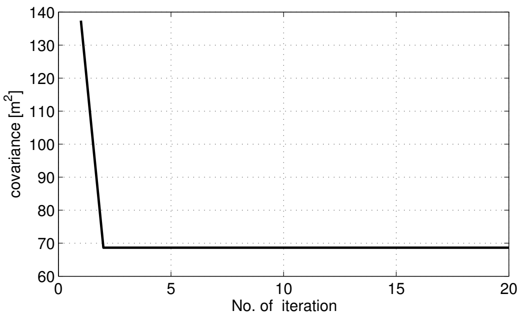
for a set of sample points, $\{z_r^{n,1}, ..., z_i^{n,M}\}$.

Consider that, if $z_i^{n,j} \in \zeta_j, \forall j = 1, ..., N$, were known, the parameters μ_i and σ_i could be found using maximum likelihood estimation, and the original problem would have become to find m mean values along every L_i^n. However, the data points collected during an experiment may not necessarily be aligned along every L_i^n. Therefore, to obtain a set of corresponding points, lines L_i^n, and thus the points of the nominal path, should be known. However, as here we assume the nominal path of a task is unknown and a set of task demonstrations, i.e. a set of paths, is the only available information, an objective function based on maximum likelihood estimation cannot be used to directly compute the Mean-Path. For this reason, we propose an iterative algorithm, inspired by the expectation-maximization method.

As this algorithm has been already introduced in Section 4.4.1, we consider here the generic iteration t in order ro formalize the computation of the innovation $I_{i,k}$.



(a)



(b)

Figure 4.5: *(a) A nominal path (blue line), two generated paths uniformly distributed about the nominal path (circle and square signs) and the computed Mean-Path (red dashed line); (b) Evolution of the residual during Mean-Path computation.*

First of all, from eq. (4.12), the maximum likelihood estimate of the mean value is

$$\hat{\mu}_i = \arg\max \ln \mathcal{L}(\mu_i, \sigma_i^2)$$
$$\hat{\mu}_i = \frac{1}{n}\sum_{j=1}^{N} z_{i,k}^j \tag{4.13}$$

Substituting $z_{i,k}^j$ from eq. (4.9) into eq. (4.13) yields

$$z_{i,k}^c := \hat{\mu}_i = x_{i,k}^r + \frac{1}{n}\sum_{j=1}^{N} z_{i,k}^j, \tag{4.14}$$

where $x_{i,k}^r \in \zeta_k^r$ and $z_{i,k}^c$ is a center point in Original-Space based on the corresponding ζ_k^r. The summation term in the right-hand side of eq. (4.14) is a center point in the corresponding Distance-Space.

Accordingly, the updating values or innovations, $I_{i,k}$, are computed as follows:

$$\begin{aligned} I_{i,k} &:= z^c_{i,k} - x^r_{i,k}, \\ x^r_{i,k+1} &= x^r_{i,k} + \lambda I_{i,k}, \end{aligned} \tag{4.15}$$

where λ is a regularization parameter that is used to guarantee the convergence as per eq. 4.17. The sequence of computed points, $x^r_{i,k+1} \in \zeta^r_{k+1}$, $i = 1, ..., M$, is taken as a Reference-Path for the next computation step.

Based on a chosen Reference-Path we can thus explicitly formalize a Mean-Path at each step. However, the path obtained by maximum likelihood estimation may not be identical to the desired Mean-Path, because the lines perpendicular to ζ^r_k and ζ^r_{k+1} are different. As a consequence, the values of the objective function in eq. (4.10) at two consecutive iterations could be different, unless the chosen Reference-Path is the desired Mean-Path. In conclusion, a Mean-Path can be iteratively computed as follows:

$$\begin{aligned} \zeta^r_{k+1} = \quad & \underset{\zeta^r_k \in \mathbb{R}^{m \times q}}{\arg\min}\; tr(\Sigma_k) \\ & \text{s.t.} \quad \Sigma_k = \mathcal{Z}^T \mathcal{Z} \\ & \qquad\;\; \mathcal{Z} = \Phi\left(\mathcal{X}, \zeta^r_k\right) \end{aligned} \tag{4.16}$$

where $\zeta^r_1 \subset \mathcal{X}$, $\zeta^{mp} = \zeta^r_{end}$. The points of ζ^r_k at each computation step, $x^r_{i,k}$, are updated using eq. (4.15). At each iteration, the residuals of the data points based on ζ^r_k and ζ^r_{k+1}, and the regularization parameter λ are used to check the convergence[6] , introduced in eq. (4.17).

$$\mathcal{Z}_{k+1} = \Phi\left(\mathcal{X}, \zeta^r_{k+1}\right),$$

$$\Sigma_{k+1} = tr\left(\mathcal{Z}^T_{k+1} \mathcal{Z}_{k+1}\right)$$

and

$$tr\left(\Sigma_{k+1}\right) < tr\left(\Sigma_k\right) \tag{4.17}$$

4.6 Experiments

In this section, we first introduce some evaluation metrics in order to evaluate the generalized path obtained by the Mean-Path algorithm. Besides, to validate the proposed approach we use the evaluation metrics to compare

[6]The proof of convergence of the proposed algorithm is available at https://sites.google.com/site/ghalamzanamir/documents or ftp://ftp.elet.polimi.it/outgoing/Amirmasoud.Ghalamzanesfahani/

Algorithm 1 Mean-Path algorithm, $\bar{\Delta}e$ is a chosen small threshold

```
 1: procedure MEAN-PATH($\mathcal{X} = \{\zeta_1, ..., \zeta_N\}$)
 2:     $k = 1$ , $\zeta_k^r \leftarrow \zeta_1$ , $\Delta e = 1 \, and \, \lambda = 1$
 3:     while $\Delta e > \bar{\Delta}e$ do
 4:         $\mathcal{Z} = \Phi(\mathcal{X}, \zeta_k^r)$
            $\Sigma_k = \mathcal{Z}^T \mathcal{Z}$,
            $e_k \leftarrow tr(\Sigma_k)$,
 5:         $z_{i,k}^d \leftarrow \mathbb{E}(z_{i,k}^j \in \mathcal{Z}, \forall j = 1, ..., N)$
            $z_{i,k}^c = \Psi(z_{i,k}^d, \zeta^r)$
            $\forall \, i = 1, ..., M$
 6:         $I_{i,k} \leftarrow \lambda(x_{i,k}^r - z_{i,k}^c) \, \forall i = 1, ..., M$
 7:         $\zeta_{k+1}^r \leftarrow \zeta_k^r + \overrightarrow{I}_k$
 8:         $\mathcal{Z} = \Phi(\mathcal{X}, \zeta_{k+1}^r)$,
            $\Sigma_{k+1} = \mathcal{Z}^T \mathcal{Z}$,
            $e_{k+1} \leftarrow tr(\Sigma_{k+1})$
 9:         if $e_{k+1} > e_k$ then
10:             $\lambda \leftarrow \frac{\lambda}{2}$
11:         else
12:             $\Delta e \leftarrow \|e_{k+1} - e_k\|$ , $k = k + 1 \, and \, \lambda \leftarrow 1$
13:         end if
14:     end while
15:     return $\zeta_k^r$
16: end procedure
```

the generalized paths obtained by the Mean-Path algorithm, nonlinear principal component and GMM/GMR [27]. We use nonlinear principal component, based on a multi-layer perceptron with an auto-associative topology [115, 114]. We present four different examples in the following. In the first example, we generate a set of uniformly distributed paths from a known nominal path. This allows us to define an evaluation metric in Section 4.6.1 with respect to a known ground truth, a known nominal path. Although the second example, which is a walking example, does not stress the high accuracy requirement, we use that to present some interesting features of the approach. In the third example, a surgical robotic task, we focus on skill assessment of a novice surgeon based on some expert noisy demonstrations. This example illustrates how the accuracy of an obtained generalized path can affect the quality of the skill assessment. Finally, in the last example, we evaluate the precision of the proposed approach, GMM/GMR and NLPCA applied to a deburring example.

4.6.1 Evaluation metrics

We start defining two metrics that can be used to evaluate the proposed algorithm. First of all, we introduce the residual of a data set based on an estimated Mean-Path as the first evaluation metric, i.e.

$$e_k^1 = tr(\Sigma_k).$$

When the ground truth is known, we can define an evaluation metric as the sum of the absolute values of the distances between the computed path, $\zeta^r_{i,k+1}$, and the known nominal path, ζ^n, as follows:

$$e_k^2 := \sum_{i=1}^{M} \|z_i^{mp} - x_i^{np}\|,$$

$$z_i^{mp} = L_i^{np} \cap \Delta_{\zeta^r_{i,k+1}},$$

where $L_i^{np} \perp \Delta_{\zeta^n}$, P_i^{np} is the i^{th} sample point of ζ^n, M is the number of nominal profile sample points, Δ_{ζ^n} and $\Delta_{\zeta^r_k}$ are computed by using eq. 4.4. Finally, another evaluation metric is given by

$$e_k^3 := \sum_{i=1}^{M} |I_{i,k}| = \sum_{i=1}^{M} \left\| \sum_{j=1}^{N} z_{i,k}^j \right\|.$$

Although, e_k^2 provides the precision of an estimated Mean-Path with respect to a ground-truth, it is only computable if the corresponding nominal path is known a priori. On the other hand, e_k^1 and e_k^3 can be always computed even if the corresponding nominal profile is unknown. In particular, e_k^3 is proposed for Mean-Path computation since it measures how large the updating values are at each computational step of the Mean-Path algorithm.

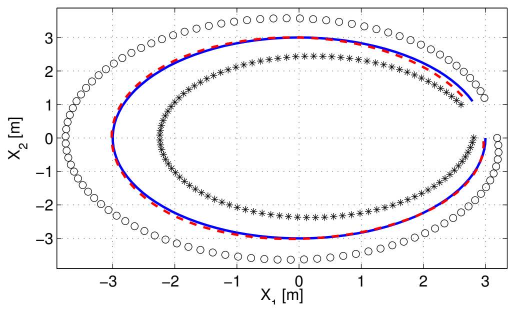
Furthermore, the minimum value of e_k^1 represents the minimum variation of the corresponding data points, i.e. the computed point of the Reference-Path corresponds to the expected value. As a consequence, the corresponding innovations are zeroimplying that e_k^1 and e_k^3 are correlated.

4.6.2 Artificially generated data set

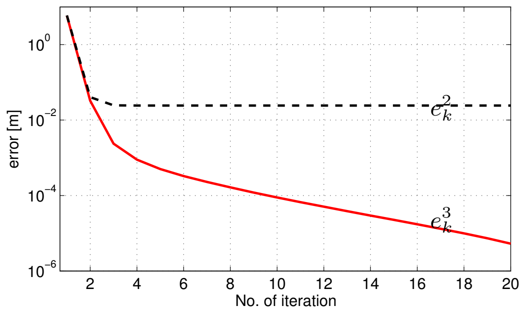
A first validation of the algorithm has been performed with a set of artificially generated data.

Two quasi-circular paths have been generated with uniform distances from a considered quasi-circular nominal path (Fig. 4.5(a)).

In this case, the Mean-Path algorithm converges after just few iterations to a Mean-Path, a good estimation of the nonlinear nominal profile (Fig.



(a)



(b)

Figure 4.6: *(a) A computed nonlinear principal component (red dashed line) for the same data set, shown in Fig. 4.5(a); (b) Evolution of e_k^3 (red line) and of e_k^2 (black dashed line) at each computation step during Mean-Path computation of the data set shown in Fig. 4.5(a).*

4.5). The computation of the Mean-Path was continued even after convergence to show the stability of the algorithm (Fig. 4.5(b)).

The evolution of e_k^1, e_k^2 and e_k^3 during Mean-Path computation are shown in Fig. 4.5(b) and 4.6(b). As it is shown in Fig. 4.5(b) and 4.6(b) e_k^1, converges to a minimum value during Mean-Path computation whereas e_k^2 and e_k^3 is going to zero.
Furthermore, GMM/GMR with 90 Gaussian components is used to compute a generalized profile from the same data set.
We also use NLPCA, a neural network model to compute a principal curve of the same data set (Fig. 4.6). This method computes a nonlinear principal component of the data set based on a multi-layer perceptron with an auto-associative topology [115]. According to the topology of the corresponding data set, the corresponding parameters, determining the structure of this neural network model, can be set. Here, we only report the results of the NLPCA model type with the best evaluation value to compute a non-

Table 4.3: *Artificially generated data set, e_1 $[m^2]$ of the obtained path by different approaches.*

	$e_1[m^2]$	$e^2[m]$
GMM/GMR	138.698	3.811
NLPCA	76.638	2.928
Mean-Path	68.653	0.3

linear principal component of the data sets as shown in Fig. 4.6(a).

It is worth mentioning that the evaluation values sum over all $M = 100$ sample points. The evaluation values for the nonlinear curves obtained by the Mean-Path algorithm, GMM/GMR, and NLPCA are reported in Table. 4.3.

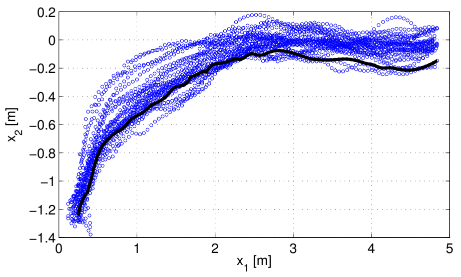
As it is shown in this table, e_1 of the path obtained by Mean-Path algorithm is less than the ones of the path obtained by GMM/GMR and NLPCA. e_1 represents the variation of the data set with respect to the path obtained by different methods, hence its value depends on the distribution of the data set with respect to each obtained path. Since the ground truth in this example is known, we can also compute e^2. This metric shows how close are the ground truth and the computed paths by different methods. Comparing e^2 of the path obtained by the Mean-Path algorithm, GMM/GMR, and NLPCA shows that Mean-Path algorithm is able to capture the ground truth with higher precision.

4.6.3 Experimentally collected data set

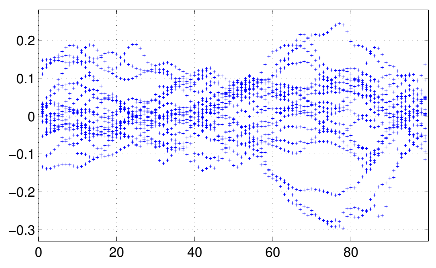
A second example considers a se t of experimentally collected human walking trajectories (Fig. 4.7(a)). Five volunteers were asked to walk from a point within circle A, in a corridor, and to enter a room and reach a point within a circle B. Each volunteer repeated the task six times, giving rise to a set of 30 paths.

The rational explanation for using the proposed method in the context of robot programming by demonstration is as follows: we assume to perform a task, e.g. moving from A to B, an individual tries to follow a reference path maximizing the squared distances to some task constraints, , e.g. the walls of the corridor, while reaching a target point.

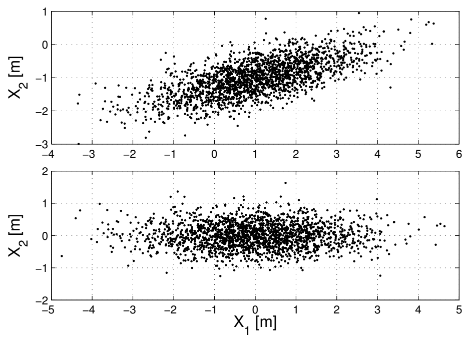
Each execution of the task is disturbed by some perturbations resulting in deviations from the reference path (Fig. 4.7). We consider that the constraints are unknown and the demonstrated paths are the only available information. In fact, a set of demonstrated paths includes information about the imposed constraints in the corresponding environment during task execution. We are interested to compute a noise free Reference-Path that



(a)



(b)



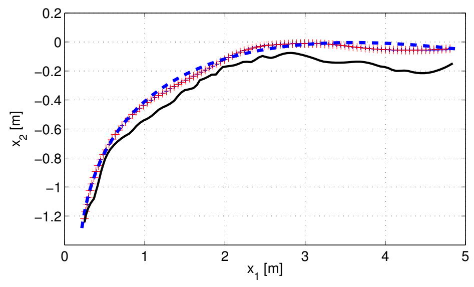
(c)

Figure 4.7: *(a) Walking experiment data set, start and goal circle marked with A and B, respectively. The constraints of the corresponding environment are represented with thick lines; (b) Distance-Space of the data setbased on the computed Mean-Path. (c) a set of 2-D data points (above) and the corresponding representation by aligning the first principal component of the data points with the first main axis (bottom).*

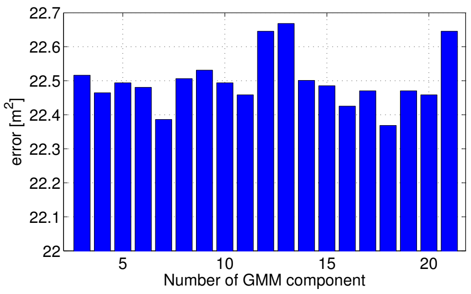
allows a mobile robot to smoothly move from A to B. A generalized path obtained by the Mean-Path algorithm with the minimum squared distances from all the demonstrated paths results in the maximum square distances from the task constraints if the demonstrations are some paths with maximum squared distances from the constraints disturbed by some perturbations.

The path obtained by the Mean-Path algorithm from the demonstrated paths along with the corresponding Reference-Path and the generalized path obtained by NLPCA are shown in Fig. 4.8(a).

Furthermore, the corresponding Distance-Space based on the obtained Mean-Path, shown in Fig.4.7(b). We also present a 2-D normally distributed data set (Fig. 4.7(c) (top)) to show the similarity between Distance-Space representation (Fig.4.7(b)) and corresponding representation of the 2-D data set in its principal component coordinate system (Fig. 4.7(c) (bot-

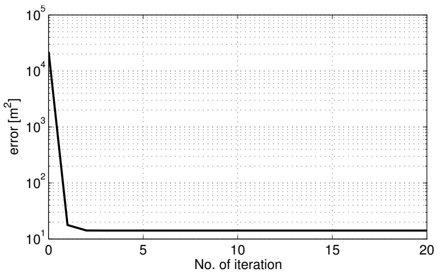


(a)

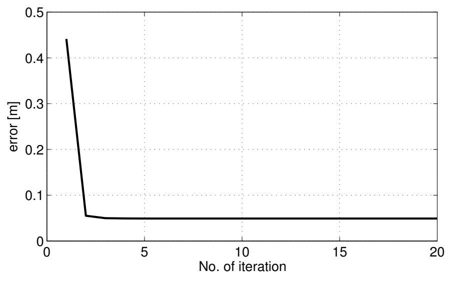


(b)

Figure 4.8: *Walking experiment data set: (a) computed nonlinear principal component (blue dashed line), Mean-Path (marked with red crosses) and the corresponding Reference-Path (black line) used by the Mean-Path algorithm; (b) e_1 of the computed path by GMM/GMR with increasing number of Gaussian components.*



(a)



(b)

Figure 4.9: *Walking experiment data set: (a) evolution of e_k^1 during Mean-Path computation; (b) evolution of e_k^3 during Mean-Path computation.*

tom)).

We computed the Mean-Path (Fig. 4.8(a)) of the data set. The algorithm converges to a solution with data residual, e_1, equal to 13.886 $[m^2]$ with $+/-0.1$ deviation for different initial Reference-Paths. The evolution of

Table 4.4: *Walking data set, e_1 $[m^2]$ of the obtained path by the Mean-Path algorithm, GMM/GMR, and NLPCA*

Mean-Path		13.886
GMM/GMR		22.364
NLPCA - Neural Network	C-B-S	NA
	C-B-H	15.372
	C-I-S	15.089
	C-I-H	NA
	NC-B-S	14.635
	NC-B-H	99.630
	NC-I-S	14.690
	NC-I-H	14.490

the e_k^1 and e_k^3 during a Mean-Path computation are shown in Fig. 4.9. In Fig. 4.8(b), e_1s of the obtained paths by GMM/GMR with different number of Gaussian components are shown. The values of the data residual, namely e_k^1, yielded by the Mean-Path algorithm, GMM/GMR, and NLPCA are reported in Table 4.4[7]. The comparison of e_1 corresponding to the path obtained by the Mean-Path algorithm, GMM/GMR, and NLPCA shows that our proposed algorithm results in a path with higher precision.

4.6.4 A surgical robotic task

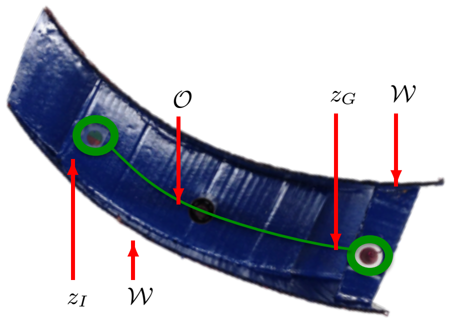
As a third example, we consider the task of picking and placing an object during surgeon training, subjected to spatial constraints that represent the small available space within a patient's body.

To obtain a constrained environment we designed the structure shown in Fig. 4.10(a), that is characterized by two walls constraining the movement of the robot tool during picking the object from z_I and placing at z_G.

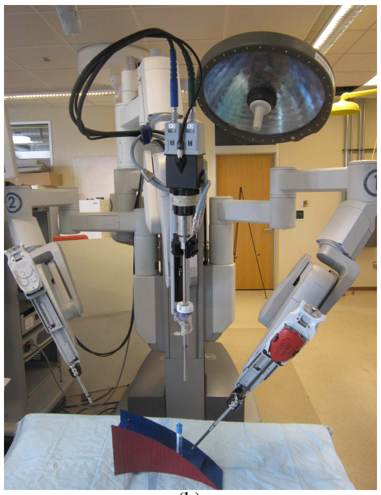
In this experiment a da Vinci robot (Fig. 4.10) was used to collect a set of demonstrations of the aforementioned task, asking the operator to maximize the distances from both walls.As a result we collected a data set $\mathcal{X}$ composed of four paths (Fig. 4.11(a)). As shown in the figure, human demonstrations are inherently noisy. Our goal is to obtain a generalized path from these noisy demonstrations that can be later used to assess the skill of a novice performing the same task [107]. In this regard, e^2, the sum of the distances between the obtained generalized path and the path followed by the operator can be used as an error metric to evaluate the novice performance.

A generalized path is computed using the Mean-Path algorithm, GM-

[7]The abbreviations used in Table 4.4 and 6.1, are the modeling parameters determining the structure of the used neural network, as follows: Circular (C), Non-Circular (NC), Inverse (I), and Bottleneck (B). For further explanation about these parameters and the corresponding network structure see [115].



(a)



(b)

Figure 4.10: *(a) The task model used for data collection with da Vinci surgical robot with a single obstacle (marker), the nominal path that expert follows in the absence of obstacles (green line), initial point z_I, goal point z_G and the walls of the structure $\mathcal{W}$ constraining the movement of the robot's arm; (b) da Vinci set up to collect expert demonstrations.*

M/GMR and NLPCA and the corresponding results are reported in the following. The paths in the Distance-Space based on the computed Mean-Path and evolution of e_1 and e^3 during Mean-Path computation are shown in Fig. 4.11(b) and Fig. 4.12(a), respectively. Furthermore, e_1s of the computed generalized path by GMM/GMR with different numbers of Gaussian components are shown in Fig. 4.12(b). As shown in Fig. 4.12(b), GMM/GMR with thirteen Gaussian components results in the minimum e_1. However, the path obtained by Mean-Path algorithm results in an e_1 three

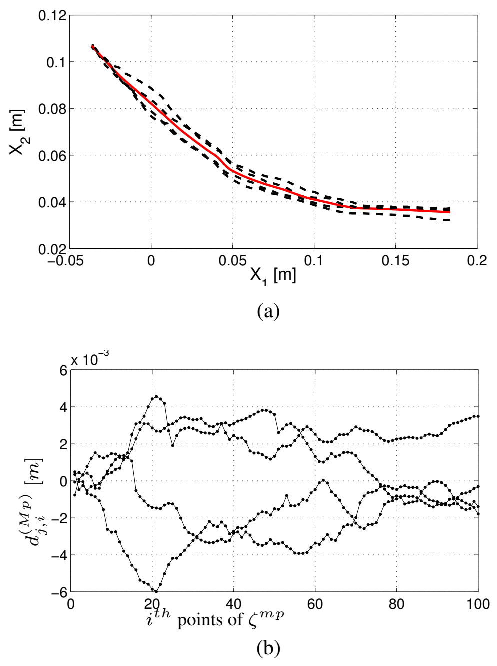
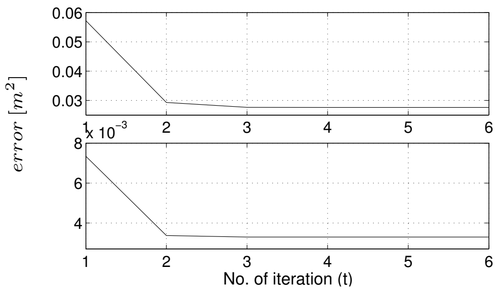


Figure 4.11: *(a) The data set collected with da Vinci robot (black dashed line) and the generalized path computed by Mean-Path (red solid line); (b) Distance-Space representation of the data set based on the computed Mean-Path.*

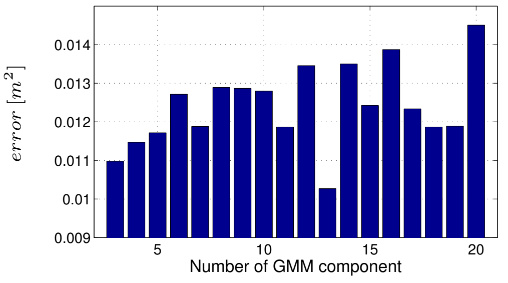
times smaller than the one obtained by GMM/GMR with the best selection of the number of Gaussian components (Fig. 4.12(a)). The use of NLPCA results in $e_1 = 0.0145\ [m^2]$, which is a bigger error value than both the one obtained by the Mean-Path algorithm and GMM/GMR.

4.7 Conclusion

In this chapter, we discussed the mimicking component of robot LfD. We propose a method to estimate the nominal profile of a task given a set of noisy human demonstrations as mimicking component of robot LfD. In many cases, human demonstrations are suboptimal and noisy solutions to the problem of performing a task. An iterative algorithm is therefore presented to estimate a noise free geometrical model of human performances. This model is highly demanded where a model of the workpiece is not available especially in small and medium size companies. Two examples with synthetic and experimental data are used to compare the accuracy of the proposed method with different methods, namely GMM/GMR and non-



(a)



(b)

Figure 4.12: *da Vinci data set: (a) evolution of e^3 (top) and e_1 (bottom) during Mean-Path computation; (b) e_1 of the generalized path obtained by GMM/GMR with different numbers of Gaussian component.*

linear PCA. Moreover a real-world problem of deburring was discussed where the resulted profiles of the Mean-Path, nonlinear principal component analysis and Gaussian mixture model/Gaussian mixture regression are compared. The comparisons between different methods illustrates that the accuracy of the nonlinear principal component analysis and the Gaussian mixture model/Gaussian mixture regression depends on the nominal profile model. For example, the nonlinear principal component analysis and the Gaussian mixture model/Gaussian mixture regression both result in higher accuracy in linear profile than circular profile. In contrast, the proposed method maintain good accuracy for different nominal profiles and results in higher precision in estimating the nominal profile.

CHAPTER 5

Robot LfD: from imitation to emulation

5.1 Outline of the chapter

Robots has been programmed to successfully perform a task in factories with a known structured environment in which the condition is controlled; however, they are still far from an autonomous agent performing different tasks in an unstructured environment. Robot learning from demonstration has shown a potential capacity to provide robots with more autonomy. In this regard, we present an approach of robot learning from demonstration that enables a robot to learn how to adapt the learned skill to an environment with varying configuration of obstacles with different object classes.

We assume here that a skill can be represented by a nominal model, discussed in the previous chapters, where a robot can use it to perform the corresponding task in a known structured environment. Nonetheless, the robot must react to some stimuli existing in an unstructured environment during task execution, e.g. an existing obstacle. We further assume that a specific reaction to a stimulus itself is a part of the skill. For example, a person may keep very far from a pet during manipulating an object while not very far from an object. Another motivation to learn a specific reaction to a stimulus is that a robot that can execute a task like humans may allow people to trust it as a coworker since its movement is more natural. This may thus ease human-robot interaction.

We aim at providing a robot with the ability to learn from human demonstrations how to react to a stimulus. The proposed approach in this chapter allows a robot to obtain a control policy from human demonstrations to perform a task. The obtained policy combines the nominal model of the skill, obtained by the method presented in the previous chapters, with an adaption functionality to a new environment with different configuration of obstacles.

5.2 Motivating example

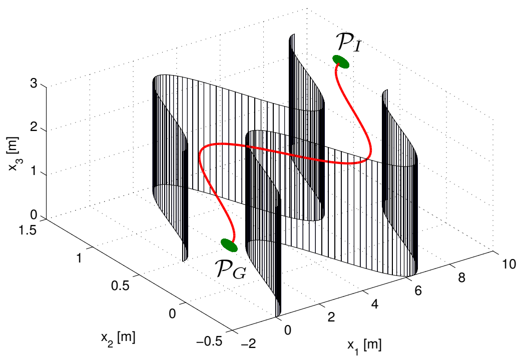
Assume that a person may wish to teach a household service robot how to set a dinner table. He/she may walk the robot and teach it how to bring the dishes from the kitchen and put them on the table. A simulation of a task is shown in Fig. 5.1.
Inspired by the observational studies, presented in Chapter 2, and corresponding to the types of observational learning, we propose an approach (Fig. 5.2) of robot learning from demonstration that

1. computes a generalized path (mimicking);
2. adapts the obtained path to a new environment (emulation);
3. scales the generalized path to a new goal point (imitation).

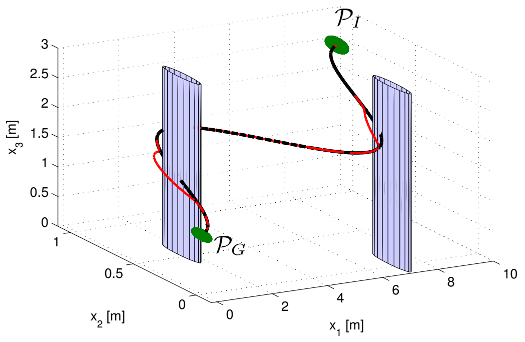
First, when *mimicking* a skill, we assume that a set of suboptimal and noisy task demonstrations are available to a robot from which the robot must compute a noise-free generalized path using GMM/GMR [22]. The robot can replicate the task using the generalized path if the environment is fixed and without any obstacle. This method alone however cannot generalize to different environments.

Second, at the *emulation* learning level, based on the computed generalized path we use an Inverse Optimal Control (IOC) approach to compute a reward function whose optimal solution is as close as possible to the demonstrations (see Section 5.3). The IOC problem recovers a reward function from a set of demonstrations in terms of a set of environment features [1]. A further literature review of recent works on inverse optimal control was presented in Chapter 1. This lets us learn how the robot should deviate from the nominal path in response to different obstacles.

Third, at the *imitation* learning level, a robot must be able to generalize a demonstration to a new goal point. In the proposed method, we use DMP to scale the generalized path obtained by GMM/GMR to a new goal point [56], resulting in what we refer to as a *nominal path*: an estimate of the underlying noise- and obstacle-free trajectory. The obtained nominal path along with the obtained parameters at the emulation learning level are used to build the reward function. The obtained reward function is then used to generate a necessary path to perform the task in a situation with a new distribution of obstacles and with a new goal point.



(a)



(b)

Figure 5.1: *(a) Mimicking component: picking an object (green sphere) from $\mathcal{P}_I$ and placing it at $\mathcal{P}_G$ in a known structured environment, and the corresponding optimal solution (red line), the structure of the environment imposes constraint on robot movement by the walls; (b) emulation component: an unknown unstructured environment (the blue bar). The position of each bar may change from one execution to another. The optimal solution to the structured environment (black dashed line) and the optimal solution to the new environment with the obstacles (red line).*

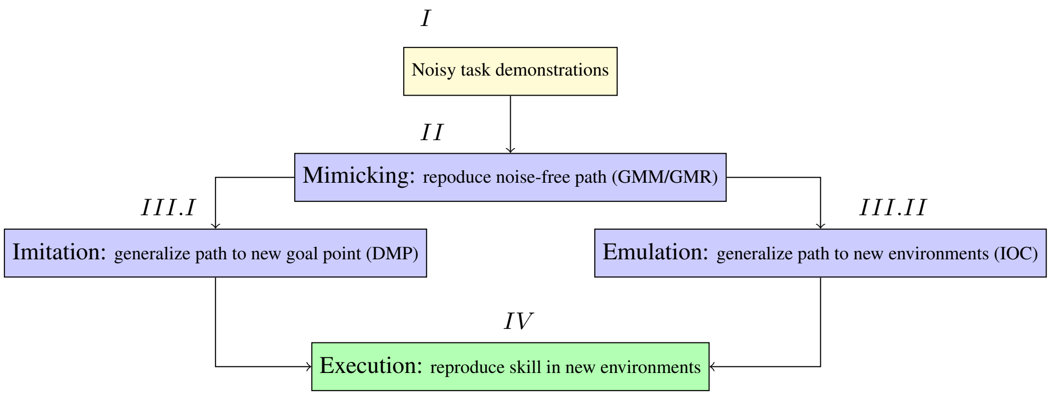

I
Noisy task demonstrations
II
Mimicking: repoduce noise-free path (GMM/GMR)
$III.I$
Imitation: generalize path to new goal point (DMP)
$III.II$
Emulation: generalize path to new environments (IOC)
IV
Execution: reproduce skill in new environments

Figure 5.2: *Overview of the different steps in the proposed method. At the first level I, a set of noisy task demonstrations are provided. At level II, a noise free generalized path is computed. Then, at level $III.I$ and $III.II$, a path is computed adapted to new goal point as well as to a new environment. Finally, the robot performs the task at level IV by using the previously obtained path.*

In contrast to our approach, prior works explicitly planned for obstacle avoidance and combined it with DMP. Those approaches neither allow a non-expert user to modify the behavior of the robot nor the robot reaction to a stimulus might look human-like.

Eventually, with this strategy, a robot incorporates the noise-free skill and the desired user response to different environmental features, both obtained from human demonstrations, into a single reward function which can be used to generalize the task to new goal points and environments. This approach allows a person to teach a robot to keep far from the cat and not very far from the chair. This is the most natural way for a non-expert user to determine the robot's behavior by showing the robot how to respond to different object classes.

5.3 Reward Function Formulation

In order to integrate different types of robot LfD into a single model of a demonstrated task, we formalize the learning from demonstration problem proposed above as an optimal control problem.

5.3.1 Optimal Control Formulation

We assume a set of task demonstrations $\mathcal{D}$ is collected (Fig. 5.3) such that each task demonstration $\zeta_d \in \mathcal{D}$, $\forall d = 1, ..., D$, is an optimal solution to an unknown reward function in the corresponding environment corrupted by noise, $\mathbf{Sc}_d = \{\mathcal{O}_1, ..., \mathcal{O}_L\}$, $\forall\ d = 1, ..., D$, where D is the number

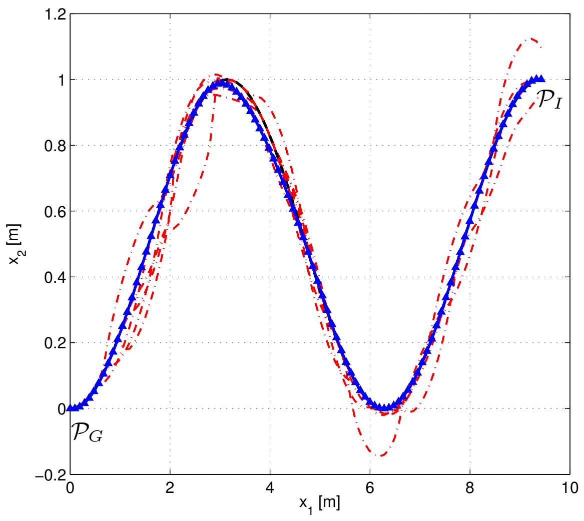


Figure 5.3: *Spatial part of a set of task demonstrations (red dashed-dotted lines) nominal path (black dashed line) and estimated nominal path (blue line marked with triangles).*

of demonstrations and L is the number of obstacles $\mathcal{O}$ in d^{th} scene. The optimal control problem is defined by a state s $\in \mathbb{R}^n$, an action $\mathbf{a} \in \mathbb{R}^m$, a state transition function $T(\mathbf{s}_k, \mathbf{x}_k) : \mathbb{R}^{n+m} \to \mathbb{R}^n$, and a reward function $R(\mathbf{s}_k) : \mathbb{R}^n \to \mathbb{R}$. We assume the reward function is a function of state, $R(\mathbf{s}_k = \{\mathbf{x}_k, \mathbf{f}_k\})$, including state of the actor $\mathbf{x} \in \mathbb{R}^p$ and the features of the environment $\mathbf{f} \in \mathbb{R}^q$ such that $n = p + q$. We compute the optimal action $\mathbf{a} \subset \mathcal{A}$ at each state $\mathbf{s} \subset \mathbf{S}$ for a new unobserved environment based on this reward function.

Assume a robot can optimally perform a task by following a nominal path of the task ζ^N in an environment without any obstacle. Nonetheless the collected demonstrations ζ_ds may not be identical to the nominal path because of the presence of obstacles in the corresponding environment and/or the noise. We use GMM/GMR, computing a generalized path/trajectory across a set of suboptimal demonstrations, to estimate the nominal model of the task. Hence, the recovered reward function must be a function of the nominal model of the task, as well as of the corresponding features of environment, e.g. position of an obstacle $\mathcal{O}$. Features of the environment depend on $\mathbf{x}$ given a scene, however, for the sake of simplicity, we write $\mathbf{f}$ instead of $\mathbf{f}(\mathbf{x})$.

We consider the problem of performing a task to be an episodic, deterministic, optimal control problem with fixed time horizon K in discrete time, with a continuous state-space and action-space and a known world model. As per [1], given a set of task demonstrations $\mathcal{D}$ in different environments, the system is going to learn the underlying reward function R of $\mathcal{D}$. We decompose the underlying reward function $R(\mathbf{x}_k, \mathbf{f}_k)$ into two components: an imitation component $R_N(\mathbf{x}_k)$, whose optimal solution will be identical to the nominal path, and an emulation component $R_A(\mathbf{f}_k)$, that encodes the response of the robot to the environmental features.

Given a reward function $R = R_N + R_A$, we assume a robot is going to maximize the expected return $\rho^\pi = \sum_{k=1}^{K-1} R(\mathbf{s}_{k+1})$, giving rise to an optimal control policy, as follows

$$\begin{aligned} \pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{k=1}^{K-1} \left(R_N\left(\mathbf{x}_{k+1}\right) + R_A\left(\mathbf{f}_{k+1}\right)\right) \\ \text{subj. to} \quad \mathbf{s}_{k+1} = T(\mathbf{s}_k, \mathbf{x}_k), \\ \mathbf{x}_k \in \mathcal{U}, \end{aligned} \tag{5.1}$$

where $\pi = \{\mathbf{a}_1, ..., \mathbf{a}_{K-1}\}$ is the sequence of actions that a robot takes to accomplish the task and $\mathcal{U} \subseteq \mathcal{A}$ is a polyhedral region that is a feasible subset of the set of all actions $\mathcal{A}$. By executing the optimal policy π^*, the robot follows a sequence of states $\bar{\zeta} = \{\mathbf{s}_1, \bar{\mathbf{s}}_2, ..., \bar{\mathbf{s}}_K\}$, where $\mathbf{s}_1$ is a given initial condition.
The imitation and emulation components of the task are discussed in the following.

Imitation component of the reward function

We represent the model producing the nominal path to a new goal point as a DMP (Fig. 5.4). This DMP is produced by the *imitation learning* step described above: it is trained based on a generalized path obtained from a GMM/GMR model as described in [22]. As such, we assume it is the noise-free optimal solution to performing a task with no obstacles in the environment. This gives us the nominal reward function (Figure 5.5):

$$\begin{aligned} R_N\left(\mathbf{x}_k : \mathbf{Q}\right) = -\left(\mathbf{x}_k - \mathbf{x}_k^{\mathrm{N}}\right)^T \mathbf{Q}\left(\mathbf{x}_k - \mathbf{x}_k^{\mathrm{N}}\right) \\ \mathbf{x}_k^N \subset \bar{\zeta}\,,\ k = 1, ..., K \end{aligned} \tag{5.2}$$

where $\mathbf{x}_k^{\mathrm{N}} \in \mathbb{R}^n$ is a point on the nominal path ζ^N, which is learned as a DMP, and where the line segment $\overline{\mathbf{x}_k^N \mathbf{x}_k}$ is perpendicular to ζ^N.

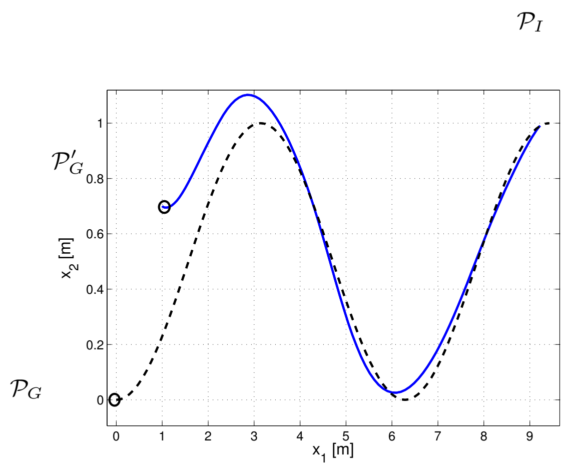


Figure 5.4: *The computed mimicking path model with goal point G and the produce path by imitation component to a new goal point G′.*

Emulation component of the reward function

The optimal solution for the emulation adaptation component problem is not invariant over different distribution of obstacles: $\mathbf{Sc}_d \ \forall \ d = 1, ..., D$. Since deviation from the nominal model in a new environment $\mathbf{Sc}_{new}$ is local, a Gaussian function with covariance matrix $\mathbf{R}$ can be learned from features of the demonstration data. This gives us the adaptation component of the reward function R_A:

$$R_A\left(\mathbf{f}_k : \mathbf{R}\right) = -exp\left(-(\mathbf{x}_k - \mathcal{O})^T \, \mathbf{R}^{-1} \, (\mathbf{x}_k - \mathcal{O})\right) \tag{5.3}$$

where $\mathbf{f}_k \in \mathbb{R}^q$ is a vector of the environmental features at $\mathbf{x}_k$, captured during d^{th} demonstration, e.g. $\mathbf{f}_k(\mathbf{x}_k, \mathbf{Sc}_d) = (\mathbf{x}_k - \mathcal{O})$, where $\mathcal{O}$ is the position of an added obstacle in d^{th} scene, $\mathbf{Sc}_d$.

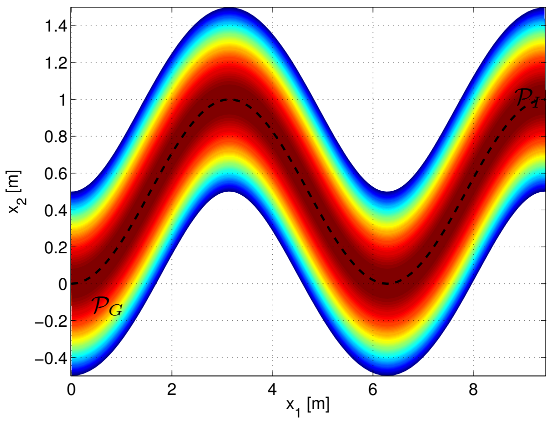
Accordingly, the general reward function (Figure 5.6) characterizing the demonstrated behavior in a different environment is a combination of the adaptation component (eq. (5.3)) and nominal component (eq. (5.2)) as follows:

$$R\left(\mathbf{x}_k, \mathbf{Sc}_d : \theta\right) = -\left(\mathbf{x}_k - \mathbf{x}_k^N\right)^T \mathbf{Q}\left(\mathbf{x}_k - \mathbf{x}_k^N\right) - e^{-\mathbf{f}_k^T \, \mathbf{R}^{-1} \, \mathbf{f}_k} \tag{5.4}$$

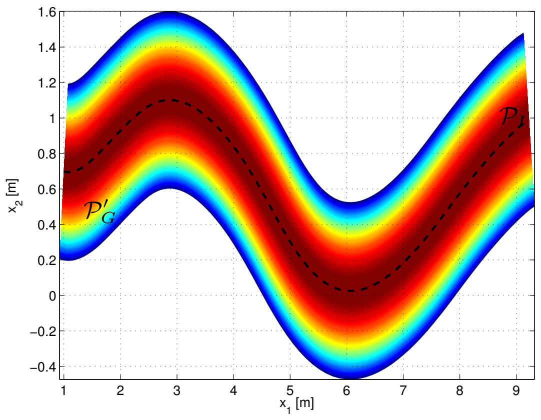
where $\theta = \{\mathbf{Q}, \mathbf{R}\}$, $\mathbf{Q}$ and $\mathbf{R}$ being positive definite matrices.

5.3.2 Inverse Optimal Control

Inverse optimal control aims at finding a reward function whose optimal solution is as close as possible to the demonstrations. Given an estimated



(a)



(b)

Figure 5.5: *(a) Contour of quadratic mimicking reward function and the obtained optimal solution (black dashed line); (b) the estimated generalized path by Mean-Path algorithm is used to train a DMP. A nominal path is generated by the DMP to a new goal point. The contour of quadratic imitation reward function based on the nominal path and corresponding optimal solution (black dashed line). The area with hot color has higher reward.*

reward function one can use the existing methods, such as dynamic programming or reinforcement learning, to find a solution, $\bar{\zeta}_{R(\theta,\mathbf{Sc}_d)}$ to eq. (5.1). Therefore, to learn the parameters of the reward function we minimize the cumulative distances between the optimal solution $\bar{\zeta}_{R(\theta,\mathbf{Sc}_d)}$ and the demon-

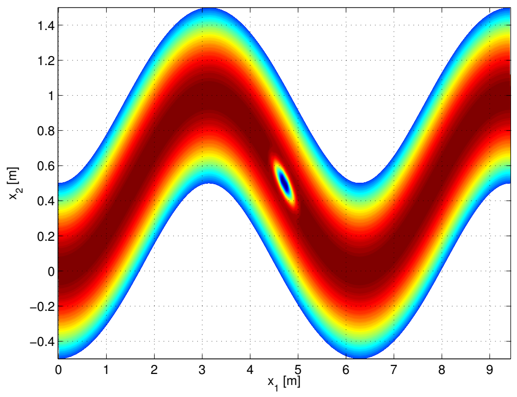


Figure 5.6: *Emulation component of the reward function corresponding to an obstacle in the environment.*

strations as follows:

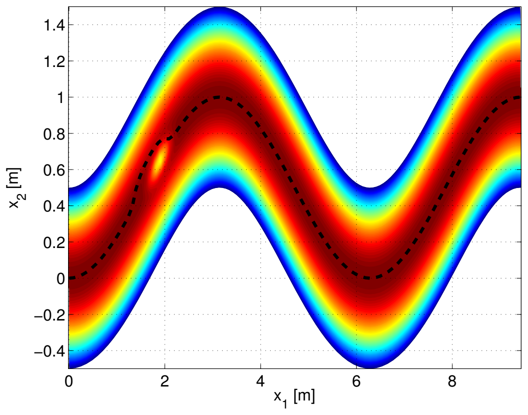
$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{d=1}^{D} \sum_{k}^{K} \left(\bar{\zeta}_{R(\theta, \mathbf{Sc}_d),k} - \zeta_{d,k} \right)^2 \tag{5.5}$$

where $\zeta_{d,k}$ and $\bar{\zeta}_{R(\theta,\mathbf{Sc}_d),k}$ are the corresponding points on the demonstration and the solution to the estimated reward function.

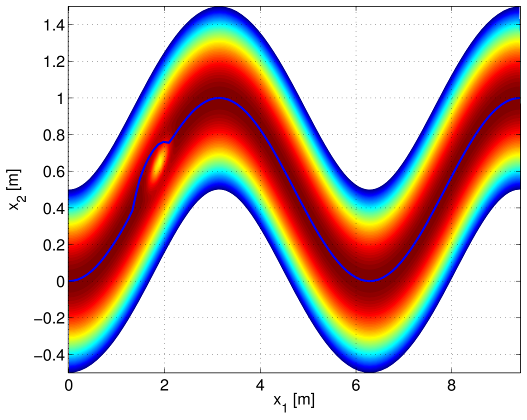
5.4 Solution to the learned reward function

In order to compute the optimal solution to the learned reward function for a new scenario, e.g. a new distribution of obstacles with different initial and goal positions, we maximize the return of eq. (5.1). In a finite-horizon problem, optimal control aims at finding the optimal policy by determining a sequence of actions $\bar{a}$ maximizing the expected return. In this paper, model predictive control is employed to find an optimal solution to the learned reward function with continuous state and action.

Consider a prediction time horizon H, the optimal action corresponding to the proposed problem in eq. (5.1) at k^{th} time step can be formulated as



(a)



(b)

Figure 5.7: *Contour of the recovered reward function; (a) the corresponding demonstration (black dashed line); (b) the obtained optimal solution to the recovered reward function. The area with hot color has higher reward.*

follows:

$$\begin{aligned}\bar{a}_k = \underset{\mathbf{x}_k}{\operatorname{argmax}} \sum_{h=k}^{H+k} - \left(\mathbf{x}_k - \mathbf{x}_k^N\right)^T \mathbf{Q} \left(\mathbf{x}_k - \mathbf{x}_k^N\right) - e^{-\left(\mathbf{f}_k^T \, \mathbf{R}^{-1} \, \mathbf{f}_k\right)} \\ \text{subj. to} \quad \mathbf{z}_{h+1} = A\mathbf{z}_h + B\mathbf{a}_h \\ \mathbf{x}_h = C\mathbf{z}_h \\ h = k, ..., (k+H)\end{aligned} \tag{5.6}$$

and

$$\begin{aligned} \mathbf{a}_h &\in \mathcal{A} \\ \mathbf{x}_h &\in \mathcal{X} \\ k = 1, 2, &..., K-1, \end{aligned}$$

where $\mathcal{X}$ and $\mathcal{A}$ are the polyhedral feasible sets of actor states and actions respectively, and $\pi^* = \{\bar{a}_1, ..., \bar{a}_{K-1}\}$ is a sequence of optimal actions where its corresponding sequence of optimal states is $\zeta^* = \{\bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_K\}$ with initial value $\mathbf{x}_1$.

In eq. (5.6) a linear dynamical system is considered as the transition function of the actor in eq. (5.1). It is assumed that the actor moves with constant velocity along the nominal path. To find a solution to eq. (5.5) and (5.6), we use a MATLAB toolbox, called *minConf*, with a quasi-Newton strategy and limited-memory BFGS updates [113]. In Figure 5.7, a task demonstration, obtained reward function and the path maximizing the return are shown.

It is worth mentioning that the asymptotic stability of the proposed MPC formulation for a path planning problem was discussed by Xu et al. [137].

The corresponding trajectory of the dynamical system for the time horizon H using a linear dynamical system in equation (5.6) can be written as follows:

$$\begin{bmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \\ x_{k+H} \end{bmatrix} = \begin{bmatrix} B & . & . & . & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & \dots & 0 \\ \vdots & & \ddots & \ddots & \\ A^{H-2}B & A^{H-3}B & \dots & AB & B \end{bmatrix} \begin{bmatrix} \mathbf{a}_k \\ \mathbf{a}_{k+1} \\ \mathbf{a}_{k+2} \\ \vdots \\ \mathbf{a}_{k+H-1} \end{bmatrix} + \begin{bmatrix} A^1 \\ A^2 \\ A^3 \\ \vdots \\ A^{H-1} \end{bmatrix} x_1 \tag{5.7}$$

At each time step k, the optimization in equation (5.6) finds a vector of inputs,

$$[\mathbf{a}_k, \mathbf{a}_{k+1}, \mathbf{a}_{k+2}, \dots, \mathbf{a}_{k+H-1}],$$

that is the optimal solution to the reward function based on equation (5.7) for the time horizon H. The first input of the computed vector of actions, $\mathbf{x}_k$, is taken as the action to be performed by the robot at time step k. The next state is then obtained by taking action $\mathbf{x}_k$ based on the transition model in equation (5.6) and this procedure repeats for every time step $k = 1, ..., K$. In this way, it is guaranteed that for a computed action at a step there is a feasible optimal solution up to the time horizon H according to the reward function. This formulation deals with some sort of local minima of the corresponding reward function (Figure 5.8). However the time horizon must be set properly. In this simulation we experimentally selected

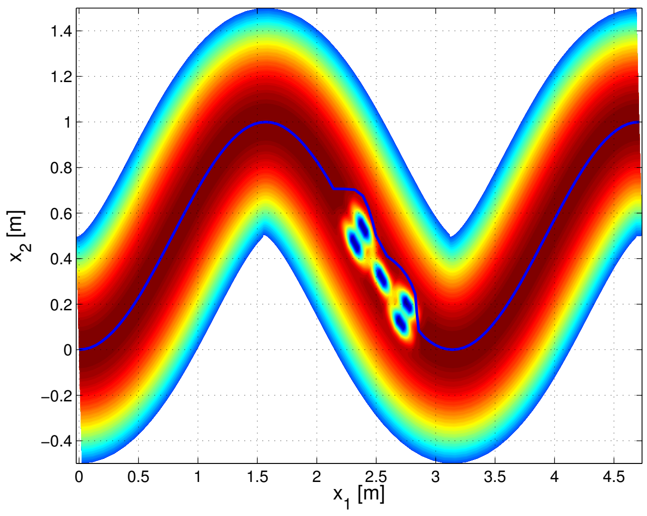


Figure 5.8: *Contour of a reward function based on an environment with five obstacles arranged to shape a concave form of reward function. The computed solution is able to cope with this concave arrangement of the obstacles.*

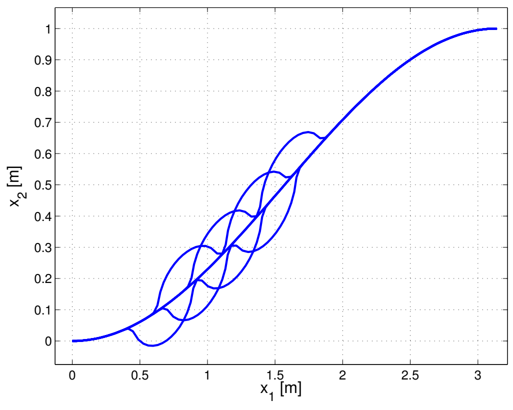


Figure 5.9: *A set of eight artificially generated paths.*

$4 \leq H \leq 12$. A small time horizon does not provide enough information about the future rewards while a big time horizon results in inaccurate prediction horizon of the reward with the linear model.

5.5 An example of using the proposed approach

In this section, to show how the proposed workflow of robot LfD in Figure 5.2 produces a task model from a set of task demonstrations, we use the proposed approach to compute a reward function explaining an artificially generated data set. First we consider a task model to be a nominal path (Figure 5.10). In order to generate a set of task demonstrations, we add an obstacle at a different position in each environment. We use the task model, nominal path, combined with a model of obstacle avoidance to simulate the demonstrations. A set of eight trajectories is then generated building our data set (Figure 5.9). We consider this data set along with the position of the added obstacle to be the only available information of the corresponding task. Thus, neither the corresponding nominal model of the task nor the demonstrated obstacle avoidance policy is explicitly provided through the available data set. We are going to estimate a corresponding model using the proposed multi-layered robot LfD that explains all the demonstrations.

First, at the mimicking level (level I of Figure 5.2), we use the Mean-Path algorithm. At this level, a corresponding generalized path across all demonstrations is estimated to represente the task model in an environment with no obstacle. Figure 5.10 shows the used nominal path to generate the trajectory and estimated one in the top and bottom figure, respectively. To evaluate the generalized path obtained by the Mean-Path algorithm we use the mean square error measuring the distances between the ground truth and the generalized path as follows:

$$MSE_{Gp} = \frac{1}{m}\sum_{i=1}^{m}(\zeta_{Np,i} - \zeta_{Gp,i})^2$$

where $\zeta_{Np,i}$ and $\zeta_{Gp,i}$ are the data points of the ground truth and the generalized paths, and m is the number of collected points of every path. The mean square error of the generalized path with respect to the nominal path is $MSE_R = 0.94 \times 10^{-3}\ [m]$.

Next, at the imitation level, we use the estimated nominal path to train a DMP model. We then use this model to generate a new path whenever the task must be generalized to a new goal point (Figure 5.12). Finally, at the emulation level, we use inverse optimal control. We form a quadratic reward function (equation (5.2)) such that a robot that follows the previously obtained path collects the maximum reward. Then, we add the emulation component (equation (5.3)) and compute the corresponding parameters characterizing the demonstrated responses to the obstacle. The obtained reward function incorporates a nominal model to perform the corresponding task and the user response to an obstacle into a single reward function. The parameters of the reward function are $Q = 10$ and $R = [2500, 215]$.

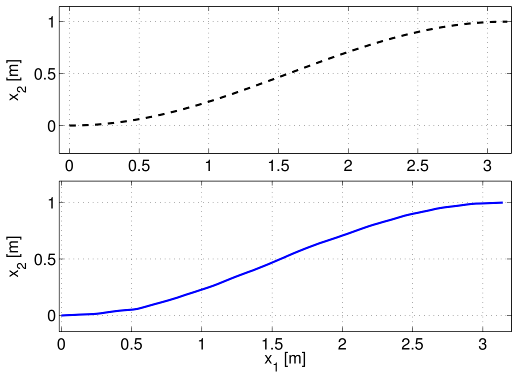


Figure 5.10: *Considered nominal path (top) and the corresponding estimated path by Mean-Path algorithm (bottom).*

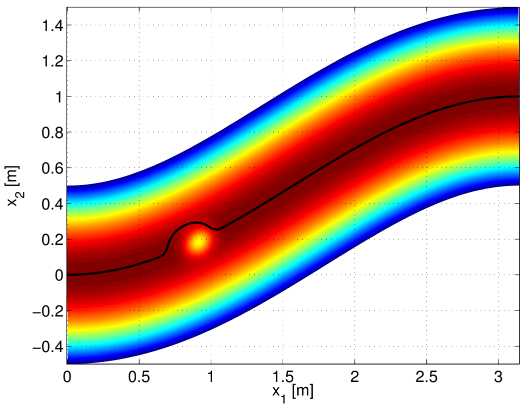


Figure 5.11: *Contour of the recovered reward function of a demonstrated path (black line) in the corresponding environment with added obstacle.*

To evaluate the path obtained by the computed reward function we define a mean square error measuring the distances between the demonstrated and reproduced paths as follows:

$$MSE_R = \frac{1}{nm}\sum_{i=1}^{m}\sum_{d=1}^{n}(\zeta_{d,i} - \zeta_{R,i})^2$$

where $\zeta_{d,i}$ and $\zeta_{R,i}$ are the data points of the demonstrated and reproduced paths. n and m are the numbers of paths and the collected points of every path. The mean square error of the reproduced paths using the obtained reward function is $MSE_R = 1.077 \times 10^{-3}$ $[m]$.

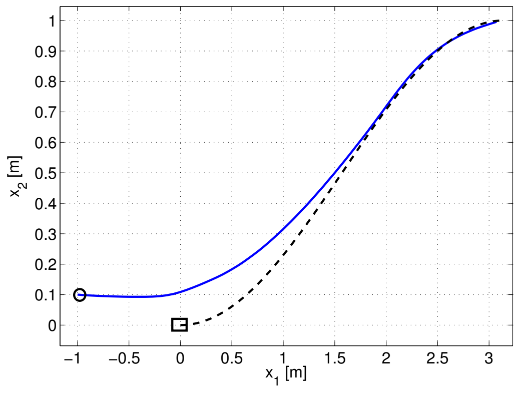


Figure 5.12: *The estimated nominal path (black dashed line) to the goal point (marked with square) and the generated path to a new goal point (marked with circle) with dynamic movement primitives model.*

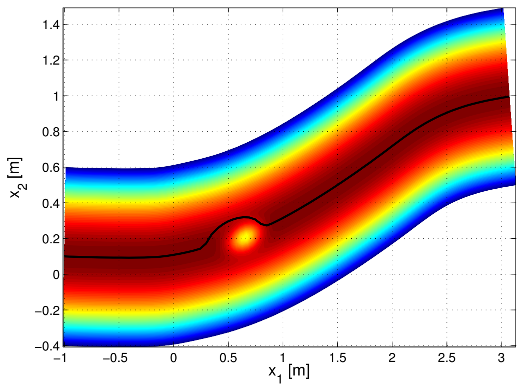


Figure 5.13: *Contour of the recovered reward function based on the path generated with dynamic movement primitives.*

This example shows how the workflow of robot LfD allows a robot to learn from a set of task demonstrations a task dependent model and a model to avoid an obstacle. The emulation component of the model, a policy to react to an obstacle,can be transferred across tasks with different nominal path (Figure 5.13), which allows a robot to reuse part of obtained knowledge for a new task execution.

5.6 Conclusion

In this chapter, we discussed robot LfD at emulation level for the problem of learning obstacle avoidance. We decomposed the robot LfD model into two components, nominal component and adaptation component (emulation component). This decomposition allows use to transfer the emulation component across different tasks. Therefore, the obstacle avoidance learned in a specific task can be used in another task performance.

This chapter represents the general picture of the proposed workflow of robot LfD, shown in Figure 1.4. First, at mimicking level, a nominal path or trajectory is computed from a set of noisy demonstrations. This noise might be measurement noise or deviation from the nominal model due to an added obstacle. The robot can use the nominal path to carry out a demonstrated task in an environment with no obstacle.

Second, at imitation level, the computed nominal path is used to train DMP, which is then used to generate a smooth path to a new goal point. Next, at emulation level, a reward function is built that characterizes both the task model and the response to the added obstacle to the corresponding environment. Finally, the learned emulation component and the path generated to a new goal point are used to build a reward function characterizing the task with a new goal point and the response to the obstacle.

CHAPTER 6

Experimental results

6.1 Outline of the Chapter

The problem of robot learning from demonstration has been discussed in the former chapters at different levels, namely mimicking, imitation, and emulation. In this chapter, a problem of computing a nominal profile of a workpiece after deburring operation is considered to be robot learning from noisy demonstrations at mimicking level. Furthermore, two experiments with a da Vinci and a UR5 robot are presented to illustrate how the workflow proposed in Chapter 2 allows a robot to automatically build a task model from a set of demonstrations.

In the example with the da Vinci robot, a pick and place task of an object within an environment, a designed structure, with an obstacle is studied. During surgeon training, pick and place task is very common. Hence, it is of utmost importance to build a model of this simple task, from a set of expert demonstrations, that it can be used to help a trainer to follow the optimal path, or to evaluate the trainer performance.

The data set collected by the da Vinci robot is divided into two sets, a training set and a test set. We propose an error metric that is a distance between a reproduced path and the corresponding path in the test or training set. Furthermore, we define a precision measure to show the improvement of the solution with respect to the generalized path computed by the Mean-Path algorithm. The comparisons of the error and the precision of the training and test set demonstrate the efficiency of the proposed method and the obtained model.

In the example performed with the UR5 robot, a model of sweeping a cube into a dustpan is learned from few demonstrations in a scene with an obstacle. The sweeping task is a typical example of household task. This model is then used by UR5 to perform the sweeping task in different scenarios, i.e. different positions of the obstacle and dustpan. This example shows how a user can teach a robot a household task with desired response to a specific object.

6.2 Deburring example

As an example of robot learning from noisy demonstrations at the mimicking level, we present a problem of automatically computing a deburring profile of a workpiece from a set of demonstrated profiles, requiring very high precision. In order to have a robot that learns from demonstrations how to autonomously perform the deburring task, the robot must compute a nominal profile of the workpiece from a set of profiles collected after the deburring operation. The nominal profile will then provide a baseline for another controller that computes the feed rate and velocity of the deburring tool. Therefore, an accurate nominal profile results in the minimum deburring residuals on the workpieces while the deburring tool does not penetrate in the workpiece.

We assume that an expert cannot perfectly remove the burrs of a set of workpieces because human demonstration is always suboptimal. Hence, the set of demonstrated profiles have still deburring residuals with small sizes at different positions along the profile of the workpieces.

In order to simulate a set of suboptimal demonstrations, a data set of nominal linear profiles and a data set of nominal circular profiles with deburring residuals have been generated (Fig. 6.3). To do so, a set of seven linear profiles with the same length and with different deburring residuals was first prepared (Fig. 6.1 and 6.2)[1]. We then consider the nominal profile, linear or circular, consisting of a number of profile segments with identical length. For each segment, a random generated number determines whether the deburring residual is added or not, then a deburring residual profile is selected randomly and added to the nominal profile.

[1]The photos are not with the same scale size.

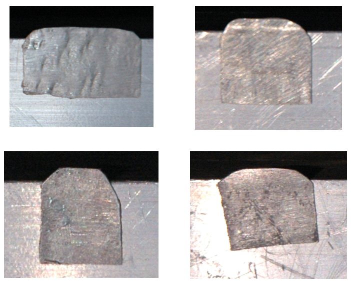


Figure 6.1: *Sample workpieces with the artificial deburring residuals.*

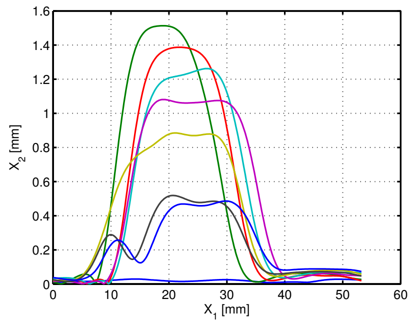


Figure 6.2: *The seven deburring residual models used to generate the data set.*

In order to estimate a nominal profile, we use the proposed algorithm, GMM/GMR [27] with different numbers of components and NLPCA [115] with different modeling types.

To evaluate the precision of each approach, to estimate the underlying nominal profile, e^2 was computed (Table 6.1)[2]. The network structure of NLPCA and the number of GMM components are selected such that the error is minimized. For the linear profile, NLPCA and GMM/GMR result in error values of 127.1 and $127.8\,[mm]$, while the Mean-Path algorithm error is $99.6\,[mm]$. The obtained circular profile by NLPCA and GMM/GMR results in relative error of 1433.4 $[mm]$ and 75.9 $[mm]$ with respect to the circular nominal profile, where the profile obtained by the Mean-Path algorithm results in an error of 75.3 $[mm]$.

In Fig. 6.4(b) and 6.5(b) the green shaded areas represent the nominal workpiece. Although the circular profile obtained with GMM/GMR has almost similar error compared with the one obtained by the Mean-Path al-

[2]In Table 6.1, NA represents the values that are not available.

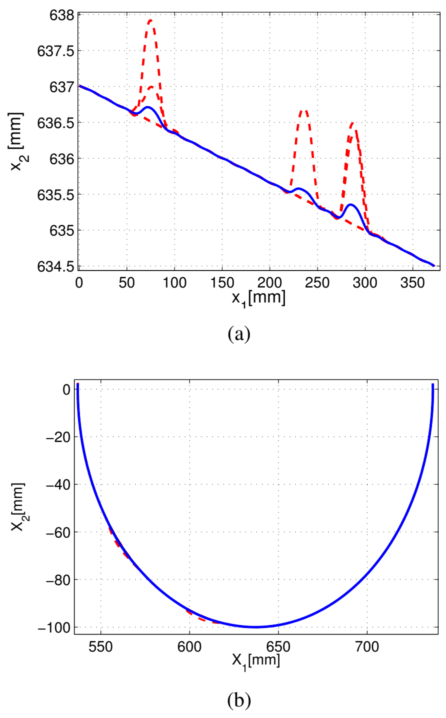


Figure 6.3: *(a) Ten linear profiles with randomly added artificial deburring residuals (red dashed lines), according to the models in Figure 6.1, and the computed mean profile (blue line); (b) Ten circular profiles with randomly added artificial deburring residuals (red dashed lines), and computed mean profile (blue line).*

gorithm, the workpiece will be destroyed if the robot follows this profile during execution of deburring as it penetrates the workpiece in some parts, see Fig. 6.5(b) where the obtained profile by GMM/GMR (black line) penetrates the green shaded area. The obtained errors with different approaches illustrate the superiority of the proposed approach for both linear and circular profile.

6.3 Experiments with a da Vinci robot

The use of robotic surgery has been increased during the last couple of decades making it a common procedure in many hospitals. For example, the da Vinci system has proved its performance in many clinical studies and it is installed in thousands of hospitals worldwide [49]. However, robotic surgery has still significant potential to improve its performance. Many

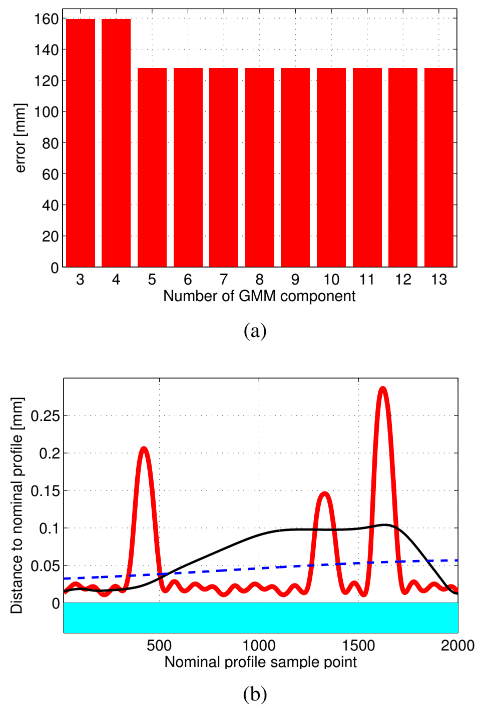


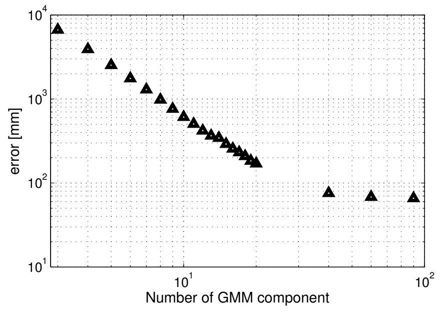
Figure 6.4: *Ten linear profiles: (a) e^2 of the obtained profile by GMM/GMR; (b) the distances between the linear nominal profile and the computed paths by the Mean-Path algorithm (red thick line), GMM/GMR with 10 components (black line) and NLPCA (blue dashed line). The shaded green area represents the workpiece.*

Table 6.1: *e^2 $[mm]$ of deburring data set by using NLPCA, GMM/GMR and Mean-Path for linear/circular profile*

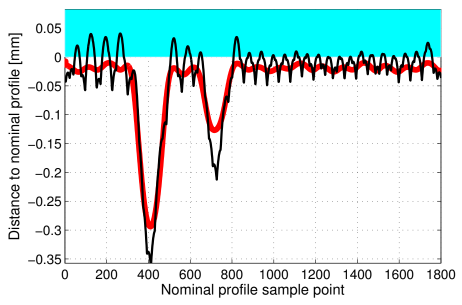
		Linear	Circular
Mean-Path		99.6	75.3
GMM/GMR		127.8	75.9
NLPCA - Neural Network	C-B-S	NA	NA
	C-B-H	NA	NA
	C-I-S	NA	1433.4
	C-I-H	NA	18534.5
	NC-B-S	127.1	NA
	NC-B-H	586.4	NA
	NC-I-S	169.5	NA
	NC-I-H	226.8	NA

studies have been conducted to develop more intelligent surgical robotic systems to improve the quality of the robotic surgery [4].

One of the research topics in this domain is surgical task automation. The main motivation of surgical task automation is to reduce the operation



(a)



(b)

Figure 6.5: *The case of ten circular profiles: (a) e^2 of the obtained profile by GMM/GMR; (b) the distances between the nominal profile and the computed paths by the Mean-Path algorithm (red thick line) and by GMM/GMR with 60 components (black line). The shaded green area represents the workpiece.*

effort of a surgeon and hopefully to reduce the surgery time [99]. However, the automation of a surgical task has challenging problems to be solved. One of these problems is the adaptation of a learned path according to the state of the environment where the environment may change from one task execution to another, e.g. obstacle avoidance in different environments.

In many surgical tasks more than one slave robotic instrument is used to perform a task. Although different robotic instruments are necessary to perform a complex task, they cause some constraints on movement of another instrument. Hence, each instrument must avoid collision with others during task execution. In order to automate a robotic surgery task, robot learning from demonstration that allows for automatic extraction of a task model in different environments can be used.

In the following section, according to the workflow proposed in Chapter 2, we decompose robot LFD into three components, namely mimicking,

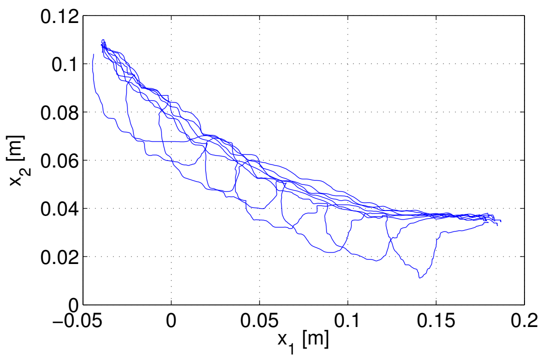


Figure 6.6: *The path collected by da Vinci during pick and place task in the presence of an obstacle at different positions.*

imitation and emulation, and we use it to build a model of a task from a set of task demonstrations.

6.3.1 Experimental setup of da Vinci example

Many surgical tasks are constrained by the small available space within a patient’s body. To obtain a constrained environment a structure is designed and used to perform a pick and place task. The task of picking and placing an object is a common task during surgeon training. The designed task is to pick an object from $\mathcal{P}_I$ and put it at $\mathcal{P}_G$ of the structure (Figure 4.10).

In this experiment, the operator was asked to move the object from $\mathcal{P}_I$ to $\mathcal{P}_G$ with da Vinci slave tool. He was asked to maximize the distances from both walls during task execution. The operator repeated the task three times (Figure 4.11) in order to collect a set of demonstrations.

Then, a marker was fixed to the scene as an obstacle and the operator repeated the task. He avoided collision with the obstacle during task execution. The obstacle can be also regarded as another instrument with which collision has to be avoided. The distance from the instrument to the marker was used as an environmental feature $\mathbf{f}$. A set of eight demonstrations with different positions of the marker in the scene was collected that constitutes our data set and can be seen in Figure 6.6. As shown in these figures, the operator responded differently to the same environmental stimulus, resulting in noisy responses to the same obstacle.

We assume here that in a real scenario a task model can be represented by a reference path or trajectory and this reference path is adapted to a new scene, e.g. a scene with an added obstacle at a new position. This adaptation is represented by deviation from the reference path. Hence, a

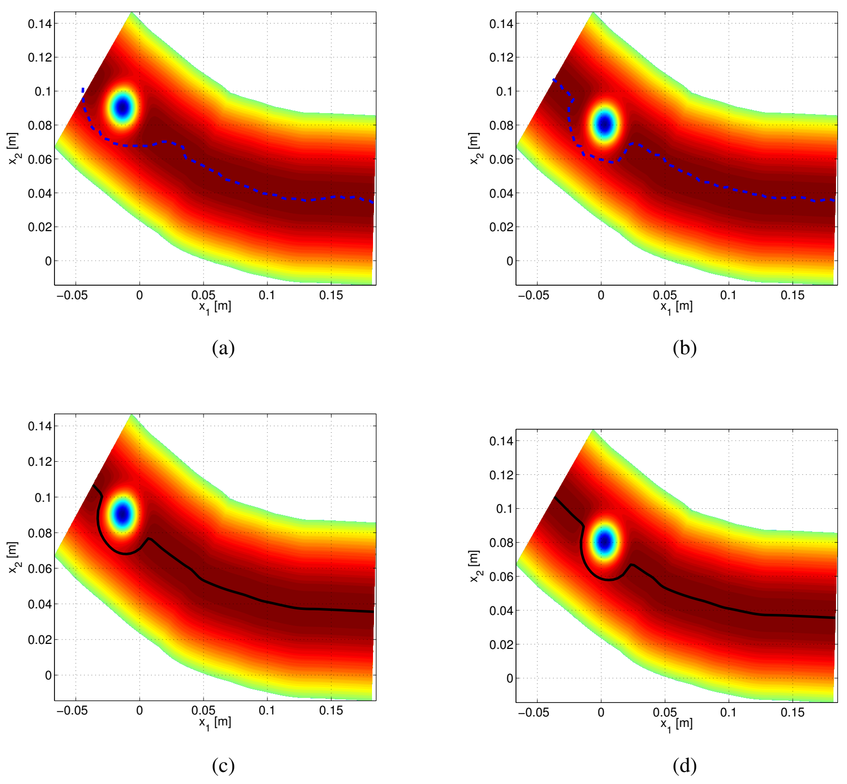
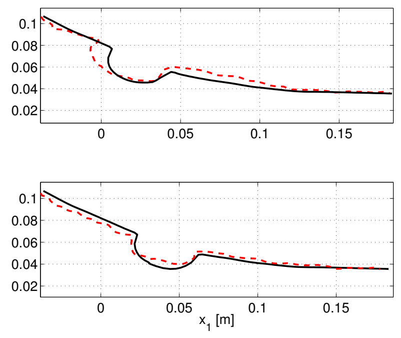


Figure 6.7: *The contour of the learned reward function learned by the training set of the da Vinci experiment. The area with hot color represents the higher reward; (a) and (b) corresponding demonstrations (blue dashed line); (d) and (c) the computed optimal solution that has the maximum cumulative reward based on the obtained reward function.*

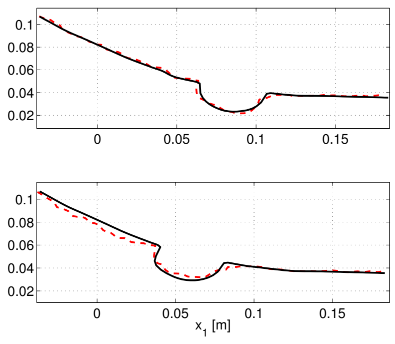
reference path and an adaptation functionality are the main components of a task model.

6.3.2 Learning the parameters of the model

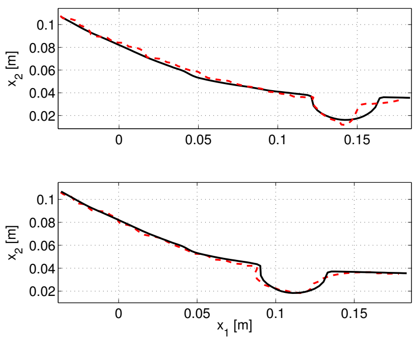
According to the proposed workflow in Figure 1.4, at the mimicking level of robot LfD, a generalized path is computed by the Mean-Path algorithm from a data set of task demonstrations without any obstacle (Figure 4.11). The evolution of the error and the residuals of the data during Mean-Path computation are shown in Figure 4.12. The robot can perform the task of moving an object from the point $\mathcal{P}_I$ to the point $\mathcal{P}_G$ in the structure without any obstacle by playing back this smooth path. Next, at the emulation level



(a)



(b)



(c)

Figure 6.8: *The demonstrated paths within the test sets (dashed red line) and the corresponding reproduction by the obtained reward function (solid black line).*

of the workflow the parameters of the reward function are computed. We consider the computed generalized path to be a reference path for building the corresponding reward function. A sequence of points that maximize the reward will result in generalized path adapted to a scene with an added obstacle.

We divide the collected data set into two sets, training set and test set (Figs. 6.7 and 6.8), where the training set is used to learn the parameters of the reward function and the test set is used to evaluate the obtained model. The parameters of the reward function are computed by using equation (5.5). The estimated parameters of the reward function result in a minimum distance between the corresponding demonstration and a path that has the maximum cumulative reward.

The obtained parameters of the corresponding reward function are as

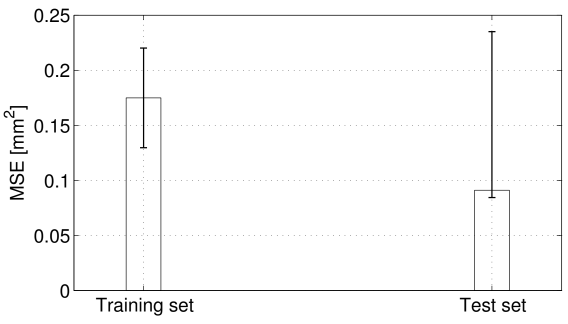


Figure 6.9: *The mean square error (MSE) of the reproduced path using the obtained reward function. The error bars show the lower and upper value of MSE in the training set and the test set.*

follows: $Q = 200$ and $R^{-1} = diag[475.8 \quad 8576.3]$. The obtained reward functions of two demonstrated paths within the training set as well as their corresponding optimal solution and demonstrations are shown in Figure 6.7.

The optimal solutions corresponding to different environments are computed using the obtained reward function. Although the demonstrations are noisy, the resulting optimal solutions with the obtained reward function are smooth paths, as shown in Figure 6.7 and 6.8. This is, indeed, highly desirable in many robotic tasks.

In order to evaluate the obtained model, we propose two metrics. First, we propose a mean square error

$$MSE_R = \frac{1}{nm}\sum_{d=1}^{n}\sum_{i=1}^{m}(\zeta_{d,i} - \zeta_{R,i})^2$$

where $\zeta_{d,i}$ and $\zeta_{R,i}$ are the data points of the demonstrated and reproduced paths. n and m are the number of paths in the training or the test set and the collected points of every path.

The mean square error of the reproduced path using the obtained model with the environment corresponding to the training set and the test set are shown in Figure 6.9. The mean square error corresponding to the training set and test set are equal to $e = 0.174924[mm]$ with $var(e) = 0.0041$ and $e = 0.0911[mm^2]$ with variance $var(e) = 0.0064$, respectively. This error sums the distances between the reproduced path and the Mean-Path as well as the distances between the responses of the operator and the model to each environmental feature. Hence, in order to evaluate how the added adaptation functionality to the reward function improves the obtained solution, we

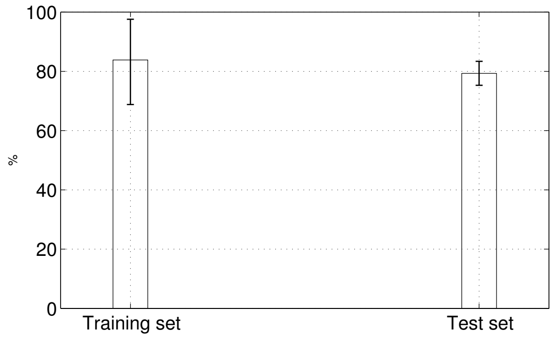


Figure 6.10: *The improvement of the reproduced path with respect to use of the computed Mean-Path.*

propose a precision/improvement metric as follows:

$$Pr = \frac{MSE_{Mp} - MSE_R}{MSE_{Mp}} \times 100\%$$

where

$$MSE_{Mp} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{i=1}^{m} (\zeta_{d,i} - \zeta_{Mp,i})^2.$$

The improvement of the obtained solution with respect to the use of Mean-Path along with their upper and lower values for the training set and the test set are shown in 6.10. Comparing the error and precision value of the test and training set (Figs. 6.9 and 6.10) illustrates that the obtained controller from the training set generates paths corresponding to the environment of the test set that are close to the demonstrations (Fig. 6.8).

Although the obtained generalized path and the adaptation model enable a robot to perform the task with different positions of the obstacle, the robot will not succeed if the target point changes to $\mathcal{P}'_G$. In this regard, dynamic movement primitives may be used at the imitation level to generate a smooth path to a new goal point. Then, a reward function based on the computed path can characterize the required path to perform the task with a new target point and position of the obstacle.

The reward function combines the imitation or mimicking (nominal) component with the emulation (adaptive) component. Hence, the resulted optimal path generalizes the observed behavior to a new target point $\mathcal{P}'_G$ and a new environment, e.g. scene with a new obstacle position. The computed generalized path with the Mean-Path algorithm results in a relatively

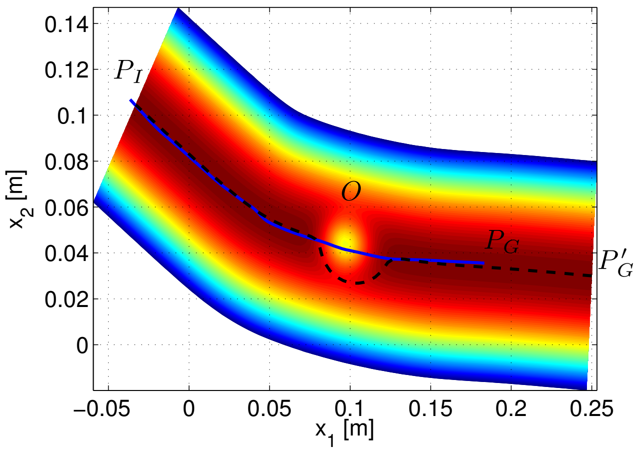


Figure 6.11: *The estimated generalized path (blue line) with the initial point $\mathcal{P}_I$ and the goal point $\mathcal{P}_G$. The path computed by dynamic movement primitives (black dashed line) for the new goal point $\mathcal{P}'_G$. The contour of the corresponding reward function based on the path computed by dynamic movement primitives and the corresponding computed path with maximum reward is shown with a black dashed line. $\mathcal{O}$ is an obstacle added to the scene.*

smooth path from a set of noisy task demonstrations. This generalized path is used to train a DMP model that scales the generalized path to a new target point that is used as a reference path to build the reward function. The emulation component is used to capture the response of the demonstrator to the features of the environment. In fact, this component adapts the reference path to a new environment. Finally, a reward function combines all these three components resulting in a path to a new goal point and adapted to an environment with a new distribution of the obstacles. This reward function can cope with noisy demonstrations, different target positions and arrangements of obstacles. The results shown in Figure 6.11 illustrate the effectiveness of the approach in capturing the response to an obstacle and generalize the learned skill to the new target point.

6.3.3 Transferring the emulation across different scenarios

In many machine learning problems a model is built based on a training data set and it is later used for classification, clustering or prediction given a new data set. However, many of the proposed methods work well under some assumptions. For example, it can be assumed that the distribution of

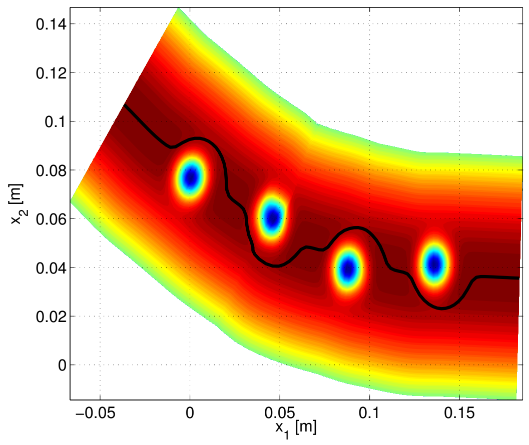


Figure 6.12: *The emulation component of the learned model is used to avoid obstacles in an environment cluttered with four obstacles.*

the training data set and of a new data set are the same. If the new data set has a different distribution, a new model may be required [101]. One way to avoid recollecting a new training data set and rebuilding a model is *knowledge transfer* or *transfer learning*. This is significantly beneficial since knowledge learned based on an experiment can be reused across many experiments with slightly different models. For example, knowledge transfer is an approach to reduce the computation cost of finding an optimal policy in reinforcement learning [66].

In the context of robot learning from demonstration, transferring the learned knowledge to a new environment or to a new task with a different model is truly advantageous. In this regard, decomposing a task model into two components is beneficial. First, the imitation and emulation components can be reused across different environments, e.g. environments with different positions of obstacles. Second, the emulation component can also be used across different tasks with different reference paths. The use of the learned knowledge across different situations enhances the learning process, and the knowledge learned in one experiment can be generalized across different scenarios.

In Figure 6.12, the obtained model is used to compute the optimal path for a more complex environment in the presence of four obstacles. This figure shows that the emulation component can be learned from a simple demonstration but it can be used in an unseen and complex environment. Next, the emulation component is used to find the optimal path of

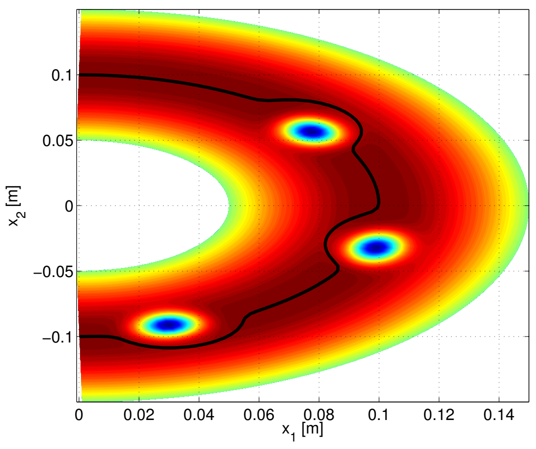


Figure 6.13: *The emulation component of the learned model with da Vinci data set is used to avoid obstacles in a task with a different task model.*

a task with completely different reference path, as shown in Figure 6.13. This example shows that the knowledge obtained in one experiment can be transferred to tasks with different models. This property of the proposed approach of robot learning from demonstration boosts the efficiency of the learning procedure.

The proposed approach of robot LfD enables a robot to cumulate knowledge from demonstrations of different tasks and build obstacle avoidance models. These models can be later used in combination with an unseen task model, as shown in Figure 6.13.

6.4 Sweeping task experiment with UR5

In order to learn a task model, e.g. sweeping, not only a robot must extract a model of how to perform the task but also it must be capable to extract a model of how to react to different environmental changes. In this regard, one of the important environmental entity that is common across many robotic tasks is the existence of an obstacle. In many robotic tasks, obstacle avoidance problem is a major concern. Hence, a robot must extract both model of the task and a strategy for obstacle avoidance.

Consider a task of sweeping rubbish, shown in Figure 6.14, where there might exist different obstacles in the scene. We aim at teaching a robot how to sweep a green cube to a dustpan and avoid collision with some objects, by providing a few demonstrations of the task. To do so, the task

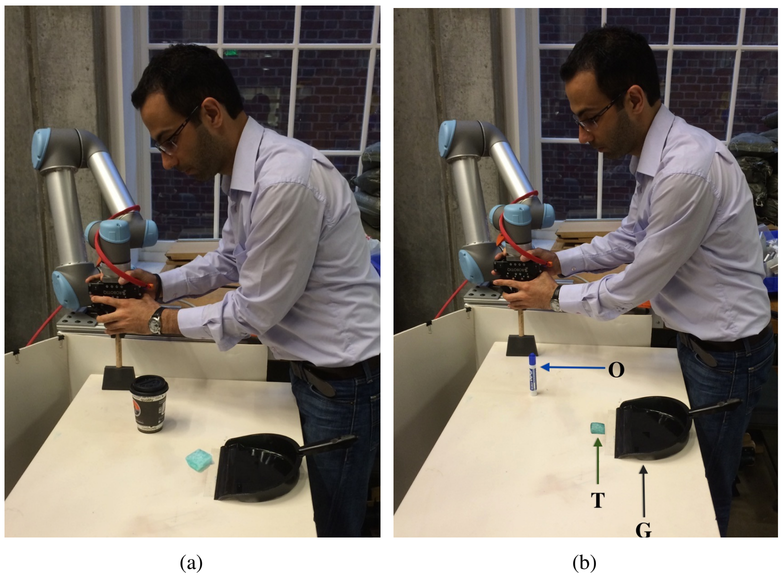


Figure 6.14: *The experimental setup of sweeping example. The UR5 robot mounted on the table where the origin of the reference frame is the center of the base of the robot (as shown in Figure 6.18) where x_1 is parallel with the width of the white table and it is from the center of the robot's base to the right side of the figure.*

of sweeping the green cube to the dustpan is demonstrated a few times in the presence of two objects, a marker and a cup, which we do not want to sweep.

The proposed approach of robot LfD is used to generalize the demonstrations over different environments with new positions of the dustpan and different positions of the obstacles. The approach also allows the robot to combine different models, e.g. the model obtained for the marker and the cup, into a single reward function. This also allows the robot to perform the task in the presence of both object classes. Further details of the sweeping example is presented in Chapter 6.

6.4.1 The UR5 robot

The UR5 robot is a six degree of freedom flexible industrial manipulator manufactured by Universal Robots (Figure 6.16). It has six revolute joints and cylindrical extruded aluminum links. Each joint has a Brushless servo motor and a harmonic drive reducer. Two sizes of motors are used in the joints where the first three joints have the bigger motors (see Table 6.2 for the specifications). This robot is designed for industrial applications

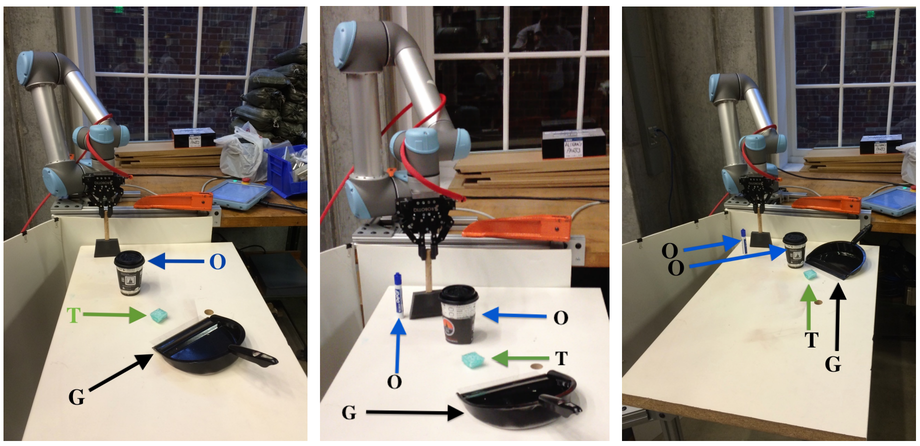


Figure 6.15: *Three different scenarios of sweeping task with unseen position of the dust-pan as well as unseen position of the marker and cup (obstacles): scenario 1, 2 and 3 are shown in (a), (b) and (c).*

suitable for small and medium-sized companies that require less than five kg of payload. Nonetheless, it is increasingly used in academic studies due to its comparatively low price. The robot weighs 18.4 kg and it can reach a point with maximum distance of 85 cm from the center of its base.

The robot is provided with a control box, a twelve inches touchscreen and a programming interface, called Polyscope (see Figure 6.16[3]). The control box comes with a Linux based Operating System. Furthermore, a teaching button on Polyscope allows a user to move the robot and record a set of waypoints required to perform a task, which can be later played back to perform the task. An emergency bottom on Polyscope is available to stop the robot in emergency situations.

The Universal Robot can be controlled at three different levels: The Graphical User-Interface (GUI) level, the script level and the C-API level. The graphical user-interface has some button to move the robot either in the joint space or Cartesian tool space. The joint angle values as well as end-effector positions and orientations can be read directly on the touchscreen.

At the script level user can program the robot using Polyscope. The company provides the robot with a programming language, called URScript. The URScript has variables, types, functions, and control statements, such as loop. In addition URScript has a number of built-in variables and functions which monitors and controls the I/O and the movements of the robot.

[3]Image is taken from : http://scandasia.com/danish-robots-a-success-in-malaysia/universal-robot-ur5/

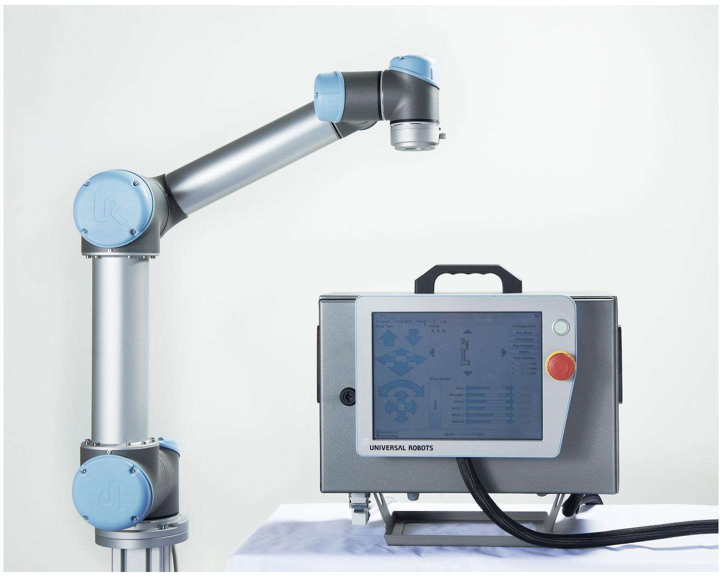


Figure 6.16: *UR5 robot and its Polyscope control panel.*

For example, URScript has a command to read the joint angle values as well as the tip center pose (TCP) value, including the position and orientation of the center of the end-effector. The controller of the robot solves the forward kinematics of the robot based on the measured joint angle values to compute the pose of the end-effector.

The robot can be programmed in two different ways using URScript. Firs, one can write a program on Polyscope using URscript to independently control the robot. The URScript program is executed in real-time on the URControl RuntimeMachine (in the control box) and the RuntimeMachine communicates with the robot with a frequency of 125 Hz. Second, Polyscope can be connected to a personal computer (PC) through an Ethernet cable. A script written on PolyScope as a client application communicates over a TCP/IP connection with another program written on the PC as the server. In our experiment, we used client-server connection over a TCP/IP communication.

6.4.2 Robot mechanical specifications

The UR5 manipulator has seven links $l_i : i = 0, ..., 6$ and six revolute joints $j_i : i = 1, ..., 6$ (Figure 6.17). Each link is moved by a brushless motor through the RuntimeMachine on the control box. The joint angle limitations as well as the maximum angular velocity of the motors at each

Table 6.2: *Electrical and mechanical specifications of the UR5 motors.*

	Big motor	Small motor
q_{max}	$\pm 2\pi$rad	$\pm 2\pi$rad
$\dot{q}_{max}$	$\pm 3.2\frac{rad}{s}$	$\pm 3.2\frac{rad}{s}$
$\ddot{q}_{max}$	$\pm 15\frac{rad}{s^2}$	$\pm 15\frac{rad}{s^2}$
τ_{max}	150 Nm	28 Nm
Static friction	0	0
Dynamic friction	0.11	0.13
Viscous friction	0.4	0.3
Dynamic friction backdrive	0.07	0.07
Viscous friction backdrive	0.6	0.25
Torque constant	0.13	0.14

Table 6.3: *Denavit-Hartenberg parameters of UR5.*

Link i	d_i [mm]	a_i [mm]	α_i [rad]
1	89.16	0	$\frac{\pi}{2}$
2	0	-425	0
3	0	-392.25	0
4	109.15	0	$\frac{\pi}{2}$
5	94.65	0	$-\frac{\pi}{2}$
6	82.3	0	0

joint and motor specifications are reported in Table 6.2.

The end-effector position and orientation can be controlled in both joint space and tool space. To control the robot in tool space the RuntimeMachine computes the forward kinematics of the robot. To derive the forward kinematics, Denavit-Hartenberg convention can be used (see Figure 6.17 and 6.18[4]). The Denavit-Hartenberg parameters of the UR5 manipulator are reported in Table 6.3 based on the reference frames shown in Figure 6.18. Accordingly, the position and orientation of i_{th} coordinate frame can be expressed in the $\{i-1\}_{th}$ coordinate frame by the following homogeneous transformation matrix:

$$T_{i-1}^{i} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i S\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

where θ_i is the joint angle of i_{th} joint. C and S denote the $cos(.)$ and $sin(.)$, respectively. The corresponding transformation matrix T_{i-1}^{i} to each joint can be computed by substituting the Denavit-Hartenberg parameters a_i, α_i, d_i and θ_i from Table 6.3 into equation (6.1), and eventually the full

[4]Figure 6.17 and 6.17 are taken from Katharina Kufieta "Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments", Master thesis, 2014

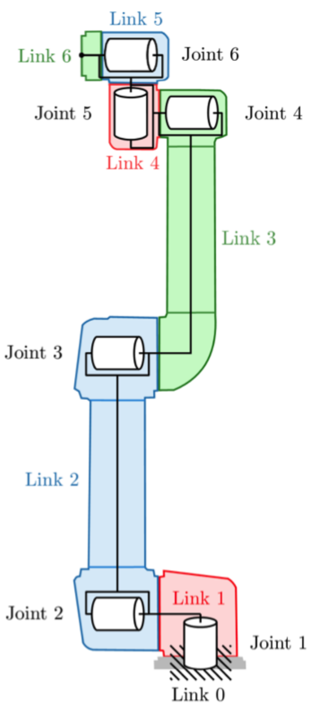


Figure 6.17: *Sketch of UR5, it has six revolute joints and seven links.*

transformation matrix T_0^6 is computed by production of the transformation matrices of joints $j_i, \forall i = 1, ..., 6$, as follows:

$$T_0^6 = T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 \tag{6.2}$$

Robot gripper

A ROBOTIQ adaptive gripper with two fingers was used to hold a poly-brush in order to perform the sweeping experiment (Figure 6.19[5]). The gripper is capable to adapt to different geometry of the object by the designed mechanism. The working range of the gripper is 84 mm. A schema of the ROBOTIQ gripper is shown in Figure 6.20.

The adaptive gripper needs a 24 V power supply at 2 A. This gripper comes with a gripper controller box that can communicate with the Universal Robot's controller with an auxiliary power supply. The control box has a *Modbus TCP* communication interface, which allows to control the gripper finger position through Universal Robot's teach pendant. This interface

[5]Figure 6.19 and 6.20 are taken from manual of the *ROBOTIQ* 2-finger gripper.

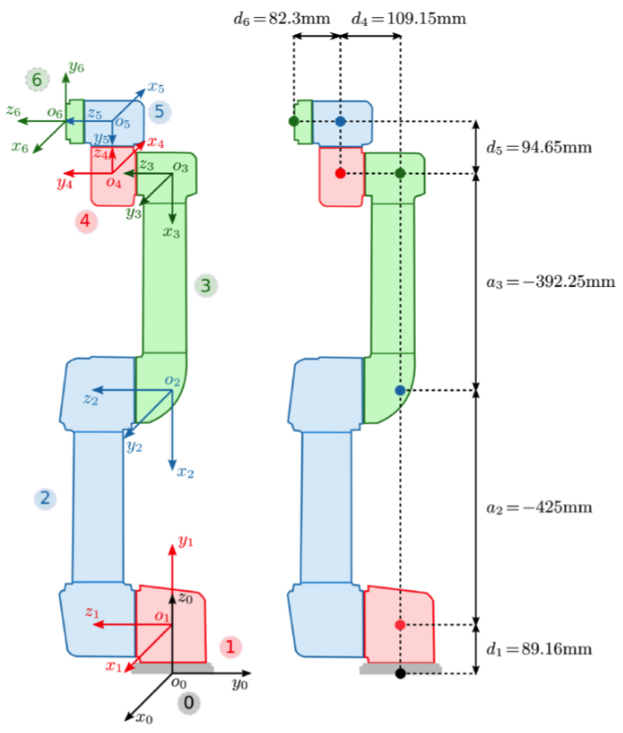


Figure 6.18: *Sketch of the UR5 and the considered coordinate frames at each joint according to Denavit-Hartenberg convention (left), and the dimension of each link (right).*

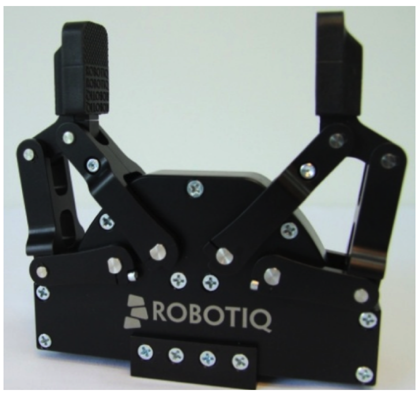


Figure 6.19: *ROBITIQ gripper used to hold the poly-brush.*

allows to control the position, speed and force of the gripper, and it gets information of gripper motor current and finger position.

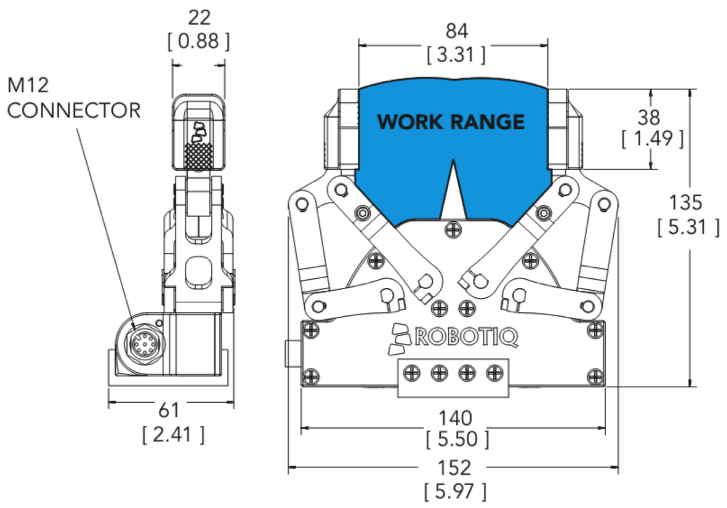


Figure 6.20: *schema of ROBOTIQ gripper. Unit of the provided dimensions is mm and the unit in [ ] are inch.*

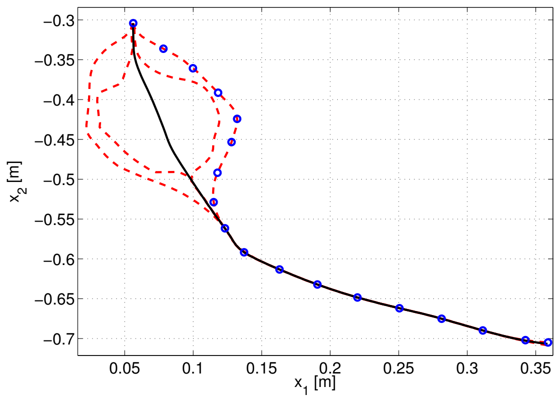
6.4.3 Sweeping data collection

In the past years, due to the development of robots that can safely perform different tasks out of the cage, including household tasks such as Roomba Vacuum Cleaning Robot, many studies have been conducted. These studies aim at making robots more intelligent and to enable them to learn a task from human demonstrations and adapt the learned task to a new environment. In this section, in the context of robot learning from demonstration, the task of sweeping is chosen to be performed by the UR5 robot.

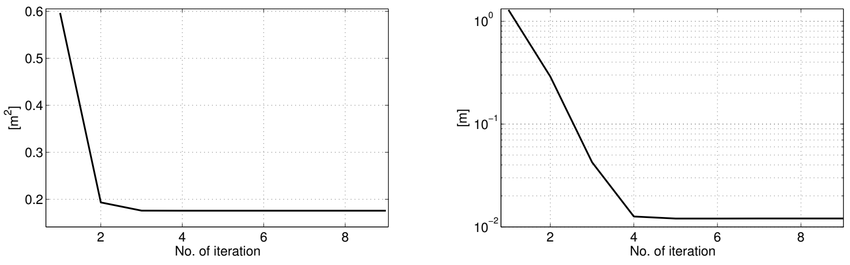
The teach pendant of UR5 has a teaching mode in which a user can pull the robot arm and move it to the desired position and orientation and save the pose of the end-effector. To provide a set of task demonstrations, the robot was moved in the teaching mode to perform the sweeping task and a set of waypoints[6] was recorded during every task demonstration. Then, the waypoints were played back by the robot and a set of data-points of the corresponding trajectory was collected.

In order to collect the data of the trajectories, the end-effector positions are obtained through a script written on Polyscope and sent over a TCP/IP connection to the counterpart script on the Personal Computer (PC). Although Polyscope can send messages over a TCP/IP to PC at 100 Hz, the time required to get the end-effector pose limits the number of data-points

[6]A waypoint is a recorded position and orientation of the coordinate frame attached to the tool center point of the UR5.



(a)



(b) (c)

Figure 6.21: *(a) Collected data during sweeping task demonstrations. The user moved the arm by pulling the end-effector while another person holding the teaching bottom and a set of waypoints (blue solid circle) are recorded. The recorded waypoint were then played back by the robot and the corresponding points of the resulted trajectory is collected (black dashed line). The computed generalized path by the Mean-Path algorithm across demonstrations (red line); (b) Residual of the data in the Distance-Space* $(tr(\Sigma))$*; (c) error of the Mean-Path computation* ε*.*

collected during each experiment. We set the speed of task execution at 20% of its maximum speed to collect necessary data points. Eventually, for the sweeping task with 15 recorded waypoints, 90 data-points were collected. The task of sweeping the green cube into the black dustpan, shown in Figure 6.14, with two types of obstacles, a cup and a marker, were demonstrated four times. We use the tool tip of the robot to pointing at the position of every obstacle to collect the necessary information of each obstacle position. This can be also done by a calibrated camera, e.g. a Kinct. The collected data-points of the different sweeping task and the corresponding waypoints are shown in Figure 6.21(a).

Table 6.4: *The evaluation metric values (e_1 $[mm^2]$ and e_3 $[mm]$) of the computed Mean-Path by using different initial Reference-Path $\zeta_r = \zeta_j,\ j = 1, ..., 4$, where ζ_js are demonstrated paths.*

$\zeta_r = j_{th}$ demonstrated path	$e_1[m^2]$	e_3 $[mm]$
j=1	1603	5.36
j=2	1554	7.44
j=3	1548	6.44
j=4	1548	6.32
mean/variance of error	1563/710	6.39/0.7236

6.4.4 Learning the model

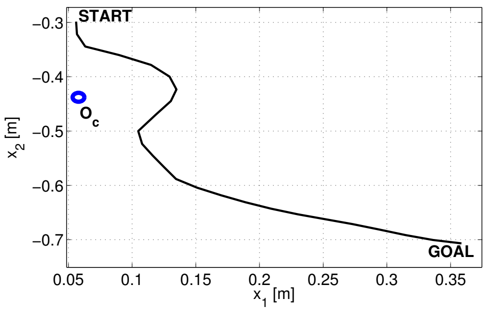
Robot learning from demonstration is a means of programming a robot to perform a new task. One way to enhance this method of programming is by partially learning the model of the task from demonstrations and explicitly defining the remaining part. In the sweeping example, the model is simplified by using some prior knowledge, e.g. the brush must be in contact with the surface of the table $x_3 = 0.192[m]$ and it must be orthogonal to the corresponding trajectory, i.e. $\alpha = 0,\ \beta = 0$ and γ can be defined such that the tool tip is orthogonal to the corresponding path [7]. Therefore, the task can be considered a 2-DOF motion.

Using the Mean-Path algorithm a generalized path across different task demonstrations is computed. In Figure 6.21 the obtained generalized path and the evolution of the error and residual of the data based on the computed generalized path during Mean-Path algorithm computation are shown. The errors of computing a generalized path by the Mean-Path algorithm, i.e. $e_1 = tr(\Sigma)$ and $e_3 = \sum_{i=1}^{m} \left\| \sum_{j=1}^{n} \overrightarrow{d}_{j,i} \right\|$, with different Reference-Path (initial value) are reported in Table 6.4. The generalized path can be considered the required reference path to sweep the green object to the dustpan without the presence of an obstacle.

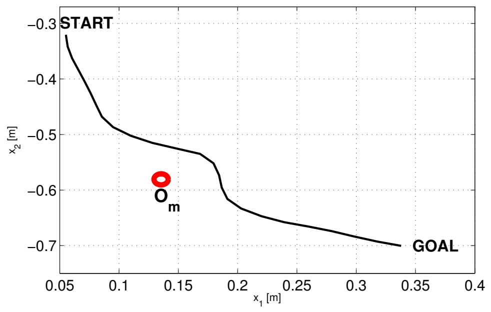
If the generalized path ζ_{Mp} computed by the Mean-Path algorithm is used to reproduce the task of sweeping in the presence of the marker and the cup, the mean square error $MSE_{Mp} = \frac{1}{nm} \sum_{d=1}^{n} \sum_{i=1}^{m} (\zeta_{d,i} - \zeta_{Mp,i})^2$, is equivalent to $2.8181[mm^2]$ and $5.8572[mm^2]$, respectively. Although the generalized path might be used to reproduce the task, it would not be feasible in the presence of the obstacle.

We use inverse optimal control problem (equation (5.5)) to recover underlying reward function of a demonstration. We consider the problem of

[7] $x = [x_1, x_2, x_3]$ is the position of the tool center point. α, β and γ are the corresponding angle of transformation matrix that transforms the reference coordinate frame to the local coordinate frame attached to the tool center point of the UR5. For example, α, β and γ are the rotation angle around x_1, x_2 and x_3 in Figure 6.23, respectively.



(a)



(b)

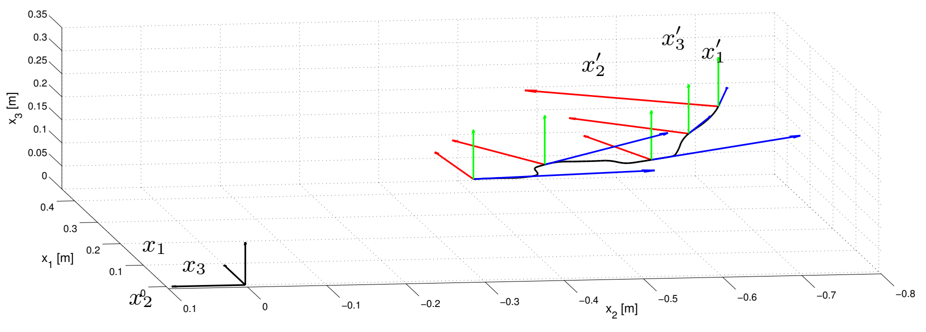
Figure 6.22: *The reproduced paths by the obtained reward function for two obstacles, the cup (a) and the marker (b).*

computing the necessary path to sweep the green cub into the dustpan in the presence of an obstacle to be an optimal control problem. In order to adapt the task model to a new environment based on the different positions of the cup or the marker a reward function is built using equation (5.4). whose parameters are as follows: $Q = 20$, $R_c = diag([\frac{1}{15.4} \quad \frac{1}{1000}])$ and $R_m = diag([\frac{1}{20.08} \quad \frac{1}{880}])$. The obtained reward function is then used to generate the required path to sweep the green cube to the dustpan and avoid colliding with the marker and the cup (Figure 6.22).

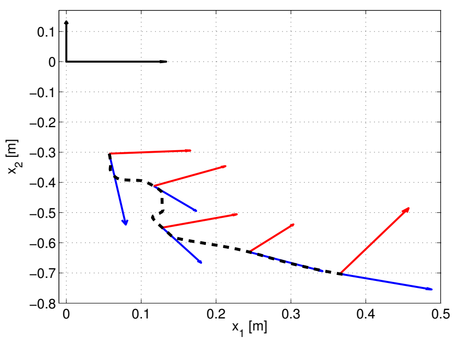
The corresponding mean square error of the produced path,

$$MSE_R = \frac{1}{nm}\sum_{d=1}^{n}\sum_{i=1}^{m}(\zeta_{d,i} - \zeta_{R,i})^2$$

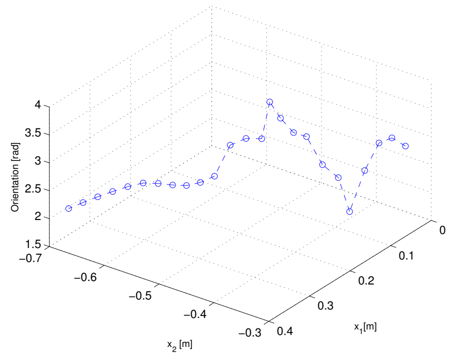
where n and m are the number of the number of demonstrated paths and the collected points of every path, are 0.2177 $[mm^2]$ and 0.2074 $[mm^2]$, respectively. Furthermore, we compute the improvement value, $Pr =$



(a)



(b)



(c)

Figure 6.23: *(a) A sample demonstration and the fixed local coordinate frame to the tool tip along the trajectory; (b) top view of the local coordinate frame along the demonstration; (c) the computed position and orientation of the tool along the demonstration.*

$\frac{MSE_{Mp}-MSE_R}{MSE_{Mp}} \times 100\%$. For the marker and the cup, the improvement val-

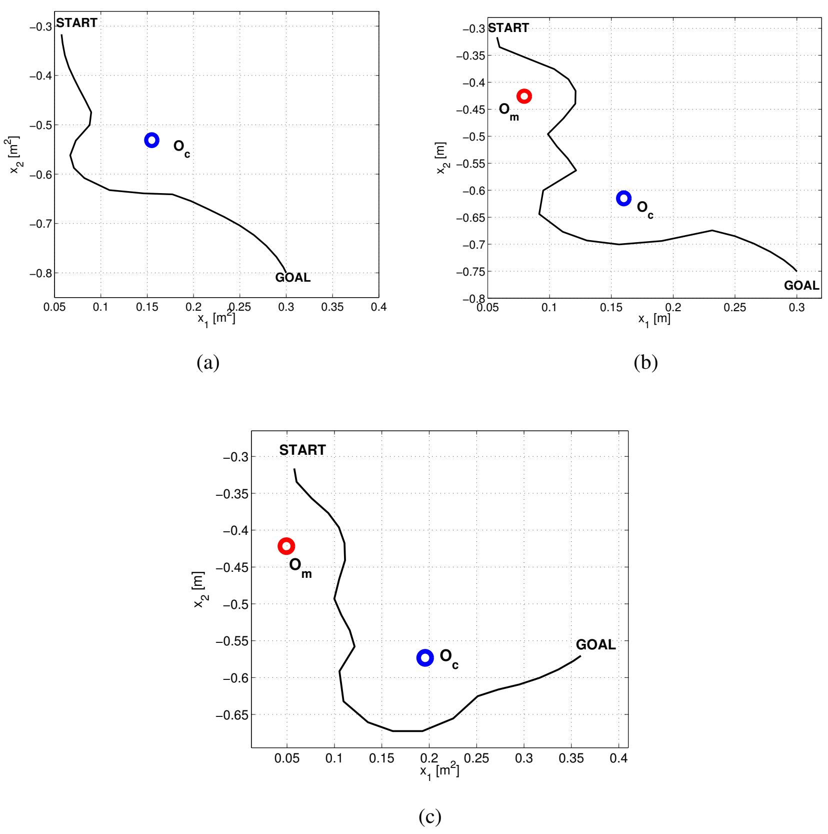


Figure 6.24: *The obtained model is used to reproduce required paths to accomplish the sweeping task in different scenarios. The computed path corresponding to scenario 1, 2 and 3 are shown in (a), (b) and (c).*

ues with respect to the use of the generalized path are 92.7% and 96.46%, respectively.

To perform the sweeping task we have to send a sequence of poses to the UR5.The pose values of the end-effector consists of three position values and three orientation values. The standard orientation values that UR5 accepts are axis angles representation. Hence, first the rotation matrix of the end-effector with respect to the reference frame at the base of the UR5 is computed (Figure 6.23). Then, the computed rotation matrix must be converted to an axis angle representation. The axis angle parameterizes a rotation with a rotation axis ω and an angle of rotation θ.

$$\begin{aligned} \theta &= arccos\left(\frac{trace(R)-1}{2}\right) \\ \omega &= \frac{1}{2sin(\theta)}\begin{pmatrix} R(3,2)-R(2,3) \\ R(1,3)-R(3,1) \\ R(2,1)-R(1,2) \end{pmatrix} \end{aligned} \tag{6.3}$$

6.4.5 Task reproduction

According to the proposed workflow in Chapter 2, three different levels of robot learning from demonstration are used in this experiment. First, the generalized path has been computed from a set of task demonstrations. This generalized path would be the best path that the robot could follow if there was no information available from the scene, i.e. in the case that there is no available information about the position of the obstacles. We adapt the generalized path to every environment by employing robot LFD at the emulation and imitation level of the workflow. This improves the solution by enabling a robot to generalize the task to both a new position of the dustpan and an environment with an obstacle. To do so, the response of the demonstrator to different obstacles is encoded into an appropriate reward function. This reward function can be used to reproduce a suitable path for a different position of the obstacles in the scene (see Figure 6.22(b)).
Furthermore, Dynamic Movement Primitives is used to generate a path to a new goal point. A new reward function based on the generated path by DMPs and the computed adaptation component has been built using the parameters of the reward function Q and R. The optimal solution to of the reward function provides the required path to perform the sweeping task with the new position of the dustpan and of the obstacles (see Figure 6.24).

The proposed workflow of robot learning from demonstration has been used to solve the problem of obstacle avoidance in different examples. The proposed method succeeds to replicate a task in a slightly different environment with good precision. One of the main advantages of the proposed method is that it solves the problem with a continuous state space. Next, the MPC formulation of the solution allows the robot to cope with some local minima of the reward (objective) function.

On the other hand, the proposed method has some limitations. First, the process of finding the parameters of the reward function is iterative and involves the problem of finding an optimal solution with the estimated reward function at each iteration. This is a computationally expensive process.

Next, a time horizon has to be determined for the MPC formulation. In order to deal with the local minima of the reward function the time horizon should be big enough. On the other hand, a very big time horizon may result in an unstable solution and a discontinuous path generation. In this work, we have chosen the sweeping to be an episodic task with one hundred time steps. We have experimentally selected a time horizon $4 \leq H \leq 12$ in the MPC formulation (equation (5.6)).

In the future, the problem of uncertainty of the measured position of an obstacle in a scenario with moving obstacles can be studied. This allows for solving a more realistic problem of robot learning from demonstration in which the objects may not be static and the sensory information may be noisy. Furthermore, a component with attractive reward values can be studied to adapt the reference path according to some target points in the scene. For instance, in the sweeping example the position of the cube to be swept can be considered in the reward function. Hence, the resulted reward function will adapt to the new position of the target.

6.5 Conclusion

In this chapter the proposed methodology has been applied to a pick and place task with a da Vinci robot and a sweeping example with UR5 robot. Robot learning from demonstration at three different levels, i.e. mimicking, imitation and emulation, has been used to build a model of the tasks from a set of task demonstrations.

The collected data set with by Vinci was divided into a training set and a test set. The model obtained from the training set is used to generate a path required to perform the task in one of the scenes of the test set. The reproduction error and precision of the generated paths corresponding to paths of the test set illustrate the efficiency of the model learned from the training set to generate the required path in a new scene.

In the sweeping example the task with two types of obstacles have been demonstrated separately. The model obtained for each type of obstacle shows how a user can teach a robot a desired response to an obstacle, which may be highly demanded in different applications. The obtained obstacle avoidance components of two types of obstacle are then used and efficiently produced the required path to perform the task in the presence of both obstacles. This shows how different components of the proposed model can be learned independently and transferred across different scenarios. Furthermore, dynamic movement primitives has been used to generate a path to a new goal point. The reward function combined the obstacle avoidance component with the generated path with DMPs. Finally the robot succeeds to perform the task with a new position and of the obstacle and of the dust-pan by following the optimal path with maximum collected reward.

Appendix

7.1 Proof of the convergence of Mean-Path algorithm

In this section the Mean-Path algorithm is briefly presented and the convergence proof of the algorithm is discussed. The Mean-Path algorithm aims at computing a path from a data set of 2-D paths capturing the most variability of the data set. A path that captures the most variability of a data set of 2-D paths has the minimum sum of the squared distances to all the paths within the data set.

Let $\zeta_r = \{x_1^r, ..., x_M^r\}$ be a 2-D path taken as a reference for distance computation, called Reference-Path. Further, x_i^r are called Reference-Points. A set of line segments, denoted by Δ_{ζ_j}, connecting two consecutive points of a path, are defined as follows,

$$\Delta_{\zeta_j} = \{\delta_{j,1}, \delta_{j,2}, ..., \delta_{j,m-1}\},$$
$$\delta_{j,i} = \overline{x_{i-1}^j x_i^j},$$

where $\delta_{j,i}$ is a line segment with initial point x_{i-1}^j and terminal point x_i^j. Because a continuous path x^j is not available, a piecewise linear approximation of the sequence of points Δ_{ζ_j} are used wherever information of continuous path is needed. An intersection point of the normal line to the Reference-Path at x_i^r and another path ζ_j is used to compute the distance between x_i^r and another path ζ_j.

We define residual of the data set of 2-D paths based on a Reference-Path as the sum of the squared distances between x_i^rs and ζ_j $\forall\, i = 1, ..., M$. Hence, the Mean-Path, capturing the most variability of the dataset, is for-

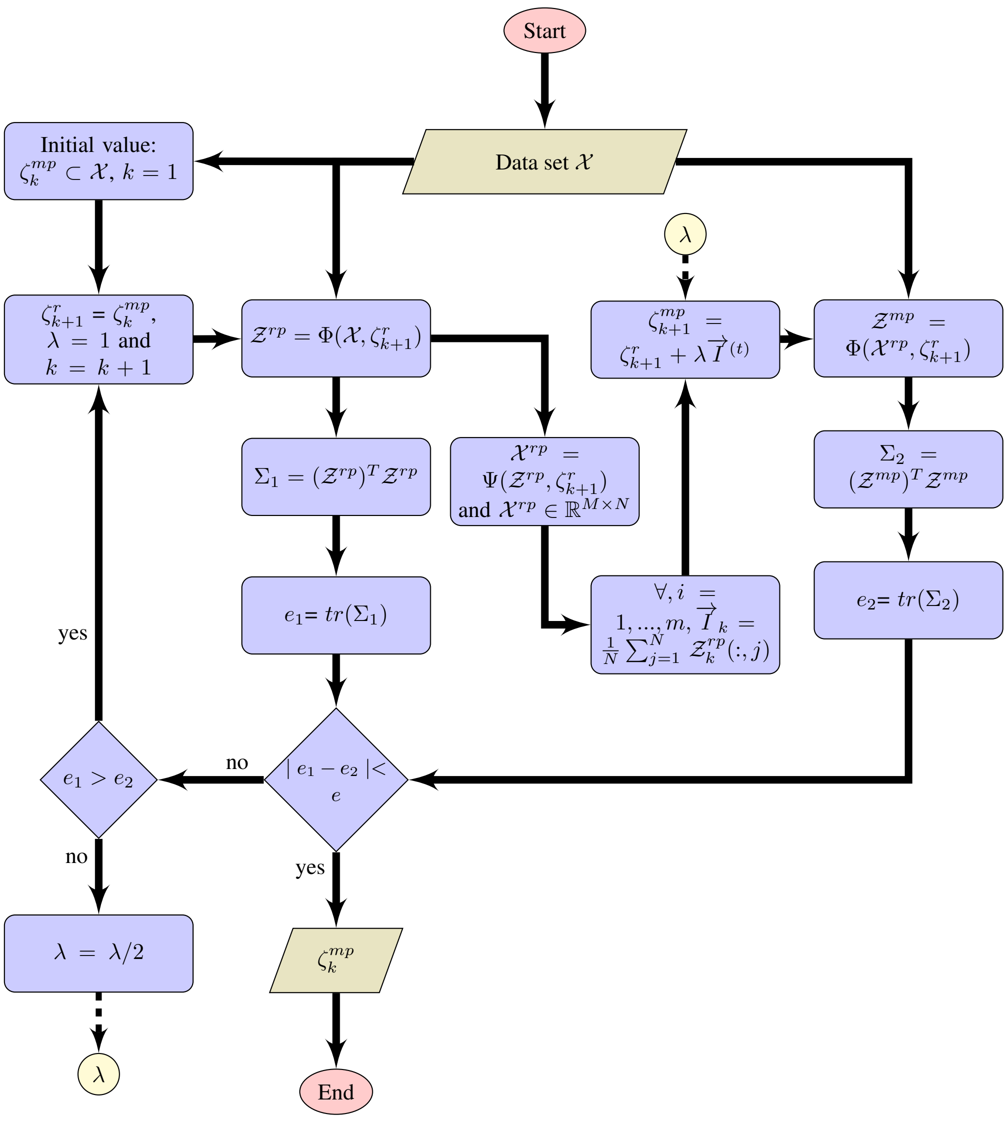

Start
Data set $\mathcal{X}$
Initial value: $\zeta_k^{mp} \subset \mathcal{X}$, $k = 1$
$\zeta_{k+1}^{r} = \zeta_k^{mp}$, $\lambda = 1$ and $k = k + 1$
$\mathcal{Z}^{rp} = \Phi(\mathcal{X}, \zeta_{k+1}^{r})$
λ
$\zeta_{k+1}^{mp} = \zeta_{k+1}^{r} + \lambda \overrightarrow{I}^{(t)}$
$\mathcal{Z}^{mp} = \Phi(\mathcal{X}^{rp}, \zeta_{k+1}^{r})$
$\Sigma_1 = (\mathcal{Z}^{rp})^T \mathcal{Z}^{rp}$
$\mathcal{X}^{rp} = \Psi(\mathcal{Z}^{rp}, \zeta_{k+1}^{r})$ and $\mathcal{X}^{rp} \in \mathbb{R}^{M \times N}$
$\Sigma_2 = (\mathcal{Z}^{mp})^T \mathcal{Z}^{mp}$
$e_1 = tr(\Sigma_1)$
$\forall, i = 1, ..., m, \overrightarrow{I}_k = \frac{1}{N} \sum_{j=1}^{N} \mathcal{Z}_k^{rp}(:, j)$
$e_2 = tr(\Sigma_2)$
yes
$e_1 > e_2$
no
$| e_1 - e_2 | < e$
no
yes
$\lambda = \lambda/2$
ζ_k^{mp}
λ
End

Figure 7.1: *This flow chart shows the Mean-Path algorithm. A set of paths of a task performances constitutes our data set. The iterative algorithm at each computation step updates the Reference-Path along a set of lines normal to the Reference-Path.*

malized in equation (G.1) as the path resulting in the minimum residual of the data set.

$$\begin{aligned}
\zeta^r_{k+1} = \quad & \underset{\zeta^{(t)}_r \in \mathbb{R}^{m\times N}}{\text{argmin}} \; tr(\Sigma_k) \\
& \text{s.t.} \quad \mathcal{Z} = \Phi(\mathcal{X}, \zeta^r_k) \\
& \qquad\;\; \Sigma = \mathbb{E}[\mathcal{Z}^T \mathcal{Z}]
\end{aligned} \tag{G.1}$$

where

$$\mathcal{Z} = \Phi^{(r)}(\mathcal{X}) := \Phi(\mathcal{X}, \zeta^r_k),$$

and $\mathcal{Z} = \{V^{r,j}_i, \forall\ i = 1, ..., M \text{ and } j = 1, ..., N\}$ is a set of corresponding data points of original dataset, $\mathcal{X} = \{\zeta_1, ..., \zeta_N\}$, in Distance-Space based on the considered Reference-Path.

A similarity measure between a Reference-Path and another path in the set, denoted by $V^{r,j}_i$, at each Reference-Point $x^r_i\ \forall\ i = 1, ..., M$, is a vector whose absolute value is a distance between x^r_i and Δ_{ζ_j}, $\forall\ j = 1, ..., n$, as follows:

$$V^{r,j}_i := \overrightarrow{x^r_i p^{r,j}_i},$$
$$\left|V^{r,j}_i\right| = \left\|\, x^r_i - p^{r,j}_i \right\|,$$

where $p^{r,j}_i$ is an intersection of Δ_{ζ_j} and $L_{i,k}$, see Figure 7.2,

$$p^{r,j}_i = \Delta_{\zeta_j} \cap L_{i,k}.$$

A description of the iterative algorithm is reported in Algorithm 2.

7.1.1 Preliminary

For a given set of task demonstrations some definitions are presented in the following in order to formalize a geometrical model of a task, called Mean-Path.

Definition 1. *Given two paths ζ^A and ζ^B, a directed line segment $\delta^A_{i_A}$ ($i_A = 1, \ldots, m_A - 1$) on ζ^A is associated with a directed line segment $\delta^B_{i_B}$ ($i_B = 1, \ldots, m_B - 1$) on ζ^B if $\delta^A_{i_A} \cdot \delta^B_{i_B} > 0$ and the line perpendicular to $\delta^A_{i_A}$ intersects $\delta^B_{i_B}$ (Fig. 7.2).*

Definition 2. *A necessary and sufficient condition for two paths ζ^A and ζ^B to be globally associated is that each directed line segment of path ζ^A ($\delta^A_{i_A}$, $i_A = 1, \ldots, m_A - 1$) is associated with a directed line segment of path ζ^B ($\delta^B_{i_B}$, $i_B = 1, \ldots, m_B - 1$) and vice versa.*

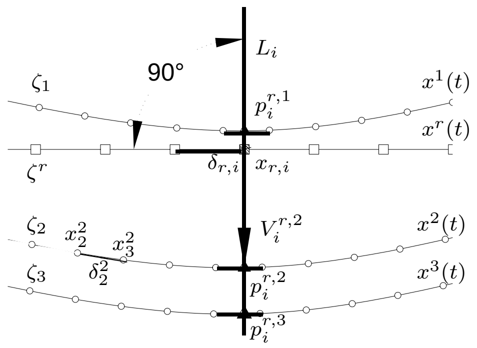


Figure 7.2: *A set of four 2-D curves (continuous ones $x^i,\ i = 1, 2, 3$, and their collected data points ζ, shown by circle signs), Reference-Path x^r and its data points ζ^r shown with square signs. The line segment $\delta_2^2 = \overline{x_2^2, x_3^2}$ linearly approximates the continuous path x^2 between x_2^2 and x_3^2. The line L_i is perpendicular to $\delta_{r,i}$. Intersections of L_i and each Δ_{ζ_j}, denoted by $p_i^{r,j}, \forall\, j = 1, 2, 3$ (marked with triangle sings). Vector $V_i^{r,2}$ determines the position of $p_i^{r,2}$ relative to x_i^r.*

Algorithm 2 Mean-Path algorithm, $\bar{\Delta}e$ is a chosen small threshold

```
 1: procedure MEAN-PATH($\mathcal{X} = \{\zeta_1, ..., \zeta_N\}$)
 2:     $k = 1$ , $\zeta_k^r \leftarrow \zeta_1$ , $\Delta e = 1$ and $\lambda = 1$
 3:     while $\Delta e > \bar{\Delta}e$ do
 4:         $\mathcal{Z} = \Phi(\mathcal{X}, \zeta_k^r)$
            $\Sigma_k = \mathcal{Z}^T \mathcal{Z}$,
            $e_k \leftarrow tr(\Sigma_k)$,
 5:         $z_{i,k}^d \leftarrow \mathbb{E}(z_{i,k}^j \in \mathcal{Z}, \forall j = 1, ..., N)$
            $z_{i,k}^c = \Psi(z_{i,k}^d, \zeta^r)$
            $\forall\, i = 1, ..., M$
 6:         $I_{i,k} \leftarrow \lambda(x_{i,k}^r - z_{i,k}^c)\ \forall i = 1, ..., M$
 7:         $\zeta_{k+1}^r \leftarrow \zeta_k^r + \overrightarrow{I}_k$
 8:         $\mathcal{Z} = \Phi(\mathcal{X}, \zeta_{k+1}^r)$,
            $\Sigma_{k+1} = \mathcal{Z}^T \mathcal{Z}$,
            $e_{k+1} \leftarrow tr(\Sigma_{k+1})$
 9:         if $e_{k+1} > e_k$ then
10:             $\lambda \leftarrow \frac{\lambda}{2}$
11:         else
12:             $\Delta e \leftarrow \|e_{k+1} - e_k\|$ , $k = k + 1$ and $\lambda \leftarrow 1$
13:         end if
14:     end while
15:     return $\zeta_k^r$
16: end procedure
```

A corresponding definition of the globally associated continuous paths $x^A(t)$ and $x^B(t)$ can be defined in the same way.

Definition 3. *Considering two differentiable and globally associated paths $x^A(t)$ and $x^B(t)$, the Mean-Path is a differentiable path whose sum of squared distances from ζ^A and ζ^B is minimum.*

It is worth mentioning that the distances are measured along the perpendicular lines to the Mean-Path.

Definition 4. *Given a nominal path of a task $\overline{x}(t)$, a task demonstration $x(t)$ is called slender demonstration if and only if $x(t)$ is differentiable and the distance of point t on $\overline{x}(t)$ to $x(t)$ ($d_{\overline{x},x}(t), \quad t \in \left[0, \overline{t}\right]$) is less than the radius of curvature of $\overline{x}(t)$ at that point.*

Theorem 1. *For two differentiable and globally associated paths $x^A(t)$ and $x^B(t)$, which are slender demonstrations of a task with the same initial point, the Mean-Path exists and is unique.*

Proof : see 7.1.2

In fact, the Mean-Path ζ^{mp} is the one minimizing the cumulative sum of squared distances to all the paths within the set. Therefore, given a set of paths (ζ) and a Reference-Path (ζ^r), the Mean-Path ($\bar{\zeta}$) is computed, at each computation step (k) in equation (G.1).

7.1.2 Proof of Convergence

In this section the proof of convergence of Mean-Path algorithm, and the proof of theorem 1 are discussed. Assume that two demonstrations of a task, differentiable and globally associated paths x^1 and x^2, are available for Mean-Path computation. They are represented by two discrete paths, i.e. two sets of sample points, ζ^1 and ζ^2, where

$$\zeta_j = \{x_1^j, x_2^j, ..., , x_N^j\} \ \forall \ j = 1, 2.$$

Definition 5. *A point belongs to a realization of the Mean-Path (shown in Fig. 7.3, 7.4, and 7.5) if and only if*

$$\|x_{i+1}^m - x^{m,b}\| = \|x_{i+1}^m - x^{m,s}\|, \qquad i = 1, 2, ..., N-1 \tag{G.2}$$

where $x^{m,s}$ and $x^{m,b}$ are the intersections of the line perpendicular to $\overline{x_i^m x_{i+1}^m}$ and paths ζ_1 and ζ_2, respectively.

Consider a point x_i^m on the Mean-Path, fixed by a boundary condition[8] or through the algorithm. Let assume that we need to compute only the

[8] If two paths have the same initial point, the initial point will be the first point of the Mean-Path (x_1^m). Otherwise, the line connecting two boundary (ending or beginning) points of two paths can be considered as the boundary condition and consequently x_1^m can be computed.

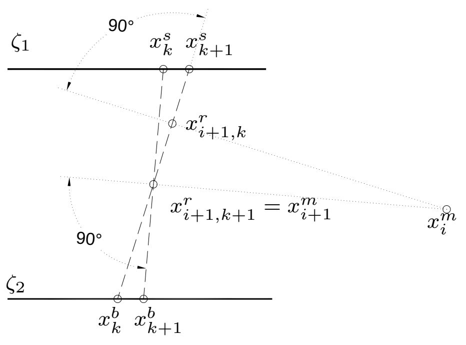


Figure 7.3: *Two line segments of parallel paths.*

next point x_{i+1}^m (eq. (G.2)) on the realization of the Mean-Path through the proposed algorithm. Hence, given a reference line segment $\overline{x_i^m x_{i+1,k}^r}$ by applying Mean-Path algorithm a point on a realization of the Mean-Path (x_{i+1}^m) can be computed, iteratively, as follows

$$x_{i+1,k+1}^r = V_{i+1,k}^c + x_{i+1,k}^r, \tag{G.3}$$

where

$$\begin{aligned} V_{i+1,k}^c &= \lambda(x_{i+1,k}^r - x_{i+1,k}^c) \\ &0 < \lambda \leq 1 \end{aligned} \tag{G.4}$$

and

$$x_{i+1,k}^c = \frac{x_k^s + x_k^b}{2}. \tag{G.5}$$

In eq. (G.3) x_k^s and x_k^b are intersections of the line $\overleftrightarrow{x_k^s x_k^b}$ perpendicular to the reference line segment $\overline{x_i^m x_{i+1,k}^r}$ by ζ_1 and ζ_2, respectively, and each computation step is denoted by t.

Lemma 1. *Considering two parallel path segments ζ_1 and ζ_2 in Fig. 7.3, and crossing line $\overleftrightarrow{x_k^b x_k^s}$ perpendicular to the given reference line segment $\overline{x_i^m x_{i+1,k}^r}$, a point on a realization of the Mean-Path (x_{i+1}^m) is computed by* (G.3) *in one step with $\lambda = 1$.*

Proof. The mean value of x_k^s and x_k^b, the intersection points of the crossing line and ζ_1 and ζ_2, is identical to the mean value of the intersection points of any other line crossing ζ_1 and ζ_2.

$$x^m_{i+1} = \frac{x^s_k + x^b_k}{2},$$
$$x^m_{i+1} = \frac{x^s_{k+1} + x^b_{k+1}}{2}.$$

□

Lemma 2. *Considering two obtuse path segments ζ_1 and ζ_2 in Fig. 7.4, and crossing line $\overleftrightarrow{x^b_k x^s_k}$ perpendicular to the given reference line segment $\overline{x^m_i x^r_{i+1,k}}$, a point computed by* (G.3) *at each computation step is closer to the point on a realization of the Mean-Path with $\lambda = 1$.*

Proof. Assume point x^m_{i+1} is given on Mean-Path, which satisfies eq. (G.2), therefore

$$\|x^m_{i+1} - x^{m,b}\| = \|x^m_{i+1} - x^{m,s}\|.$$

If the line $\overleftrightarrow{x^{m,b}x^{m,s}}$ perpendicular to $\overline{x^m_i x^m_{i+1}}$ rotates around x^m_{i+1} (e.g. line $\overleftrightarrow{x^s_k x^b_k}$), the mean value ($x^r_{i+1,k+1}$) of the intersection points position (x^s_k and x^b_k, which are the intersections of the line $\overleftrightarrow{x^s_k x^b_k}$ by the paths ζ_1 and ζ_2) can be calculated through eq. (G.3), as well. On the other hand, considering line segment $\overline{x^m_i x^r_{i+1,k}}$ perpendicular to $\overleftrightarrow{x^s_k x^b_k}$ and passing through x^m_i, it can be confirmed that

$$\begin{aligned} x^r_{i+1,k+1} \in \overline{x^m_{i+1} x^r_{i+1,k}} \rightarrow \\ \|x^m_{i+1} - x^r_{i+1,k+1}\| < \|x^m_{i+1} - x^r_{i+1,k}\| \end{aligned} \tag{G.6}$$

Hence, given a reference line segment $\overline{x^m_i x^r_{i+1,k}}$ and computing a point on a realization of the Mean-Path using eq. (G.3) results in $x^r_{i+1,k+1}$ such that

$$\theta^{(t+1)} < \theta^{(t)},$$

where θ denotes the angle of the line segment of the Reference-Path and the Mean-Path. Therefore, the updated value $x^r_{i+1,k+1}$ converges to x^m_{i+1}. □

Lemma 3. *Considering two acute path segments ζ_1 and ζ_2 in Fig. 7.5, and crossing line $\overleftrightarrow{x^b_k x^s_k}$ perpendicular to the given reference line segment $\overline{x^m_i x^r_{i+1,k}}$, a point computed by* (G.3) *at each computation step is closer to the point on a realization of the Mean-Path if*

$$V^c_{i+1,k} \cdot \overrightarrow{x^r_{i,k+1} x^c_{i+1,k+1}} > 0.$$

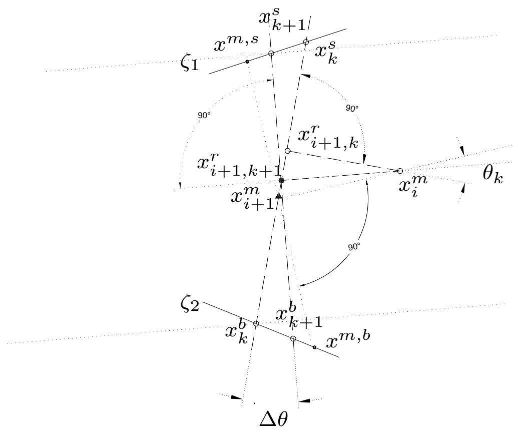


Figure 7.4: *Two line segments of obtuse paths.*

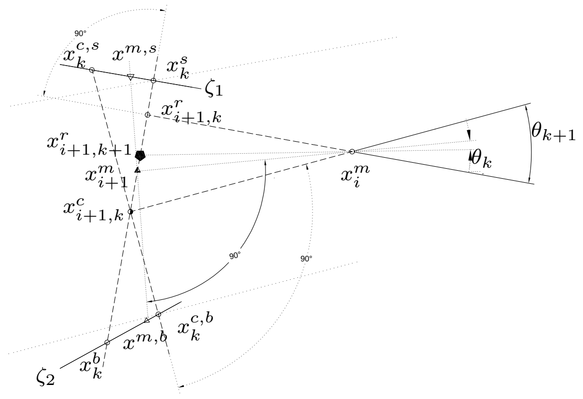


Figure 7.5: *Two line segments of acute paths.*

Proof. Consider a Reference-Point $x^r_{i+1,k}$ such that

$$\|x^s_k - x^r_{i+1,k}\| < \|x^b_k - x^r_{i+1,k}\|,$$

where x^s_k and x^b_k are the intersections of the line perpendicular to $\overline{x^m_i x^r_{i+1,k}}$, i.e. $\overleftrightarrow{x^s_k x^b_k}$, by ζ_1 and ζ_2 in Fig. 7.5. The mean value of the intersection points ($x^c_{i+1,k}$) is computed using eq. (G.5).

Eventually, with an argument similar to the one used in Lemma 2, it can be shown that

$$\|x^r_{i+1,k} - x^c_{i+1,k}\| > \|x^r_{i+1,k} - x^m_{i+1}\| \tag{G.7}$$

and

$$\|x^{c,s} - x^c_{i+1,k}\| > \|x^{c,b}_k - x^c_{i+1,k}\|, \tag{G.8}$$

where $x^{c,s}$ and $x^{c,b}$ are intersections of the paths by the line perpendicular to $\overline{x^m_i x^c_{i+1,k}}$. Consequently, in order to guarantee the convergence of the algorithm to a point on a realization of the Mean-Path, $x^r_{i+1,k+1}$ is computed such that

$$x^r_{i+1,k+1} \in \overline{x^r_{i+1,k} x^m_{i+1}},$$

From equation (G.5) and (G.8), for different values of λ it follows that

$$\lambda << 1 \rightarrow x^r_{i+1,k+1} \in \overline{x^r_{i+1,k} x^m_{i+1}} \tag{G.9}$$

$$\rightarrow V^c_{i+1,k} \cdot \overrightarrow{x^r_{i+1,k+1} x^c_{i+1,k+1}} > 0, \tag{G.10}$$

and

$$\lambda = 1 \rightarrow x^r_{i+1,k+1} \in \overline{x^c_{i+1,k} x^m_{i+1}} \tag{G.11}$$

$$\rightarrow V^c_{i+1,k} \cdot \overrightarrow{x^r_{i+1,k+1} x^c_{i+1,k+1}} < 0. \tag{G.12}$$

It can be also shown that

$$V^c_{i+1,k} \cdot \overrightarrow{x^r_{i,k+1} x^c_{i+1,k+1}} > 0 \rightarrow \ \theta^{(t)} > \theta^{(t+1)}. \tag{G.13}$$

Therefore, to guarantee the convergence of the algorithm λ must be selected in such a way that

$$V^c_{i+1,k} \cdot \overrightarrow{x^r_{i,k+1} x^c_{i+1,k+1}} > 0.$$

□

Definition 6. *The radius of curvature R_i is defined here as as follows,*

$$R_i = \min(d_{i+1}, d_{i-1})$$
$$i = 1, 2, ..., M \tag{G.14}$$

where

$$d_{i+1} = L_{i,k} \cap L_{i+1,k}$$
$$d_{i-1} = L_{i,k} \cap L_{i-1,k}$$

If there is no intersection d_i is considered infinity ($d_0 = d_{m+1} = \infty$).

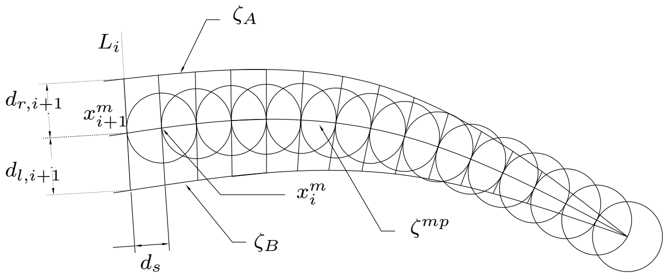


Figure 7.6: *A pair of associated paths (ζ^A and ζ^B) and their Mean-Path. $d_{r,i+1}$ and $d_{l,i+1}$ are distances between the Mean-Path and the Objective-Paths. A tangent line to the circle centered at x^m_i is selected such that $d_{r,i+1}$ and $d_{l,i+1}$ are equal.*

Lemma 4. *For a pair of slender task demonstrations (introduced in definition 3), given a point x^m_i on the Mean-Path (eq. (G.5)) and a small step size r, point x^m_{i+1} on a circle with radius r (Fig. 7.6) exists whose distances to both demonstrations along the tangent line to the circle are identical.*

Proof. The demonstrated paths can be considered linear in the neighbors (with small step size r) of the intersection points of the line tangent to circle and the paths (ζ^A and ζ^B). Therefore, in three cases including parallel, obtuse and acute paths, with the same argument used in Lemma 1, 2 and 3 it can be shown that there is only one point on a circle with radius r whose distances to both demonstrations along the tangent line to the circle are identical. □

As it is discussed in Lemma 1, 2 and 3, given any Reference-Path the computed Mean-Path at each iteration remains in the convex hull of the set of paths. In the case that the chosen Reference-Path is parallel to the Mean-Path[9], we can compute the Mean-Path in one computation step, independent of the distance between the two paths. Moreover, in the case that the Reference-Path is not parallel to the Mean-Path, the algorithm, indeed, reduces the distances to the Mean-Path at each iteration. In fact, after each iteration not only the distances reduce, but also the slopes of the resulting path are modified in such a way that they are closer to the ones of the Mean-Path.

[9]Two paths are parallel if and only if every line perpendicular to the one path will be perpendicular to another path as well.

Proof of Theorem 1

Proof. For a given pair of globally associated paths, which are slender demonstrations of a task, using Lemma 4 while r tends to zero results in existence of a differentiable Mean-Path for the pair of paths. $\square$

Proof of convergence of Algorithm 2

Proof. Given a boundary crossing line, the first point on the Mean-Path is computed using eq. (G.3) with $\lambda = 1$. Moreover, considering a differentiable Reference-Path, within the convex hull of the demonstrated paths, and updating all the points of the Reference-Path using Algorithm 2, if a point of the Reference-Path ($P_{r,i}$) converges to the one on the realization of the Mean-Path (eq. (G.2)), the next point of the Reference-Path ($P_{r,i+1}$) will be close to the next point on the realization of the Mean-Path. On the other hand, since the perpendicular line at each step of computation deviates insignificantly with respect to the one in the last computation step, the paths crossed by the line can be considered linear in the neighbors of the intersections. Consequently, convergence of Mean-Path algorithm is sequentially proved through the convergence of the algorithm for every single point, which has been proved in Lemma 1, 2 and 3. $\square$

Conclusion and future works

Robots are used to perform many repetitive and precise tasks. However, the programming time and cost of a robot restricts the use of robots, especially in non-production-line uses. To tackle this issue and in order, although robot programming by demonstration has been introduced by which a robot can learn to perform a task from demonstrations, still a major concern is how a robot can generalize task demonstrations across different conditions.

Based on the analogy between robot learning from demonstrations and human learning from observationwe propose a multi-layer approach to robot learning from demonstration, including mimicking, imitation, and emulation. This approach enables a robot to learn a model to perform a task from noisy demonstrations and to generalize it to a new start and goal point as well as to different environments. We, further, classify the existing methods of robot learning from demonstrations, such as Gaussian mixture model/Gaussian mixture regression (GMM/GMR), dynamic movement primitives, and inverse optimal control according to the different layer of robot learning from demonstrations.

We demonstrate how the proposed approach applies to a practical example of sweeping rubbish into a dustpan.

We further extend the proposed approach to enable a robot to reproduce the learned task in a dynamic environment with moving obstacles using a Kalman filter to obtain a prediction of the obstacle position in a predication horizon. However, this is a very recent result which was not available during preparation of this thesis.

Although we present a framework of robot learning from demonstrations that enables a robot to learn a task from noisy demonstrations and generalizes the task to different goal points and positions of the obstacles, there are many rooms to explore and many directions for future works.

To extend the Mean-Path algorithm to higher dimensional space is a future work. To do so, one may need to compute the intersection of a trajectory and a subspace orthogonal to a reference trajectory.

Furthermore, it is interesting future work to extend the proposed reward function formulation to a higher dimensional space. For the current inverse optimal control process, we iteratively minimize the sum of the distances between a demonstrated trajectory and the one generated by a candidate reward function. This makes the algorithm computationally expensive as at each iteration a trajectory must be obtained based on the candidate reward function. A main question, however, is how to speed up the inverse optimal control processes in this context.

Here, we formalize the reward function based on the distance of the end-effector and obstacle. However, in many tasks other environmental features are important to the task. Another possible direction of future work may be to investigate other features which are important for task execution and use them to form the corresponding reward function.

Finally, at the emulation level of robot learning from demonstration, we investigate the problem of obstacle avoidance during task execution, however, for different aspects of emulation learning, such as affordance learning, a different model needs to be adopted.

Glossary

GMM/GMR	Gaussian mixture model and Gaussian mixture regression.
DMP	Dynamic Movement Primitives.
GMM	Gaussian Mixture Model.
GMR	Gaussian Mixture Regression.
HMM	hidden Markov model.
IOC	Inverse Optimal Control.
LfD	Learning from Demonstration.
NLPCA	Nonlinear Principal Component Analysis.
PbG	Programming by Guiding.
PCA	Principal Component Analysis.
RPD	Robot Programming by Demonstration.

Bibliography

[1] P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *the proceedings of the twenty-first international conference on Machine learning*, pages 1–7, 2004.

[2] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*, 19:1, 2007.

[3] E. Aertbelien and H. Van Brussel. An observation model and segmentation algorithm for skill acquisition of a deburring task. In *the proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 635–640, 1999.

[4] N. Ahmidi, G. Hager, L. Ishii, G. L Gallia, and M. Ishii. Robotic path planning for surgeon skill evaluation in minimally-invasive sinus surgery. In *Medical Image Computing and Computer-Assisted Intervention*, pages 471–478. Springer, 2012.

[5] A. Alissandrakis, C. L Nehaniv, and K. Dautenhahn. Imitation with alice: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32(4):482–496, 2002.

[6] H. Asada. A geometrical representation of manipulator dynamics and its application to arm design. *Journal of dynamic systems, measurement, and control*, 105(1):131–142, 1983.

[7] C. G Atkeson and S. Schaal. Learning tasks from a single demonstration. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 1706–1712, 1997.

[8] P. Bakker and Y. Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, pages 3–11, 1996.

[9] A. Bandura, J. E Grusec, and F. L. Menlove. Observational learning as a function of symbolization and incentive set. *Child development*, 37(3):499–506, 1966.

[10] L. Bascetta, G. Ferretti, A. Ghalamzan, G. Magnani, M. Pirotta, and M. Rocco, P. andRestelli. owards and autonomous robotic deburring system. In *Italian national conference on Motion Control*, 2013.

[11] M. A. Bauer. Programming by examples. *Artificial Intelligence*, 12(1):1–21, 1979.

Bibliography

[12] A. W Biermann and R. Krishnaswamy. Constructing programs from example computations. *IEEE Transactions on Software Engineering*, 3(1):141–153, 1976.

[13] A. Billard and Ajay K. Tanwani. Transfer in inverse reinforcement learning for multiple strategies. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3244–3250, 2013.

[14] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. *Robotics and autonomous systems*, 47(2):69–77, 2004.

[15] A. Billard, S. Calinon, and F. Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384, 2006.

[16] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. Technical report, MIT Press, 2008.

[17] S. Bitzer and S. Vijayakumar. Latent spaces for dynamic movement primitives. In *the 9th IEEE-RAS International Conference on Humanoid Robots*, pages 574–581. IEEE, 2009.

[18] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011.

[19] M. Brand and H. Aaron. Style machines. In *the proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, 2000.

[20] R. W. Byrne. Culture in great apes: using intricate complexity in feeding skills to trace the evolutionary origin of human technical prowess. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):577–585, 2007.

[21] S. Calinon. Continuous extraction of task constraints in a robot programming by demonstration framework. *Doctoral dissertation, EPFL*, 2007.

[22] S. Calinon. *Robot Programming by Demonstration*. EPFL Press, 2009.

[23] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *the proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262, 2007.

[24] S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 367–372, 2008.

[25] S. Calinon, F. Guenter, and A. Billard. Goal-directed imitation in a humanoid robot. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 299–304, 2005.

[26] S. Calinon, F. Guenter, and A. Billard. On learning the statistical representation of a task and generalizing it to various contexts. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 2978–2983, 2006.

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

[28] S. Calinon, F. D'halluin, E. L. Sauser, D. Caldwell, and A. Billard. Learning and reproduction of gestures by imitation: An approach based on hidden markov model and gaussian mixture regression. *IEEE Robotics & Automation Magazine*, 17(2):44–54, 2010.

[29] S. Calinon, E. L Sauser, A. Billard, and D. G Caldwell. Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. In *IEEE International Conference on Robotics and Automation*, pages 2671–2676, 2010.

[30] S. Calinon, Zhibin Li, T. Alizadeh, N. G Tsagarakis, and D. G Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *the 12th IEEE-RAS International Conference on Humanoid Robots*, pages 323–329, 2012.

[31] S. Chitta, I. Sucan, and S. Cousins. Moveit! *IEEE Robotics and Automation Magazine*, 19 (1):18–19, 2012.

[32] S. Cho and S. Jo. Incremental online oearning of robot behaviors from selected multiple kinesthetic teaching trials. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(3):730–740, 2013.

[33] S. Dasgupta and L. J Schulman. A two-round variant of em for gaussian mixtures. In *the proceedings of the 6th conference on Uncertainty in artificial intelligence*, pages 152–159. Morgan Kaufmann Publishers Inc., 2000.

[34] A. P Dempster, N. M Laird, and D. B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[35] G. Di Pellegrino, L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Understanding motor events: a neurophysiological study. *Experimental brain research*, 91(1):176–180, 1992.

[36] T. Dietz, U. Schneider, M. Barho, S. Oberer-Treitz, M. Drust, R. Hollmann, and M. Hägele. Programming system for efficient use of industrial robots for deburring in sme environments. In *the proceedings of 7th German Conference on Robotics (ROBOTIK)*, pages 1–6, 2012.

[37] R. Dillmann, M. Kaiser, and A. Ude. Acquisition of elementary robot skills from human demonstration. In *International Symposium on Intelligent Robotics Systems*, pages 185–192, 1995.

[38] R. Dillmann, T. Asfour, M. Do, R. Jäkel, A. Kasper, P. Azad, A. Ude, S. R Schmidt-Rohr, and M. Lösch. Advances in robot programming by demonstration. *KI-Künstliche Intelligenz*, 24 (4):295–303, 2010.

[39] D. Dong and T. J McAvoy. Nonlinear principal component analysis based on principal curves and neural networks. *Computers & Chemical Engineering*, 20(1):65–78, 1996.

[40] K. Dvijotham and E. Todorov. Inverse optimal control with linearly-solvable mdps. In *the proceedings of the 27th International Conference on Machine Learning*, pages 335–342, 2010.

[41] C. Ek, D. Song, K. Huebner, and D. Kragic. Task modeling in imitation learning using latent variable models. In *the 10th IEEE-RAS International Conference on Humanoid Robots*, pages 548–553. IEEE, 2010.

[42] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 358–363, 2006.

[43] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[44] H. Friedrich, S. Mnch, R. Dillmann, S. Bocionek, and M. Sassin. Robot programming by demonstration (rpd): Supporting the induction by human interaction. *Machine Learning*, 23 (2):163–189, 1996.

[45] Z. Ghahramani and M. Jordan. Supervised learning from incomplete data via an em approach. In *the proceeding of Advances in Neural Information Processing Systems*. Citeseer, 1994.

[46] J. J. Gibson. *The ecological approach to visual perception*. Psychology Press, 2013.

[47] S. F Giszter, F. A Mussa-Ivaldi, and E. Bizzi. Convergent force fields organized in the frog's spinal cord. *The journal of neuroscience*, 13(2):467–491, 1993.

[48] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

[49] G. Guthart, Salisbury J., and John K. The intuitivetm telesurgery system: Overview and application. In *ICRA*, pages 618–621, 2000.

[50] D. C. Halbert. *Programming by example*. PhD thesis, University of California, Berkeley, 1984.

[51] L. Han, X. Wu, W. Liang, G. Hou, and Y. Jia. Discriminative human action recognition in the learned hierarchical manifold space. *Image and Vision Computing*, 28(5):836–849, 2010.

[52] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[53] R. Heise. Demonstration instead of programming: focussing attention in robot task acquisition. *Master's thesis, Department of Computer Science, University of Calgary*, 1989.

[54] H. Hoffmann, P. Pastor, D. H. Park, and S. Schaal. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In *IEEE International Conference on Robotics and Automation*, pages 2587–2592, 2009.

[55] G. E Hovland, P. Sikka, and B. J McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 2706–2711, 1996.

[56] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *the proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1398–1403. IEEE, 2002.

[57] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[58] K. Ikeuchi and T. Suehiro. Toward an assembly plan from observation. I. task recognition with polyhedral objects. *IEEE Transactions on Robotics and Automation*, 10(3):368–385, 1994.

[59] T. Inamura, N. Kojo, and M. Inaba. Situation recognition and behavior induction based on geometric symbol representation of multimodal sensorimotor patterns. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5147–5152, 2006.

[60] M. Ito, K. Noda, Y. Hoshino, and J. Tani. Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks*, 19(3):323–337, 2006.

[61] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *IEEE International Conference on Robotics and Automation*, pages 2700–2705, 1996.

[62] S. B. Kang and K. Ikeuchi. A robot system that observes and replicates grasping tasks. In *the proceedings of 5th International Conference on Computer Vision*, pages 1093–1099, 1995.

[63] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[64] J. S. Kelso. *Dynamic patterns: The self-organization of brain and behavior*. MIT press, 1997.

[65] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *the proceeding of European Conference on Computer Vision*, pages 201–214. Springer, 2012.

[66] G. Konidaris and A. Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *the Proceedings of the 23rd international conference on Machine learning*, pages 489–496. ACM, 2006.

[67] P. Kormushev, S. Calinon, and D. G Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, 2010.

[68] D. Kulić, W. Takano, and Y. Nakamura. Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *The International Journal of Robotics Research*, 27(7):761–784, 2008.

[69] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgotter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, 28(1):145–157, 2012.

[70] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6):799–822, 1994.

[71] K. J Kyriakopoulos and G. N Saridis. Minimum jerk path generation. In *the proceedings of IEEE International Conference onRobotics and Automation*, pages 364–369. IEEE, 1988.

[72] C. Lee and Y. Xu. Online, interactive learning of gestures for human/robot interfaces. In *the proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 2982–2987, 1996.

[73] D. Lee and Y. Nakamura. Mimesis scheme using a monocular vision system on a humanoid robot. In *IEEE International Conference on Robotics and Automation*, pages 2162–2168, 2007.

[74] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *the proceedings of the 29th International Conference on Machine Learning*, pages 41 – 48, 2012.

[75] S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Neural Information Processing Systems*, pages 19–27, 2011.

[76] B. Lim, S. Ra, and F. Park. Movement primitives, principal component analysis, and the efficient generation of natural motions. In *the Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4630–4635. IEEE, 2005.

[77] H. I. Lin and C. C. Lai. Learning collision-free reaching skill from primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2383–2388, 2012.

[78] Y. Y. Lin, T. L. Liu, and C. S. Fuh. Multiple kernel learning for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1147–1160, 2011.

[79] S. Liu and H. Asada. Teaching and learning of deburring robots using neural networks. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 339–345, 1993.

[80] X. Liu, K. Li, M. McAfee, and G. W Irwin. Improved nonlinear pca for process monitoring using support vector data description. *Journal of Process Control*, 21(9):1306–1317, 2011.

[81] M. Lopes, F. S. Melo, and L. Montesano. Affordance-based imitation learning in robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1015–1021. IEEE, 2007.

[82] T. Lozano-Perez. Robot programming. *the proceedings of the IEEE*, 71:821–841, 1983.

[83] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 100(2):108–120, 1983.

[84] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *the proceedings of the 5th Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, 1967.

[85] M. Manisera. Assessing stability in nonlinear PCA with hierarchical data. In *Statistical Models for Data Analysis*, pages 217–224. Springer, 2013.

[86] G. McLachlan and D. Peel. *Finite mixture models*, volume 299. Wiley-Interscience, 2000.

[87] G. McLachlan and D. Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[88] A. N. Meltzoff and M. K. Moore. Imitation of facial and manual gestures by human neonates. *Science*, 198(4312):75–78, 1977.

[89] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki. Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, 1(3):251–265, 1988.

[90] K. Mombaur, A. Truong, and J. Laumond. From human to humanoid locomotion?an inverse optimal control approach. *Autonomous robots*, 28(3):369–383, 2010.

[91] S. Muench, J. Kreuziger, M. Kaiser, and R. Dillman. Robot programming by demonstration (rpd): Using machine learning and user interaction methods for the development of easy and comfortable robot programming systems. In *the proceedings of the International Symposium on Industrial Robots*, pages 685–685, 1994.

[92] M. Muhlig, M. Gienger, S. Hellbach, J. J Steil, and C. Goerick. Task-level imitation learning using variance-based movement optimization. In *IEEE International Conference on Robotics and Automation*, pages 1177–1184, 2009.

[93] E. Murphy-Chutorian and M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626, 2009.

[94] C. L. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In *Interdisciplinary Approaches to Robot Learning*, 1999.

[95] B. Nemec and A. Ude. Action sequencing using dynamic movement primitives. *Robotica*, 30 (05):837–846, 2012.

[96] A. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

[97] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *the proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 241–248, 2003.

[98] M. Oram and D. Perrett. Integration of form and motion in the anterior superior temporal polysensory area (STPa) of the macaque monkey. *Journal of neurophysiology*, 76(1), 1996.

[99] T. Osa, N. Sugita, and M. Mamoru. Online trajectory planning in dynamic environments for surgical task automation. In *Robotics: Science and Systems (RSS)*, 2014.

[100] U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *The Journal of Machine Learning Research*, 12(1):1249–1286, 2011.

[101] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[102] M. Pardowitz, S. Knoop, R. Dillmann, and R. Zollner. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(2):322–332, 2007.

[103] D. H. Park, H. Hoffmann, P. Pastor, and S. Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *the 8th IEEE-RAS International Conference on Humanoid Robots*, pages 91–98, 2008.

[104] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[105] A. Piazzi and A. Visioli. Global minimum-jerk trajectory planning of robot manipulators. *IEEE Transactions on Industrial Electronics*, 47(1):140–149, 2000.

[106] N. Ratliff, B. Ziebart, K. Peterson, J. Bagnell, M. Hebert, A. K Dey, and S. Srinivasa. Inverse optimal heuristic control for imitation learning. In *international conference on Artificial Intelligence and Statistics*, 2009.

[107] C. E Reiley, H. C Lin, D. D Yuh, and G. D Hager. Review of methods for objective surgical skill evaluation. *Surgical endoscopy*, 25(2):356–366, 2011.

[108] G. Rizzolatti and L. Craighero. The mirror-neuron system. *Annual Review Neuroscience*, 27 (1):169–192, 2004.

Bibliography

[109] J. Saunders, C. L Nehaniv, and K. Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *the proceedings of the 1st conference on Human-robot interaction*, pages 118–125, 2006.

[110] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3 (6):233–242, 1999.

[111] S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[112] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Robotics Research*, pages 561–572. Springer, 2005.

[113] M. Schmidt. *Graphical model structure learning with l1-regularization*. PhD thesis, University of British Columbia, 2010.

[114] M. Scholz. Validation of nonlinear pca. *Neural processing letters*, 36:21–30, 2012.

[115] M. Scholz, M. Fraunholz, and J. Selbig. Nonlinear principal component analysis: neural network models and applications. In *Principal manifolds for data visualization and dimension reduction*, pages 44–67. Springer, 2008.

[116] R. Sethi and A. Roy-Chowdhury. Activity recognition by integrating the physics of motion with a neuromorphic model of perception. In *Workshop on Motion and Video Computing*, pages 1–8. IEEE, 2009.

[117] X. Shi, Y. Lv, Z. Fei, and J. Liang. A multivariable statistical process monitoring method based on multiscale analysis and principal curves. *International Journal of Innovative Computing, Information and Control*, 9(4):1781–1800, 2013.

[118] K. Shimokura and S. Liu. Programming deburring robots based on human demonstration with direct burr size measurement. In *the proceedings of IEEE International Conference on Robotics and Automation*, pages 572–577, 1994.

[119] G. S Sidhu, N. Asgarian, R. Greiner, and M. R. Brown. Kernel principal component analysis for dimensionality reduction in fMRI-based diagnosis of ADHD. *Frontiers in systems neuroscience*, 6(1):1–17, 2012.

[120] J. J. Steil, Frank R., R. Haschke, and H. Ritter. Situated robot learning for multi-modal instruction and imitation of grasping. *Robotics and Autonomous Systems*, 47(2):129–141, 2004.

[121] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59(11):910–922, 2011.

[122] W. R. Tanner. *Industrial Robots: Fundamentals*. Society of Manufacturing Engineers, 1981.

[123] D. E Thompson and J. Russell. The ghost condition: imitation versus emulation in young children's observational learning. *Developmental Psychology*, 40(5):882 – 899, 2004.

[124] S. B Thrun and T. M Mitchell. Integrating inductive neural network learning and explanation-based learning. In *International Joint Conference on Artificial Intelligence*, pages 930–936, 1993.

[125] M. Tomasello. *Do apes ape?* in Social learning in animals: The roots of culture. (pp. 319-346), Academic Press, 1996.

[126] S. K. Tso and K. P. Liu. Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. In *the proceedings of The IEEE International Conference on Industrial Technology*, pages 294–298, 1996.

[127] C. P. Tung and A. C. Kak. Automatic learning of assembly tasks using a dataglove system. In *the proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–8, 1995.

[128] A. M. Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950.

[129] A. Ude, A. Gams, T. Asfour, and J. Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[130] D. L Vail, M. M Veloso, and J. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multi agent systems*, pages 1331 –1338, 2007.

[131] S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. *Autonomous Robots*, 12(1):55–69, 2002.

[132] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *the proceedings of Conference on Computer Vision and Pattern Recognition*, pages 511–518. IEEE, 2001.

[133] F. Wallner, M. Kaiser, H. Freidrich, and R. Dillman. Integration of topological and geometrical planning in a learning mobile robot. In *the Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 1–8. IEEE, 1994.

[134] A. Whiten and R. Ham. On the nature and evolution of imitation in the animal kingdom: reappraisal of a century of research. *Advances in the Study of Behavior*, 21(1):239–283, 1992.

[135] A. Whiten, N. McGuigan, S. Marshall-Pescini, and L. M Hopper. Emulation, imitation, over-imitation and the scope of culture for child and chimpanzee. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1528):2417 – 2428, 2009.

[136] D. Wood. Social interaction as tutoring. *In Interaction in human development (C. Snow, M. Bornstein, J. Bruner)*, pages 59–80, 1989.

[137] B. Xu, A. Kurdila, and D. Stilwell. A hybrid receding horizon control method for path planning in uncertain environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4887–4892. IEEE, 2009.

[138] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1385–1392. IEEE, 2011.

[139] H. Zhang, H. Chen, N. Xi, G. Zhang, and J. He. On-line path generation for robotic deburring of cast aluminum wheels. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2400–2405, 2006.

Bibliography

[140] R. Zhang and W. Wang. Learning linear and nonlinear PCA with linear programming. *Neural Processing Letters*, 33(2):151–170, 2011.

[141] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *the 28th AAAI Conference on Artificial Intelligence*, pages 1433 –1438, 2008.

[142] G. Ziliani, A. Visioli, and G. Legnani. A mechatronic approach for robotic deburring. *Mechatronics*, 17(8):431–441, 2007.