Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik

Human-centered Assistive Robotics (HCR)

Task parameterized robot skill learning via programming by demonstrations

Affan Pervez

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. Sandra Hirche
Prüfer der Dissertation:	
	1. Prof. Dr. Dongheui Lee
	2. Prof. John Folkesson, Ph.D. Royal Institute of Tecnology (KTH), Sweden

Die Dissertation wurde am 12.04.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am **23.07.2018** angenommen.

Acknowledgment

First of all I would like to thank almighty Allah, the most beneficent and the most merciful. Without his will, this work would not have been possible. Then I would like to thank my parents for their well wishes and prayers. I dedicate this thesis to my mother, for all the sacrifices that she incurred to make me a better person. Her consistent desire for me to learn more and more really helped a lot, to pave the path, of the journey through my PhD.

I will also like to admit the contribution of all the teachers from school till today, without whose efforts I could not have reached this level. Especially Prof. Lee for showing trust on my capabilities and providing me with this wonderful opportunity of working under her supervision. I still remember the unusual delay caused during my VISA application of Germany and the patience she showed until I joined the lab. Additionally it is due to her guidance that I have developed the capability of conducting research independently. Moreover I would also like to thank German Research Foundation (DFG) which covered most of the funding of my research activities during my stay at Human Centered Assistive Robotics (HCR) group.

My colleagues at HCR were also a great help to me. I would like to thank Matteo Saveriano for having wonderful discussions and for working along with me as a Teaching Assistant in four consecutive semesters of Machine Learning in Robotics course, Sang-ik An for teaching me that it is never late in life and Thomas Eiband for translating my thesis abstract in German. I would also like to thank Shile Li, Pietro Falco and Karna Potwar for being wonderful lunch buddies and Sussanne Schneider for handling all of the administrative matters.

Contents

Abstract

Robotics researchers have been always in pursuit of developing machines that can act like humans. To achieve such an ambition, it is important to understand how humans learn. From a new born, until an adult, a human goes through several stages of developments in acquiring new skills. We do not learn every thing on our own and most of it is taught to us by our parents and teachers. For a robot to act like humans, it is also important for it to learn like humans. An important aspect of learning like humans is to be able to mimic and generalize a demonstrated skill. Programming by Demonstration (PbD) focuses on this aspect of robotics.

Multimodal density estimation is a preliminary step of the PbD approaches presented in this thesis. The main challenges in Expectation Maximization (EM) based density estimation are not knowing the original number of clusters in the data, initialization of model parameters for EM and overfitting or under fitting of the learned model. These challenges are addressed by the simulated annealing based EM approach presented in this work. The performance of the presented approach outperforms other EM based approaches in similar settings and is validated through synthetic as well as real datasets. Next I show that a learned mixture model can be utilized for PbD tasks.

With kinesthetic teaching as an input modality I present a Gaussian Mixture Model (GMM) based task parameterized skill learning approach. Existing PbD approaches require a large amount of demonstrations. Additionally they also perform poorly when generalizing beyond the demonstrated regions. The approach presented in this thesis addresses these shortcomings by presenting a new task parameterized skill learning method. As EM can utilize incomplete data, it is shown that the generalization capabilities of a learned skill can be significantly improved by introducing incomplete data spanning the task parameters i.e. the input space.

In PbD, variables of interest, which are also termed as task parameters, have to be identified and extracted during motion reproduction. To avoid extraction of task parameters, it is shown that by modelling the forcing terms of a Dynamic Movement Primitive (DMP) with a Convolution Neural Network (CNN), the high dimensional camera image can be processed directly to reproduce a learned skill. The proposed architecture possesses the desirable attributes associated with a dynamical system, as well as generalization properties associated with the deep learning.

Apart from kinesthetic teaching, this thesis also focuses on developing PbD approaches by utilizing teleoperated demonstrations. Since it is not always possible to co-locate a human teacher along with a robot, for instance in deep sea or space applications, teleoperated demonstrations have to be utilized for learning. The challenges which arise from teleoperated demonstrations are their high temporal and spatial variations, different initial and final configurations and incomplete demonstrations. This thesis addresses these challenges by proposing a simultaneous trajectories

synchronization and encoding algorithm. Moreover it is shown that the learned control can also be utilized in shared teleoperation setting, by executing the learned skill in conjunction to a human operator. The PbD approaches presented in this thesis outperform the state of the art PbD approaches devised for similar tasks in simulated as well as real robot experiments.

Zusammenfassung

In der Robotik-Forschung wird seit langem das Ziel verfolgt, Maschinen zu entwickeln, die wie Menschen agieren können. Um dieses Ziel zu erreichen, ist es wichtig das menschliche Lernen zu verstehen. Vom Neugebohrenen bis zum Erwachsenen durchläuft ein Mensch mehrere Entwicklungsstufen zum Erlangen neuer Fähigkeiten. Dabei lernen wir nicht alles von selbst, das meiste wird uns durch unsere Eltern und Lehrer vermittelt. Damit ein Roboter wie ein Mensch agieren kann, ist es auch wichtig dass er wie ein Mensch lernt. Ein wichtiger Aspekt des menschlichen Lernens es es, eine demonstrierte Fähigkeit zu imitieren und zu generalisieren. Programmierung durch Demonstration (PbD) fokussiert sich auf diesen Aspekt der Robotik.

Multimodale Dichteschätzung ist ein erster Schritt der PbD-Ansätze, die in dieser Arbeit vorgestellt werden. Die hauptsächlichen Herausforderungen bei der auf Expectation-Maximization (EM) basierenden Dichteschätzung, ist die fehlende Information über die Anzahl der Cluster in den Daten, sowie die Initialisierung der Modelparameter für EM und das Über- oder Unteranpassen des gelernten Models zu vermeiden. Diese Herausforderungen werden durch den vorgestellten Simulated-Annealing-basierten EM-Algorithmus bewältigt, der in dieser Arbeit vorgestellt wird. Die Leistung des präsentierten Algorithmus übertrifft andere EM-Algorithmen in ähnlichen Szenarien und wird sowohl durch künstliche als auch reelle Datensätze validiert. Im folgenden wird gezeigt, wie ein gelerntes Mixture-Model für PbD-Aufgaben verwendet werden kann.

Mit kinästhetischem Teachen als Eingabemodalität wird ein Gaussian-Mixture-Model (GMM)-basierter Ansatz vorgestellt, der das Lernen von parametrisierbaren Fähigkeiten ermöglicht. Bestehende PbD Ansätze erfordern eine große Anzahl an Demonstrationen. Außerdem funktionieren diese nur eingeschränkt, wenn versucht wird außerhalb der demonstrierten Region zu generalisieren. Der in dieser Arbeit vorgestellte Ansatz behandelt diese Defizite durch die Präsentation eines neuen Algorithmus zum Lernen von aufgabenparametrisierbaren Fähigkeiten. Da EM auch unvollständige Daten nutzen kann, zeigt sich, dass die Generalisierungsmöglichkeit der gelernten Fähigkeit signifikant durch das Einführen dieser unvollständigen Daten verbessert wird, welche im Eingangsraum der Aufgabenparameter liegen.

Bei PbD müssen Variablen von Interesse, die auch als Task-Parameter bezeichnet werden, während der Bewegungswiedergabe identifiziert und extrahiert werden. Um das Extrahieren von Aufgabenparametern zu vermeiden, wird gezeigt, dass ein hochdimensionales Kamerabild direkt verarbeitet werden kann, um eine gelernte Fähigkeit zu reproduzieren. Dabei werden die Forcing-Terms einer Dynamic-Movement-Primitive (DMP) mit einem Convolutional-Neural-Network (CNN) modelliert. Die vorgestellte Architektur besitzt die erwünschten Eigenschaften eines dynamischen Systems und erreicht die Generalisierungseigenschaften mittels Deep-Learning.

Neben kinästhetischem Teachen konzentriert sich diese Arbeit auch auf die Entwicklung von PbD-Ansätzen unter Verwendung von teleoperierten Demonstrationen. Da es für einen menschlichen Demonstrator nicht immer möglich ist sich neben einem Roboter aufzuhalten, z.B. in der Tiefsee oder bei Raumfahrtanwendungen, müssen teleoperierte Demonstrationen eingesetzt werden. Die Herausforderungen die sich durch teleoperierte Demonstrationen ergeben, sind große zeitliche und räumliche Variationen, verschiedene Anfangs- und Endkonfigurationen und unvollständige Demonstrationen. Diese Arbeit befasst sich mit diesen Problemstellungen durch die Vorstellung eines simultanen Trajektorien-Synchronisations- und Codierungsalgorithmus. Darüber hinaus wird gezeigt, dass die gelernte Steuerung auch während gemeinsamer Teleoperation durch Ausführung der gelernten Fähigkeit zusammen mit einem menschlichen Operator ausgeführt werden kann. Die in dieser Arbeit vorgestellten PbD-Ansätze übertreffen jene des Stands der Technik für ähnliche Aufgaben sowohl in simulierten als auch in realen Roboterexperimenten.

Publications resulting from this thesis

International journals

Affan Pervez, and Dongheui Lee "Learning task-parameterized dynamic movement primitives using mixture of GMMs." Intelligent Service Robotics (2017): 1-18.

Affan Pervez, Hiba Latifee, Jee-Hwan Ryu, and Dongheui Lee "Motion Encoding with Asynchronous Trajectories of Repetitive Teleoperation Tasks and its Extension to Human-Agent Shared Teleoperation." ***(under progress)***

Peer-reviewed conferences

Affan Pervez, Hiba Latifee, Jee-Hwan Ryu, and Dongheui Lee "Human-Agent Shared Teleoperation: A Case Study Utilizing Haptic Feedback." Asia Haptics, 2018.

Affan Pervez, Yuecheng Mao, and Dongheui Lee "Learning Deep Movement Primitives using Convolutional Neural Networks." 17th International Conference on Humanoid Robotics (Humanoids). IEEE-RAS, 2017.

Affan Pervez, Arslan Ali, Jee-Hwan Ryu, and Dongheui Lee "Novel learning from demonstration approach for repetitive teleoperation tasks." World Haptics Conference (WHC), IEEE, 2017. **(Ranked among top ten percent papers)**

Affan Pervez, and Dongheui Lee "A Componentwise Simulated Annealing EM Algorithm for Mixtures." Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz). Springer, Cham, 2015.

International workshops

Omair Ali, Affan Pervez, and Dongheui Lee "Robotic Calligraphy: Learning From Character Images." 8th International Workshop on Human-Friendly Robotics (HFR), 2015.

Nomenclature

Abbreviations

CEM^2	Component-Wise EM for Mixtures
CSAEM^2	Component-Wise Simulated Annealing EM for Mixtures
CNN	Convolutional Neural Network
D-DMP	Deep-Dynamic Movement Primitive
DAEM	Deterministic Annealing Expectation Maximization
DMP	Dynamic Movement Primitive
DOF	Degree of Freedom
DTW	Dynamic Time Warping
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
GPR	Gaussian Process Regression
GPU	Graphics Processing Unit
GP	Gaussian Process
HMM	Hidden Markov Model
IDGMM	Incomplete Data Gaussian Mixture Model

LWPR	Locally Weighted Projection Regression
LWR	Locally Weighted Regression
MMDL	Mixture Minimum Discription Length
PbD	Programming by Demonstration
PC	Personal Computer
ReLU	Rectified Linear Unit
TP-DMP	Task Parameterized-Dynamic Movement Primitive

Conventions

$\mathbf{A}$	Matrix
$\mathbf{a}$	Vector
$\mathbf{a}^\top$, $\mathbf{A}^\top$	Transpose of vector, matrix
$\mathbf{A^{-1}}$	Inverse of matrix
$\dot{\mathbf{a}}$	First-order time derivative
$\mathcal{F}(\cdot)$	Scalar function
$\mathcal{N}(\cdot)$	Multivariate Gaussian distribution
$\|\cdot\|$	Norm
a, A	Scalar
$max(\mathbf{A})$	Maximum value in a matrix
$max(\mathbf{a})$	Maximum value in a vector
$min(\mathbf{A})$	Minimum value in a matrix
$min(\mathbf{a})$	Minimum value in a vector

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1. Motivation

Scientists have always been in pursuit of developing the machines and artificial intelligence algorithms that can outperform humans. A key component in achieving such goals is to investigate that *can the machines learn like humans*. To acquire such a capability, a robot should be able to learn directly from a human teacher. An infant does not learn everything on its own but a lot of them are taught by its parents. In the same way, a robot's learning speed can be significantly increased if it can learn a skill from a human teacher. This can also eliminate the need to perform trial and error learning by a robot, which can consume a lot of time. Programming by Demonstration (PbD) is an active research area in robotics where a human teaches a skill to a robot.

In PbD, a skill at trajectory level can be encoded by utilizing dynamical systems or statistical modeling. Encoding of human movements can be performed in task space, joint space or torque space. Moreover PbD can encode discrete movements as well as limit cycles. A preliminary step of many statistical modeling based PbD approaches is to encode the demonstrations by using a Gaussian Mixture Model (GMM). The most widely used method for fitting a GMM is Expectation Maximization (EM). EM maximizes the likelihood locally and hence can converge to a local maxima. Other problems that are associated with EM based density estimation are initialization of model parameters, convergence to the boundary of the parameter space and estimating the appropriate model complexity. All of these issues have to be considered when employing a GMM based PbD approach.

Humans can quickly acquire basic motor skills from a teacher. PbD also aims to achieve such capabilities. In PbD, instead of programming each task from scratch, the tasks are learned from human demonstrations. This presents a natural way of quickly encoding different tasks and can significantly reduce the setting up time in industrial applications. PbD does not mean the exact replaying of the motions but the learned task should be generalizable for a wide range of scenarios. Consider for instance the task of throwing a ball in the basket. After observing few demonstrations, the robot should be able to generalize the throwing motion for different basket positions. In PbD, external environmental variables, for instance the basket position in the ball throwing task, are termed as task parameters.

A lot of research has been done to adapt a learned skill while considering the task parameters. Humans can learn a skill by observing only a few demonstrations. Similar capability is also desir-

able by robots as collecting too many demonstrations can be a time consuming and tedious task. As mentioned earlier, learning in PbD does not mean the exact replicating of the demonstrations. Thus the robot should be able to generalize the learned skill for a wide range of situations. Most of the existing PbD approaches perform well when interpolating within the demonstrated ranges of the task parameters but their performance deteriorate when extrapolating beyond the demonstrated ranges of the task parameters. In PbD, learning from relatively few demonstrations and for performing well in the extrapolation regions are the important topics that need further investigation.

Deep neural networks have shown promising results in many domains i.e. computer vision, speech recognition, natural language processing etc. Deep learning approaches can perform supervised, unsupervised and reinforcement learning. In deep learning, multiple layers correspond to different levels of abstraction. A relatively new research direction is to utilize deep learning for PbD scenarios. Since deep convolutional neural networks can extract object/task specific features, they are a useful tool for processing high dimensional camera images. This provides an exciting new research direction for encoding PbD tasks by directly utilizing high dimensional camera images. The challenge which arises when utilizing deep learning for PbD is that deep learning approaches usually require a large amount of training data. Collecting too many demonstrations often creates a feasibility issue for many real world robotics tasks. Also since deep learning provides a black box approximator, the outcome can be nondeterministic. These issues have to tackled when applying deep learning for PbD tasks. In this thesis, we handle these challenges by combining deep learning with a model besed approach, as will be shown later.

An important aspect of PbD is to collect demonstrations from a teacher. There are different modalities that a human can utilize for providing demonstrations to a robot. The two main approaches for providing demonstrations are kinesthetic teaching and teleoperation. In kinesthetic teaching, a human teacher actually holds the robot's end-effector to demonstrate a task. Teleopearation is employed when the humans cannot be colocated with the robots. Examples include nuclear waste handling, deep water and space applications where teleopearation is the only way for providing demonstrations in such scenarios. Teleoperated demonstrations usually have high temporal/spatial variations. That is why the existing approaches developed for PbD via kinesthetic teaching often fail to scale well when applied to teleoperation settings.

Apart from task automation, correct task execution is also important. A wrong task execution can be dangerous for the robot as well as the environment. Human supervision can be deployed to avoid unsafe interactions. Since the goal of task automation is generally to reduce the workload of a human operator, another approach is the control sharing inbetween the learned model and a human operator. This ensures correct task execution while at the same time it also reduces the mental workload of the human operator.

This thesis focuses on the challenges highlighted so far. It aims at developing a PbD approach that can learn from very few human demonstrations and can generalize beyond the demonstrated range of task parameters. As Gaussian Mixture Model (GMM) based density estimation is an integral part of the PbD approaches presented in this thesis, the challenges associated with GMM based density estimation via Expectation Maximization are also addressed in this thesis. Additionally it is also shown that with deep learning, the camera image can be directly utilized for generalizing a learned skill. The developed approach preserves the desirable attributes required for the PbD tasks while also preserving the generalization capabilities associated with deep learning. Moreover a PbD approach to learn from teleoperated demonstrations is also presented in this thesis. The presented approach can handle high spacial/temporal variations in the teleoperated demonstrations. It is also shown that the model learned for teleoperation can either be completely autonomously executed or in shared control settings along with the human operators.

1.2. Related works

This section presents the related works about the chapters to follow.

1.2.1. GMM and EM

Finite mixture models provide a probabilistic tool for modeling arbitrarily complex probability density functions. This ability enables mixture models to have applications in many domains such as regression, pattern recognition, image segmentation and machine learning [49], [57, 60]. The usual approach for fitting a GMM is to start with a fixed number of components and then applying EM to get the maximum likelihood (ML) parameter estimate of the model. The EM is usually applied multiple times with different initializations. K-means can be used for initializing each EM cycle. At the end the model parameters which yield the highest likelihood value are selected [11,31]. This approach has the following drawbacks: It has to apply EM multiple times for each different initialization and it only provides parameters estimate for a fixed model complexity (number of components). If the model complexity cannot be determined in advance, then the same procedure of multiple startups has to be applied to the models with different complexity levels (more computational burden). Afterwards a model selection criterion (discussed in section 2.4.2) can be used to select one among the competing set of models with different complexities.

As the likelihood function is multimodal, it is possible that during EM the estimate may converge to the boundary of the parameter space. This means that the weight of one of the component may approach to a small value, with its covariance matrix becoming singular and hence making the likelihood value to approach infinity. When this occurs, EM should be aborted and restarted with different initialization of the parameters. *Deterministic Annealing EM* (DAEM) algorithm [105] has been proposed for avoiding the estimate to converge at the boundary of the parameter estimate during EM. DAEM uses the ideas from statistical mechanics, by applying the concept of maximum entropy. The objective function is considered as the thermodynamic free energy, which is regulated by a temperature parameter β. During DAEM β is gradually increased. $\beta = 0$ makes the posterior distribution uniform while $\beta = 1$ makes the posterior as the original distribution. EM is first applied at a higher temperature, which is represented with a small β value. After convergence, the temperature value is slightly decreased and then again EM is applied to the parameters estimate of the last iteration. This cycle continues until the temperature value reaches its minimum value with the maximum β value of one. This corresponds to the transition of posterior distribution from nearly uniform to the original distribution. DAEM can avoid covariance matrix of the components from becoming singular but can get trapped at saddle points. For escaping a saddle point, different search strategies are proposed by [105]. The drawback of DAEM is high computation cost due to application of EM at each temperature value. Search around the saddle point further increases the computational burden of DAEM.

Broadly speaking there are three main approaches for estimating the number of components [30]: EM based methods [28, 67], Variational Bayesian methods [108] and stochastic methods by using Markov Chain Monte Carlo (MCMC) sampling [9, 88]. Variational Bayesian methods avoid overfitting but only provide an approximate solution while Stochastic methods are computationally very expensive. Moreover there are two types of EM based methods in which the number of components needs not to be fixed in advance: Firstly *divisive* where the estimate starts from a single component which split into multiple components as the algorithm proceeds [13,22,114] and secondly *agglomerative* where the estimate starts from a large number of components which are decreased as the algorithm proceeds [30,31]. A variety of ways have been proposed for spliting or merging GMM components [13, 64, 106, 113, 114]. In [37] and [30] Kullback-Leibler divergence has been used, while adapting the components, for keeping the prior and posterior densities close

to each other.

1.2.2. Programming by Demonstration

Imitating a task through observations is inherently easy for humans, but surprisingly a challenge for robots. In general, a robot has to be pre-programmed for performing different tasks. A slight change in a task or the environment require re-programming of the robot, which can be a tedious and a time consuming process [10]. Programming by Demonstration (PbD), also known as imitation learning, provides an intuitive way to readily teach new skills to the robots [8, 10]. In PbD, instead of programming, a task is learned from human demonstrations. In general there are two main approaches for learning motor skills; firstly those based on mimicking motion data using dynamical systems, i.e. Dynamic Movement Primitives (DMP) [72,99], secondly those relying on statistical Machine Learning (ML) techniques, i.e. Gaussian Mixture Models (GMMs) [21] and Hidden Markov Models (HMMs) [59, 61]. DMPs consider one-shot learning and provide spatial and temporal scalability properties as well as guaranteed convergence to the goal position. Learning is done at the acceleration level and many variations exist for DMPs. For example [70] has shown that additional coupling terms defined using potential fields can be combined with a DMP. This provides an obstacle avoidance behavior while performing the motion reproduction. [74] has shown that sensor traces can be incorporated into a DMP for achieving online motion adaptation. Their approach provides robustness against external disturbances and uncertainty in the goal position. The benefit of this approach is demonstrated through an object grasping task, where the uncertainty in an object's position can be tackled by utilizing the sensory feedback, which the robot also received for the demonstrated grasping motions. An extension of [74] is [73] where they have used sensor traces to continuously predict all associated sensory signals, by combining a sequence of DMPs for achieving complex tasks. [35] has presented a way to encode the rhythmic DMPs without any prior knowledge about their frequency and waveform.

Statistical ML based approaches directly learn from spatial data and can easily encode multiple demonstrations at a time. These approaches encode the trajectories along with their associated variances. There are many benefits of estimating the variances of the encoded trajectories. For instance variance in case of external perturbations can indicate how strictly the generated motion should be followed in a certain region [10]. For a compliant robot, the variance information is also useful for setting the stiffness gains. Moreover multiple constraints can be encoded with statistical ML approaches, by extracting redundancies across the demonstrations [15]

Reinforcement learning (RL) can take a lot of iterations to converge. This in general is not desirable in robotics, as it can be detrimental for a robot. PbD is often applied to speed up the learning process for RL. The goal of PbD for RL is to come up with a good initialization of the model parameters. RL is then applied as a second step to further refine the learned parameters values. Additionally in a RL task, the learned variances can be utilized for setting the exploration gains [50]. In a pan cake flipping task, the imitation alone fails to achieve the desired behavior. By utilizing the variance information, [50] significantly speeded up the learning process and learned the desired pan cake flipping behavior after only 50 trials. In the same manner a humanoid robot successfully learned to target aims in archery by utilizing PbD and RL [51].

PbD is not limited to skill transfer from a teacher to a robot but can also be utilized for learning collaborative tasks. For instance a HMM model can be incrementally refined for incorporating a physical human interaction by using a motion refinement tube [59]. The encoding motion can also be utilized for performing collaborative behaviors [20, 90]. For e.g. in a collaborative object manipulation task, the robot not only learns the desired path but also the required interaction forces/torques to carry the object through the desired path. The robot's impedance can be modified along a task execution for facilitating collaborative behavior. In [91], the robot's impedance is

modified by estimating virtual stiffness matrices from the demonstrations.

1.2.3. Task Parameterized skill learning

Task parameterized skill learning falls under the general category of PbD. Task parameterized learning aims at developing PbD approaches that considers external environmental factors. Examples include an object's position in an obstacle avoidance task and the position of the target in an aiming task. This makes the motion encoding challenging because now these external environmental factors, which we also term as task parameters, govern the behavior of the motion. As in PbD, there are also two main approaches for task parameterized learning in PbD i.e. those relying on statistical machine learning techniques and those based on dynamical systems.

Many statistical machine learning based approaches for task parameterized learning utilize GMMs. Among the GMM based approaches, [21] used different frames of reference for capturing distinct aspects of multiple demonstrations. Task parameters are defined as frames of reference. For generalization, the already learned GMMs are placed with respect to new frames and multiplied to retrieve a Task Parameterized-GMM (TP-GMM) for a new scenario. A similar method is [17], where they have shown that such an approach can also provide some extrapolation capability. Additionally the TP-GMM can also handle missing frames of reference during demonstrations as well as reproductions. The missing frames of reference can be a result of variable number of objects or visual occlusion [5]. However, TP-GMM based approaches lack typical properties associated with DMPs, e.g. spatial amplification of the movement and guaranteed convergence to a goal position. Additionally the frames of references cannot be arbitrarily placed and the user needs to have a certain level of understanding about how to place the frame of references for the TP-GMM to be successful.

Task parameterized learning can also be performed by utilizing dynamical systems based approaches. Task specific variations of DMPs are considered in [33, 65, 103, 104]. Basis targets can be extracted for the corresponding style parameters of each demonstration [65]. The basis targets and style parameters are combined to get appropriate basis functions. For generalization to a new situation with different task parameters, for example different height of an obstacle in a point to point reaching task, one has to provide appropriate style parameters. To get the style parameters for novel situations, the mapping from task parameters to style parameters is learned by supervised learning. A more direct 2-Step (2S) approach is considered in [33, 104]. In the first step a mapping from task parameters to the DMP parameters is learned. It can either be learned by Locally Weighted Regression ($\mathbf{LWR}^{2S}$), by placing local kernels along the task parameters [104], or a function approximator such as Gaussian Process Regression ($\mathbf{GPR}^{2S}$) [33]. In the second step, the inferred DMP parameters are used for motion generation. An extension of [104] is [103], where they used Locally Weighted Projection Regression (LWPR) [109] for avoiding manual placement of the basis functions, which significantly reduces the required number of basis functions as compared with LWR. They also emphasized that any suitable function approximator can be used for directly encoding forcing terms of the DMPs, eliminating need to follow the 2-Step procedure. When using GPR for learning the direct encoding, they called their approach $\mathbf{GPR}^{1S}$, where the superscript 1S and 2S emphasize on the 1-Step and 2-Step approaches respectively. These approaches can provide generalization capability when interpolating along the task parameters, but they do not perform well for very few demonstrations and cannot extrapolate beyond the demonstrated regions of task parameters, as is also shown in the experiments.

Probabilistic movement primitives (ProMPs) have been shown to have better inference capabilities than a DMP and can combine or blend multiple demonstrations to achieve task specific generalizations. However, ProMPs require a large number of demonstrations for learning. According to [95], ProMPs have a high-dimensional covariance matrix and many free parameters.

That is why they suffer from overfitting without a large number of demonstrations.

1.2.4. Programming by Demonstration with Deep learning

Deep learning is the domain of machine learning which utilize deep architectures for learning data representations. It can perform supervised, semi-supervised and unsupervised learning. Its inspiration is loosely connected with the biological nervous system. The idea behind multiple layers is to have multiple levels of abstractions/representations. A $(n-1)$ layer network requires exponentially large amount of parameters to model the same model complexity as that of a n layer network. Deep networks are powerful function approximators and have shown promising results when utilizing high dimensional inputs such as camera images [54]. This is due to their ability to learn data specific features. The main challenges which are usually encountered when training deep networks are vanishing gradient problem and overfitting issues. Parameters initialization technique is usually utilized for handling the vanishing gradient. The common techniques which are employed for avoiding overfitting are using large amount of training data, weights dropout during training and sparse connections.

Levine et al. have shown that Convolutional Neural Networks (CNNs) can be used for generating motor actions, by extracting useful features directly from camera images [63]. Their experiment consists of learning robotic grasping from monocular images by using RL. They used 800,000 grasp attempts executed over 14 robotic manipulators. Another similar self-supervising deep learning based approach is presented in [82]. It utilizes 50,000 tries and 700 robot hours to learn the object grasping task. The amount of resources needed and the learning time makes the applicability of these approaches infeasible for real procedure [24, 62]. A drawback of [62] is that the task has to be first formulated as an optimization problem and requires the user to define a cost function for the executed actions. These constraints limit the applicability of their approach to complex tasks. A PbD approach based on a combination of convolutional auto-encoders and a fully connected neural network is presented in [112]. The auto-encoder is utilized to come with a lower dimensional compact representation of the image. This lower dimensional representation is then passed to the fully connected layers for motion reproduction. A shortcoming of this approach is that it cannot achieve the generalization properties which are typically associated with the DMP based approaches, for instance temporal rescaling of the motion and the variation of the goal position. The network in [112] has to be retrained from scratch, even for changing the speed of the motion execution.

1.2.5. Programming by Demonstration for Teleoperation

The common teaching modalities for providing demonstrations in PbD include (i) vision based human motion tracking and (ii) kinesthetic teaching. In vision based motion tracking, kinematics information is extracted by a motion tracking system, which is then mapped to a physical model that resembles the teacher. This type of modality can be utilized if a human teacher has to teach a skill to a humanoid robot [43] [69]. A limitation of vision based tracking systems is that the robot should have similar kinematics as that of the teacher. If the kinematics of the teacher and the robot are different then there arises a mapping issue. Kinesthetic teaching involves providing physical action as a learning paradigm where a human teacher physically holds and moves the robot's body to demonstrate the skill. Kinesthetic teaching rules out the mapping problem [58] [97]. This type of modality is also often utilized by the sports coaches when they guide the players by actually holding their limbs to generate the motion. For kinesthetic teaching, the robot is put to gravity compensation mode in which it feels weightless to the teacher and thus can be easily moved to

generate the demonstrations. Compared with vision based tracking systems, kinesthetic teaching is a more widely used modality for teaching simple tasks.

However, learning paradigm based on kinesthetic teaching is not feasible for certain applications where human demonstrator cannot be co-located with a robot, for instance in nuclear waste handling, underwater manipulation and space applications. In those cases, teleoperation often can be considered as the only way for operating a robot located at such inaccessible places [3, 42]. In teleoperation setup, the operator executes motion on the master device, which is then transmitted and executed on the slave device. There are several studies which use PbD with teleoperation. One of the earliest work was presented in 1993 which illustrates the skills learning using Hidden Markov Models (HMMs) [25]. In [79], NASA space humanoid robot learned how to perform low level task "reaching and grasping", whereas high-level commands were given by the operator. Rozoetal. [93, 94] used Gaussian Mixture Model (GMM) and Gaussian Mixture Regression (GMR) for teaching a rigid-container emptying skills to a robot via a haptic interface. Other work considered encoding of force/torque signal with large time discrepancies by using a GMM with temporal information encapsulated by a HMM [92]. In [100], authors used a teleoperation setup for learning of human grasping skills, by utilizing motion and force data. To avoid an operators burden, multiple operators can be employed to control the robot. This also raises an interesting question about how should the control be shared inbetween multiple operators. [75] has shown that if they are controlling the same degree of freedom, then their authority can be dynamically distributed based on their field of views. Employing multiple operators can be economically infeasible. To avoid multiple operators, another option is to utilize a learned controller along with a human operator. Now in case of control sharing, the human can actually feel the intention of the learned controller in the form of a haptic feedback. The haptic feedback in turn can guide the human operator to execute the task efficiently, increases the teleoperated task performance and can be utilized online. Additionally virtual fixtures, which are constraints on the executed motion, can be included to help the operator in executing a task [2]. With virtual fixtures, human operator takes care of the intelligent part i.e. where to place a glass full of water, while the constraints during the motion execution i.e. keeping the glass upright are automated.

There are tasks in which the time/phase variable has to be the input for generating the motion. For instance if a robot has to insert a peg in a hole, then with the end-effector positioned directly above the hole, it cannot be decided that whether it is performing an insertion action or a withdrawal action. With the phase variable as an input, it can be determined that where the robot is in the execution of the movement. The main limitations of the existing phase variable based learning approaches for teleopearation are that they cannot utilize demonstrations with large temporal and spatial variations, partially executed demonstrations and demonstrations which start and end at different regions. Also they rely on learning separate models for autonomous control and for shared teleopearation control inbetween the learned model and a human operator.

1.3. Overview of the thesis

Figure 1.1 presents a general overview and the connection in between different chapters of this thesis.

- Chapter 2 highlights the challenges encountered in fitting a GMM with the EM algorithm. It then presents a systematic approach for initializing GMM components. The initialization approach uses exploration strategy to cluster nearby data points. Afterwards it presents a combination of simulated annealing and the EM algorithm. The presented agglomerative clustering approach can determine the optimal model complexity and avoids overfitting issues. The performance of the presented approach is compared and validated on synthetic as

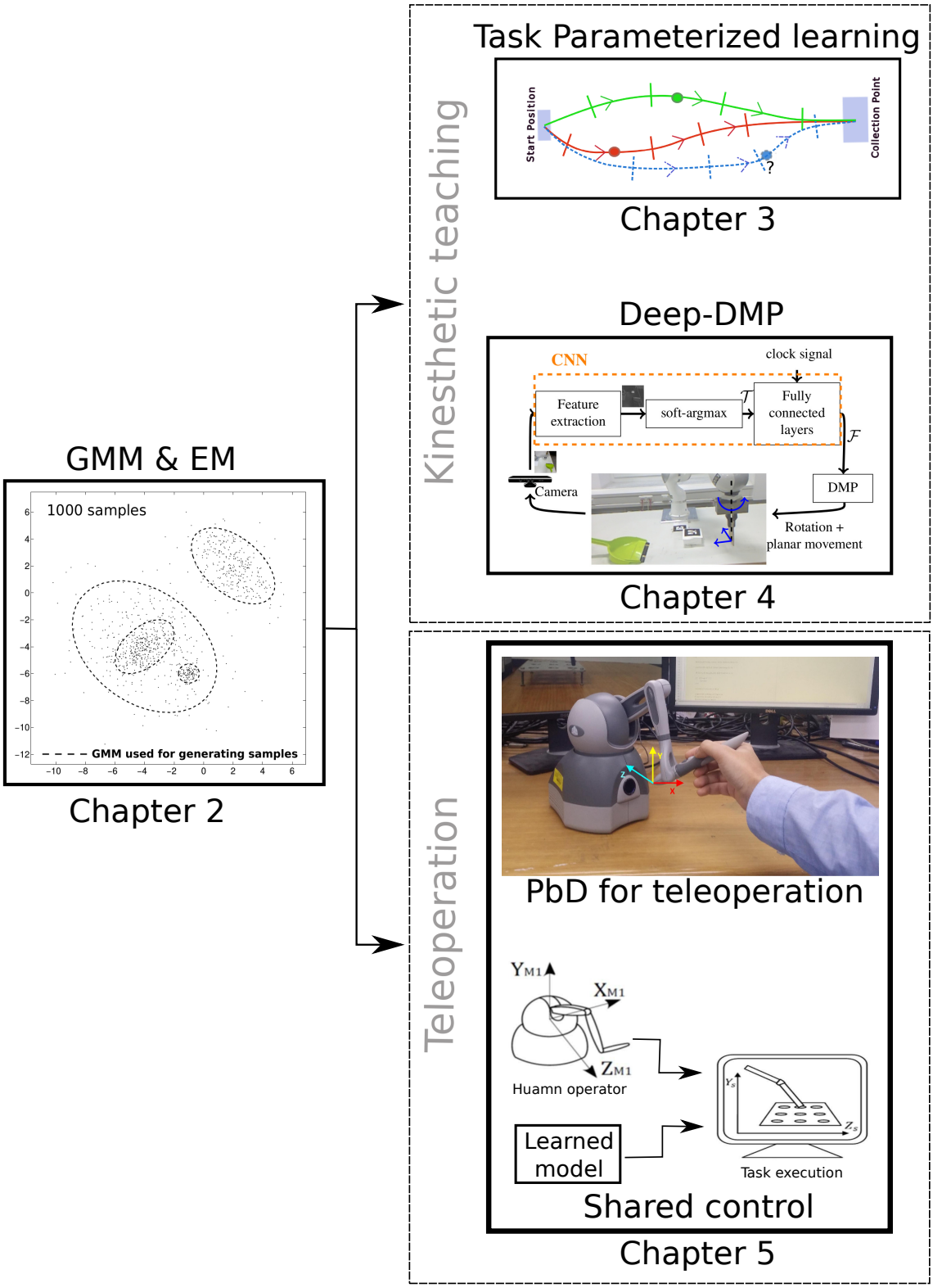


Figure 1.1.: A general overview of the thesis.

well as real datasets. The material presented in this chapter is published in:

Affan Pervez, and Dongheui Lee "A Componentwise Simulated Annealing EM Algorithm for Mixtures." Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz). Springer, Cham, 2015.

- Chapter 3 shows that task parameterized skill learning can be performed by modeling the forcing terms of a DMP with a mixture of GMMs. The prescribed combination raises an overfitting issue, which is resolved by filling the input space with incomplete data. An EM algorithm is devised to handle the additional incomplete data. The proposed approach can interpolate as well as as extrapolate beyond the demonstrated limits of task parameters. The performance of the presented approach is validated through real robot experiments. The material presented in this chapter is published in:

  Affan Pervez, and Dongheui Lee "Learning task-parameterized dynamic movement primitives using mixture of GMMs." Intelligent Service Robotics (2017): 1-18.

- Chapter 4 presents an approach for task specific skill learning by utilizing deep learning. In this chapter it is shown that the forcing terms of a DMP can be modeled with a CNN. This enables the DMP to directly utilize high dimensional images as input. The proposed architecture possesses all the desirable attributes associated with a DMP. As compared with a Task Parameterized DMP model, the use of a CNN eliminates the need to extract task parameters for motion reproduction. The presented approach shows robustness against online perturbation of the object's position, change of goal position and the perturbation in camera images. Its performance is validated through real robot experiments. The material presented in this chapter is published in:

  Affan Pervez, Yuecheng Mao, and Dongheui Lee "Learning Deep Movement Primitives using Convolutional Neural Networks." 17th International Conference on Humanoid Robotics (Humanoids). IEEE-RAS, 2017.

- Chapter 5 present approaches for applying PbD to teleoperation. Firstly the problems encountered when applying the existing PbD approaches to teleoperated demonstrations is discussed. Secondly an EM based algorithm for simultaneously aligning and encoding demonstrations is presented. The presented approach shows superior performance when compared with separate alignment and encoding strategies. It is also shown that the learned model can be also utilized in shared teleopearation settings, where the task is simultaneously executed by the learned model and the human operators. The performance of the presented approaches is evaluated on a master-slave teleoperation setup. The material presented in this chapter is published in:

  Affan Pervez, et al. "Novel learning from demonstration approach for repetitive teleoperation tasks." World Haptics Conference (WHC), IEEE, 2017.

  Affan Pervez, et al. "Human-Agent Shared Teleoperation: A Case Study Utilizing Haptic Feedback." Asia Haptics, 2018."

  *Affan Pervez, et al. "Motion Encoding with Asynchronous Trajectories of Repetitive Teleoperation Tasks and its Extension to Human-Agent Shared Teleoperation." **(under progress)***

- Chapter 6 concludes the thesis by summarizing all the works presented in the earlier chapters and highlights the main findings of this thesis. A general idea about the connection of the earlier chapters is also presented in this chapter. Lastly the future working directions, for improving and enhancing the material presented in this thesis, are also identified and discussed in this chapter.

1.4. Contributions

Following are the contributions of the works presented in different chapters.

- Chapter 2 presents the approaches for initializing and fitting GMMs. The initialization approach is presented in Section 2.3 which provides a good initial model for applying EM. This is verified by utilizing a model selection criterion [30]. The Componentwise Simulated Annealing EM for Mixtures (CSAEM^2) algorithm is presented in Section 2.4. The presented approaches outperform the approaches presented in [30, 31].
- Chapter 3 presents an approach for learning Task Parameterized-DMP (TP-DMP). For modeling the forcing terms of a DMP, it uses a generative model instead of a discriminative model, as shown in Section 3.3. The local maxima problem during the likelihood maximization is resolved with DAEM, as shown in Section 3.3. The data scarcity problem is solved by using additional incomplete data and the Expectation Maximization (EM) algorithm [28], as shown in Section 3.4. The detailed derivation of the incomplete data EM is also provided in the Appendix. Through simulated and real robot experiments, it is shown that the presented approach requires very few demonstrations for learning and provides superior interpolation as well as extrapolation capabilities, as shown in Section 3.5. The presented approach outperforms the approaches presented in [21, 33, 104, 109].
- Chapter 4 presents an approach for learning Deep-DMP (D-DMP). The D-DMP approach is presented in Section 4.2 . It utilizes the camera images directly to learn a motion primitive. D-DMP also eliminates the need to first extract task parameters before the motion reproduction, as shown in Section 4.2. Section 4.3 shows that the proposed approach preserves the generalization aspects associated with a DMP model while the generalization capabilities associated with feature learning are exploited by using a CNN. The presented approach performs better than the approaches presented in [62, 63, 112].
- Chapter 5 presents an approach for PbD in teleoperation. The PbD approach for simultaneously synchronizing and encoding teleoperated demonstrations is presented in Section 5.3. Section 5.4 shows that the same model can be utilized for shared teleoperation control, where the motion is simultaneously executed by a learned controller and the human operators. The presented approach outperforms the approaches presented in [93, 94].

1.5. Conclusion

This chapter covers a broad overview of the thesis. It also highlights the motivation of this thesis. A general overview of the chapters to follow is presented here i.e. GMM based density estimation, task parameterized learning for PbD, deep learning for Pbd and PbD for teleoperated demonstrations. A detailed discussion and the experimental results of the topics introduced in this chapter is presented in the following chapters.

CHAPTER 2

Gaussian Mixture Model and the EM algorithm

2.1. Motivation

This chapter addresses the problem of fitting finite Gaussian Mixture Model (GMM) with unknown number of components to the univariate and multivariate data. GMM based density estimation is a preliminary step of the imitation learning strategies presented in the later chapters. That is why this chapter considers this problem first. The typical method for fitting a GMM is Expectation Maximization (EM). There are many challenges involved when applying EM for fitting a GMM; such as how to initialize the GMM, how to restrict the covariance matrix of a component from becoming singular and setting the number of components in advance. Here a simulated annealing EM algorithm is presented and its usefulness in avoiding convergence to the boundary of the parameter space is presented. Additionally a systematic initialization procedure by using the principals of stochastic exploration is presented. The experiments demonstrates the robustness of the approach on different datasets.

2.2. Overview

2.2.1. Gaussian Mixture Models

A GMM with k components is parameterized by $\boldsymbol{\Theta}_{(k)} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{k}$ where $\pi_1, \ldots, \pi_k$ are mixing coefficients / priors, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ are means and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$ are covariance matrices. The probability density function of $\mathbf{y}$ is said to follow the k-component GMM if it can be written as $\mathcal{P}(\mathbf{y}|\boldsymbol{\Theta}_{(k)}) = \sum_{m=1}^{k} \pi_m \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, subject to the constraints $0 < \pi_m < 1$ and $\sum_{m=1}^{k} \pi_m = 1$. The log-likelihood of a dataset $\mathbf{y}_{\mathtt{obs}} = \{\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(n)}\}$ for a GMM is defined as:

$$\mathcal{L}(\boldsymbol{\Theta}_{(k)}, \mathbf{y}_{\mathtt{obs}}) = \sum_{i=1}^{n} log \sum_{m=1}^{k} \pi_m \mathcal{N}(\mathbf{y}^{(i)}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{2.1}$$

2.2.2. The EM algorithm

EM is an iterative procedure for estimating the ML value of the parameters of a probabilistic model with latent variables [28]. At each iteration there is an old estimate of the parameters and the goal

is to find a new estimate of the parameters such that the new parameters fit the data better than the old estimate. The EM formulation considers $\mathcal{Y}$ as incomplete data with the hidden labels $\mathcal{Z}$. The parameters are updated by cycling through the E-step and the M-step. In the E-step, the expected value of complete data log-likelihood is calculated using the old parameters estimate, which is used for maximization in the M-step. The update equations of EM for a GMM are thoroughly covered by Bishop (2006). For avoiding singularity, it is a common practice to add a small regularization term $\lambda\mathbf{I}$ in all covariance matrics of the GMM at each update cycle. The EM is terminated when the increase in log-likelihood becomes very small i.e. $\frac{\mathcal{L}^{t+1}}{\mathcal{L}^t} - 1 < \varepsilon$.

2.3. Stochastic exploration initialization approach

2.3.1. Inspiration of the proposed approach

Many nature inspired algorithms exist for solving discrete optimization problems [53]. It is a well known fact that humans by nature are social animals. A person who is alone will tend to find other people and will stay in a community, if possible. Separation from community can induce separation anxiety disorder. Not only humans but animals also exhibit such behavior [52]. Even cockroaches which prefer to live in dark socialize with nearby cockroaches [39]. Among hiding and searching for food, cockroaches also undergo through friend finding phase and this can be considered as one of their basic behaviors [39]. Also many animals exibit anti-predator adaptation through group living. Given this, now consider a scenario where an individual is alone at a location. Seeking someone is important due to the above mentioned reasons. If there is no heuristic about the surrounding then with equal probability it will start to look in any direction. Suppose that there are many individuals like this at a place, what will happen when one comes in interaction with someone else? Most probably its search pace will decrease, since living in a group provides more stability. Now they also have a heuristic about the search direction. The future search will be more focused toward the directions which have already yielded positive results.

2.3.2. Exploration phase

Surprisingly it is not difficult to import the proposed approach for initializing a GMM. Now each observation will act like an individual so a Gaussian is placed at each observation. The perception of each individual is encoded in its covariance matrix. In the beginning the covariance matrix is diagonal, with small equal positive value δ in the diagonal. A fraction of minimum positive distance in between data points can be used as δ .The small value of δ reflects the limited know-

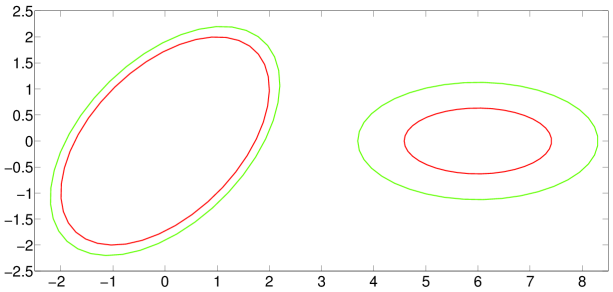


Figure 2.1.: A single exploration step with the red ellipses as the original GMM with a larger prior value for the left component. The green ellipses representing the GMM after exploration.

how of each individual about their surrounding and equality projects the identical information about all directions. The prior value of each component is $\frac{1}{n}$ at this stage where n is the total number of observations. After this the next step is exploration phase. The exploration is done by slightly expanding each component and then testing that whether it has sufficient overlap with one of its neighbouring Gaussians. The exploration step can be set as a fraction of minimum positive distance between the mean of the clusters. The rate of expansion of each Gaussian is inversely proportional to its prior value. This corresponds to slow exploration for components with high prior values and fast exploration for components with low prior values. For expanding a Gaussian, an exploration term is added in the standard deviation (σ) of each eigenvector of its covariance matrix. The magnitude of each exploration term is proportional to its corresponding σ. This corresponds to higher exploration in major principal axis. It is worth noting that the shape of a Gaussian is preserved by this approach but is slightly expanded. The prior values for left and right components in Figure 2.1 are 0.8 and 0.2 respectively and it can be observed that higher exploration is performed in the major principal axis of each component and the component with a low prior value explores more than the other. The expansion of each Gaussian correspond to its exploration and hence an increase in perception of the surrounding.

2.3.3. Merging mechanism during exploration phase

Once two Gaussians have sufficient overlap they can be merged. For detecting overlap between two components a coefficient $\mathbf{C}$ is defined. When two Gaussians are multiplied then the resulting density is again a Gaussian but multiplied with this coefficient [80] i.e. $\mathcal{N}(\boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_Q) \times \mathcal{N}(\boldsymbol{\mu}_R, \boldsymbol{\Sigma}_R) = \mathbf{C}_{QR}\mathcal{N}(\boldsymbol{\mu}_S, \boldsymbol{\Sigma}_S)$ where $\mathbf{C}_{QR} = \mathcal{N}(\boldsymbol{\mu}_Q; \boldsymbol{\mu}_R, \boldsymbol{\Sigma}_Q + \boldsymbol{\Sigma}_R)$. The $\mathbf{C}$ has a property that as the Gaussians expand via exploration, it keeps on increasing and then at a certain point it starts to decrease. As the $\mathbf{C}$ passes its peak value, which is detected as its value decreases in the next iteration, the Gaussians are merged. So the metric of closeness is determined by using the Gaussians and not just by the mere euclidean distance. The mentioned property of C can intuitively be reasoned by considering the pdf value of a point (other than mean) for a given Gaussian (for e.g. $\boldsymbol{\mu}_Q$). This pdf value goes to zero if covariance of the Gaussian ($\boldsymbol{\Sigma}_Q + \boldsymbol{\Sigma}_R$) either converges to the mean ($\boldsymbol{\mu}_R$) or becomes infinite in all axis. Intermediate values of covariance result in positive pdf values. This explains the behavior of this function i.e. first to increase and then to decrease (transition of $\mathbf{C}$ from $0 \rightarrow + \rightarrow 0$).

The merging of two Gaussians is a well posed problem [114] and the closed form solution exists. The parameters of the gaussian P formed by merging gaussians Q and R are described in Equation (2.2).

$$\begin{aligned} \pi_P &= \pi_Q + \pi_R \\ \boldsymbol{\mu}_P &= \frac{\pi_Q\boldsymbol{\mu}_Q + \pi_R\boldsymbol{\mu}_R}{\pi_P} \quad \boldsymbol{\Sigma}_P = \frac{\pi_Q(\boldsymbol{\Sigma}_Q + \boldsymbol{\mu}_Q\boldsymbol{\mu}_Q^\top) + \pi_R(\boldsymbol{\Sigma}_R + \boldsymbol{\mu}_R\boldsymbol{\mu}_R^\top)}{\pi_P} - \boldsymbol{\mu}_P\boldsymbol{\mu}_P^\top \end{aligned} \tag{2.2}$$

The exploration and merging cycles continue until the component count reaches a predifined value k_{max} where $k_{max} \gg k_{optimal}$. k_{max} can be manually set manually or results about the upper bound on the number of components [111] can be used for setting k_{max} as a multiple of upper bound on components.

Table 2.1.: Algorithm for GMM initialization by Stochastic Exploration approach

Input: $\mathbf{y}_{\mathtt{obs}}, k_{max}$ ▷ To speedup, $\mathbf{y}_{\mathtt{obs}}$ = resampled data and $n = k_{start}$
Set $k \leftarrow n$, $p = 0$, R=20
for m:=1 $\rightarrow n$ **do**
  $\pi_m = \frac{1}{n}$, $\boldsymbol{\mu}_m = \mathbf{y}^{(m)}$, $\boldsymbol{\Sigma}_m = \delta^2 \mathbf{I}$ ▷ $\delta = min(D > 0)/R$ where D is a $n \times n$ matrix with $D(q,r) = \|\mathbf{y}^{(q)} - \mathbf{y}^{(r)}\|_2$
end for
$\boldsymbol{\Theta}_{(k)} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{k}$
while $k > k_{max}$ **do**
  define $k \times k$ matrix D with $D(q,r) = \|\boldsymbol{\mu}_q - \boldsymbol{\mu}_r\|$
  set $ExpStep = (min(D > 0) \times k)/R$
  set $ExpProp = |\pi - \max(\pi) - \min(\pi)|$
  for m:=1 $\rightarrow k$ **do**
    $[EigVal, EigVec] = eig(\boldsymbol{\Sigma}_m)$ ▷ Extract eigenvalues and eigenvectors
    for s:=1 $\rightarrow d$ **do**
      $EigVal(s) = \left(\sqrt{EigVal(s)} + ExpProp(m) \times ExpStep \times \frac{\sqrt{EigVal(s)}}{sum(\sqrt{EigVal})}\right)^2$
    end for
    $\boldsymbol{\Sigma}_m = reconstruct(EigVal, EigVec)$ ▷ Reconstruct covariance after adding exploration term
  end for
  if $(MergeCriterion(\boldsymbol{\Theta}_{(k)}) == True)$ **then**
    $\boldsymbol{\Theta}_{(k-r)} \leftarrow Merge(\boldsymbol{\Theta}_{(k)})$ ▷ Merge $2r$ components that fulfill the merge criterion
    set $k \leftarrow k - r$
  end if
end while
return $\Theta_{(k)}$

2.3.4. Computational complexity

The pseudo-code of the stochastic exploration initialization approach is presented in Table 2.1. For a k component GMM one has to do $\frac{k(k-1)}{2}$ calculations of $\mathbf{C}$ for each exploration step. Since in the beginning of the algorithm $k = n$, this can pose significant computation load. One way of swiftly doing these calculations is to implement them on a GPU. The calculation of $\mathbf{C}$ among different components of a GMM are completely independent of each other and can be executed in parallel. An elegant way to bound the computational load is to sample (without replacement) k_{start} observations from the dataset where $k_{start} < n$. The number of observations representing a GMM component are proportional to its weight value and thus in the downsampled data, the number of observations selected from those observations should also reflect the component's weight. The proposed initialization procedure can then be applied on this downsampled data and thus the computation load of the initialization procedure becomes independent of the total number of observations n and is bounded by k_{start}. So now for the Stochastic Exploration initialization approach the algorithmic complexity will be $\mathcal{O}(k_{start}^2)$ instead of $\mathcal{O}(n^2)$.

2.4. Component-Wise Simulated Annealing EM for Mixtures

The proposed CSAEM^2 algorithm is the combination of Component-Wise EM for Mixtures (CEM^2) and simulated annealing algorithm. As mentioned by [23] that CEM^2 is a serial decomposition of optimization problem with a coordinatewise maximization. It goes through E-step and then applies M-step to update only one component at a time. The components can be updated adaptively but for simplicity they are updated sequentially, by going from 1 to k and then starting again from 1. [31] showed that in terms of computational burden, CEM^2 is only slightly more expensive than usual EM with batch update of parameters. The component-wise update also avoids the gaussian from becoming singular, as is discussed later.

For simulated annealing there is a temperature parameter τ which is gradually decreased to zero. The temperature value defines the probability of taking annealing step. The initial value of temperature is set to a small value as higher value of temperature can completely disturb the discovered GMM. During annealing iterations only one component is updated at a time.

$$\gamma_{q,i} = \frac{\left(\pi_q \mathcal{N}(y_i;\mu_q,\Sigma_q)\right)^{\Phi_q}}{\sum_{l=1,l\neq q}^{k} \pi_l \mathcal{N}(y_i;\mu_l,\Sigma_l) + \left(\pi_q \mathcal{N}(y_i;\mu_q,\Sigma_q)\right)^{\Phi_q}}$$

where $0 \leq \Phi_q \leq 1$. The amount of annealing Φ_q is inversely proportional to the weight of a component q and is calculated as $\Phi_q = \frac{\pi_q}{\max(\pi)}$. The annealing induces the fuzziness in the membership of the components with no annealing $(\max(\Phi) = 1)$ for the component with highest weight and highest annealing $(\min(\Phi))$ for the component with lowest weight. The annealing step usually decrease the likelihood value which is analogous to suboptimal / random step in standard simulated annealing approach but it has some useful properties. Firstly it increases the survival probability of weak components (components with small weights) by increasing their weights and spread. Secondly it redistribute the weights of the components more evenly. The mentioned properties are more thoroughly discussed in section 2.5.2.

A component can be removed during CSAEM^2 if its value become too small, (Section 2.4.1) which is an indication of singularity like situation. If CSAEM^2 converges (with respect to likelihood value) then the $(k-1)$ components GMM can be obtained by simply merging two components of the k components GMM. All possible $(k-1)$ component GMMs are generated and the one which yield highest likelihood value is selected. CSAEM^2 and components merging are repeatedly applied until the components count k reaches k_{min} (usually one). The parameters $\hat{\Theta}_{(k)}$ of all the models generated after applying CSAEM^2 are stored as candidate GMMs. The pseudo-code of the CSAEM^2 algorithm is presented in Table 2.2.

2.4.1. Component annihilation condition during CSAEM^2

Similar to [113], a component removal mechanism for avoiding convergence to the boundary of the parameter space is introduced. During EM a components prior (π_m) may approach to $\frac{1}{n}$ and its covariance matrix may become almost singular. As EM is a local optimization algorithm, this type of problem is inherent in EM based approaches and can make the likelihood value unbounded. A component can be eliminated if it becomes too weak i.e. its π_m becomes very small. In this work a component m is removed if $n\pi_m < \alpha(d+1)$ where $\alpha \geq 1$. As mentioned by [67], a component requires the support of at least $d+1$ observations for avoiding the covariance matrix to become singular i.e. $n\pi_m \geq (d+1)$. Apart from that, those observations should not be on a hyperplane. $\alpha = 2$ is used in the experiments for insuring that the covariance matrix of each component stays away from becoming singular (the higher the value of α is, the less probable the points will be

Table 2.2.: Algorithm for Componentwise Simulated Annealing EM for Mixtures (CSAEM^2)

Input: $\boldsymbol{\Theta}_{(k)} = \{\pi_l, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\}_{l=1}^{k}$
Define: $AnnIter, \tau, \varepsilon, \alpha, \lambda$
$m = 1, \ldots, k \quad i = 1, \ldots, n$
$u_{m,i} = \mathcal{N}(y_i; \mu_m, \Sigma_m)$
$Temp = \tau, iter = 1, \mathcal{L}^0 \leftarrow 0$
while 1 **do**
  $\Phi = \pi / \max(\pi)$
  $rem = 0, Rnd \sim U(0,1)$ ▷ Sample from Uniform distribution
  for q:=1 $\rightarrow k$ **do**
    if $\big((iter < AnnIter)$ & $(Rnd < Temp)\big)$ **then**
      $\gamma_{q,i} = \frac{\left(\pi_q u_{q,i}\right)^{\Phi_q}}{\sum_{l=1, l\neq q}^{k} \pi_l u_{l,i} + \left(\pi_q u_{q,i}\right)^{\Phi_q}}$ ▷ Annealing
    else
      $\gamma_{q,i} = \frac{\pi_q u_{q,i}}{\sum_{l=1}^{k} \pi_l u_{l,i}}$
    end if
    $E_q = \sum_{g=1}^{n} \gamma_{q,g}$
    $\pi_q = \frac{E_q}{n}$
    if $\pi_q < \frac{\alpha \times (d+1)}{n}$ **then**
      $\pi_q = 0$ ▷ Kill component
      $rem = rem + 1$
    else
      $\mu_q = \frac{\sum_{g=1}^{n} \gamma_{q,g} y_g}{E_q}$
      $\Sigma_q = \frac{\sum_{g=1}^{n} \gamma_{q,g} (y_g - \mu_q)(y_g - \mu_q)^\top}{E_q} + \lambda \mathbf{I}$
      $u_{q,i} = \mathcal{N}(y_i; \mu_q, \Sigma_q)$
    end if
    $\pi = \pi / sum(\pi)$ ▷ Immediately redistribute weights
  end for
  if $rem > 0$ **then**
    $\boldsymbol{\Theta}_{(k-r)} = RemComp(\boldsymbol{\Theta}_{(k)})$ ▷ Remove killed components
    $k = k - r$
  else
    if $\left(\left|\frac{\mathcal{L}^{iter}}{\mathcal{L}^{iter-1}} - 1\right| < \varepsilon \;\; \& \;\; iter > AnnIter\right)$ **then**
      $Break\_While$
    end if
  end if
  $Temp = \min\left(0, \tau - \frac{\tau \times iter}{AnnIter}\right)$
  $iter = iter + 1$
end while
return $\Theta_{(k)}, k$

Table 2.3.: The complete algorithm

As described in Table 2.1, apply the Stochastic Exploration initialization procedure to get the $k = k_{max}$ component GMM $\boldsymbol{\Theta}_{(k)}$
while $k \geq k_{min}$ **do**
  $\{\hat{\boldsymbol{\Theta}}_{(k)}, k\} \leftarrow \text{CSAEM}^2(\mathbf{y}_{\texttt{obs}}, \boldsymbol{\Theta}_{(k)})$ ▷ CSAEM^2 algorithm in Table 2.2
  Calculate and store $\mathcal{C}\left(\hat{\boldsymbol{\Theta}}_{(k)}, k\right)$ and $\hat{\boldsymbol{\Theta}}_{(k)}$
  Switch to $(k-1)$ component GMM $\boldsymbol{\Theta}_{(k)}$ yielding highest likelihood value
end while
$\hat{k} = \text{argmin}_k \left\{\mathcal{C}\left(\hat{\boldsymbol{\Theta}}_{(k)}, k\right), k = k_{min}, \ldots, k_{max}\right\}$
return $\hat{\boldsymbol{\Theta}}_{(\hat{k})}$

on a hyperplane). If the component deleting strategy is used with general EM algorithm then it has a failure mode. As mentioned by [31], with a large value for k_{max} many or all components may get removed simultaneously. With the CEM^2, if a component is removed, its weight is immediately distributed among other components, thus increasing the survival probability of the other components.

2.4.2. Model selection criterion

Once there are a certain set of candidate models with the different number of components ranging from k_{min} to k_{max}, the next task is to select the optimal model among them. The log-likelihood value $\mathcal{L}$ of a dataset for a given GMM cannot be used directly as it is an increasing function of model complexity k. There are a wide variety of model selection criteria [31]. The role of any criterion is to minimize some objective function $\mathcal{C}$ for the optimal number of components i.e. $\hat{k} = \text{argmin}_k \left\{\mathcal{C}\left(\hat{\boldsymbol{\Theta}}_{(k)}, k\right), k = k_{min}, \ldots, k_{max}\right\}$.

Mixture Minimum Description Length (MMDL) criterion is used here [30]. It is similar to the *Bayesian Information Criterion* (BIC) [101] and *Minimum Description Length* (MDL) [87] but has an additional term. This additional term allows extra components for representing skewness. A skew Gaussian requires more than one components for representation. [30] showed that MDL and BIC can underestimate the number of components by representing a skew Gaussian with a single component. The MMDL criterion for a GMM is defined as

$$\mathcal{C}_{\texttt{MMDL}}\left(\hat{\boldsymbol{\Theta}}_{(k)}, k\right) = -\mathcal{L}(\hat{\boldsymbol{\Theta}}_{(k)}, \mathbf{y}) + \frac{N(k)}{2}\log n + \frac{N(1)}{2}\sum_{m=1}^{k}\log(\pi_m)$$

where $N(k)$ is the number of free parameters in a k component GMM. For an unrestricted GMM the number of free parameters are $N(k) = (k-1) + (kd) + \frac{kd(d+1)}{2}$. The complete algorithm for GMM initialization and fitting is presented in Table 2.3.

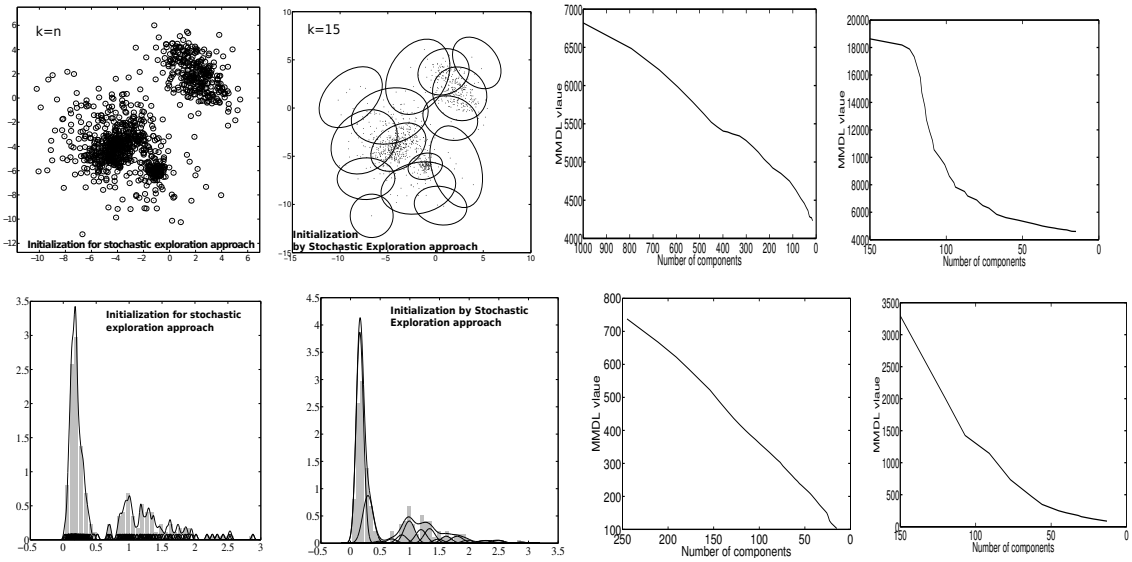


Figure 2.2.: Initialization results of synthetic (first row) and enzyme (second row) datasets with stochastic exploration approach. First column shows the $k = n$ component GMMs. Second column shows the reduced 15 components GMMs with the Stochastic exploration initialization approach. Third column shows the evolution of MMDL value when complete datasets are utilized. Last column shows the evolution of MMDL value when downsampled datasets are utilized.

2.5. Comparison with State-of-the-Art

2.5.1. Results of Stochastic exploration initialization approach

Figure 2.2 shows the results of stochastic exploration approach on the synthetic dataset (section 2.5.2) and the enzyme dataset [86] (from top to bottom in Figure 2.2 respectively). The first column contains the highly overfitted $k = n$ component GMMs while the second column contains the simplified $k = k_{max}$ component GMMs obtained by stochastic exploration. In the next step these k_{max} component GMMs are used for initializing CSAEM^2. Since the GMMs obtained by the proposed initialization approach have evolved from the data itself, they provide a good representation of the distribution. The validity of this claim is demonstrated by the last two columns of Figure 2.2. The third column contains the MMDL value for the GMMs formed during transition from the $k = n$ component GMM to the $k = k_{max}$ component GMM. The initialization approach consistently decreases the MMDL value and thus improves the model representation while decreasing the number of components. The same behavior can be observed in the last column which contains the MMDL value for the GMMs formed during transition from the $k = k_{start}$ component GMM to the $k = k_{max}$ component GMM. The MMDL value was calculated by using all of the n observations and not just the downsampled k_{start} observations. It can be noticed that the stochastic exploration approach still decreases the MMDL value, which means improving model representation while decreasing the component count k. Hence this demonstrates the usefulness of this approach for providing a good initial model representation and its robustness against starting from $k = k_{start}$ components instead of $k = n$ components. As the initialization approach is based on the idea of exploration without any maximization steps, it cannot encounter the overfitting issues.

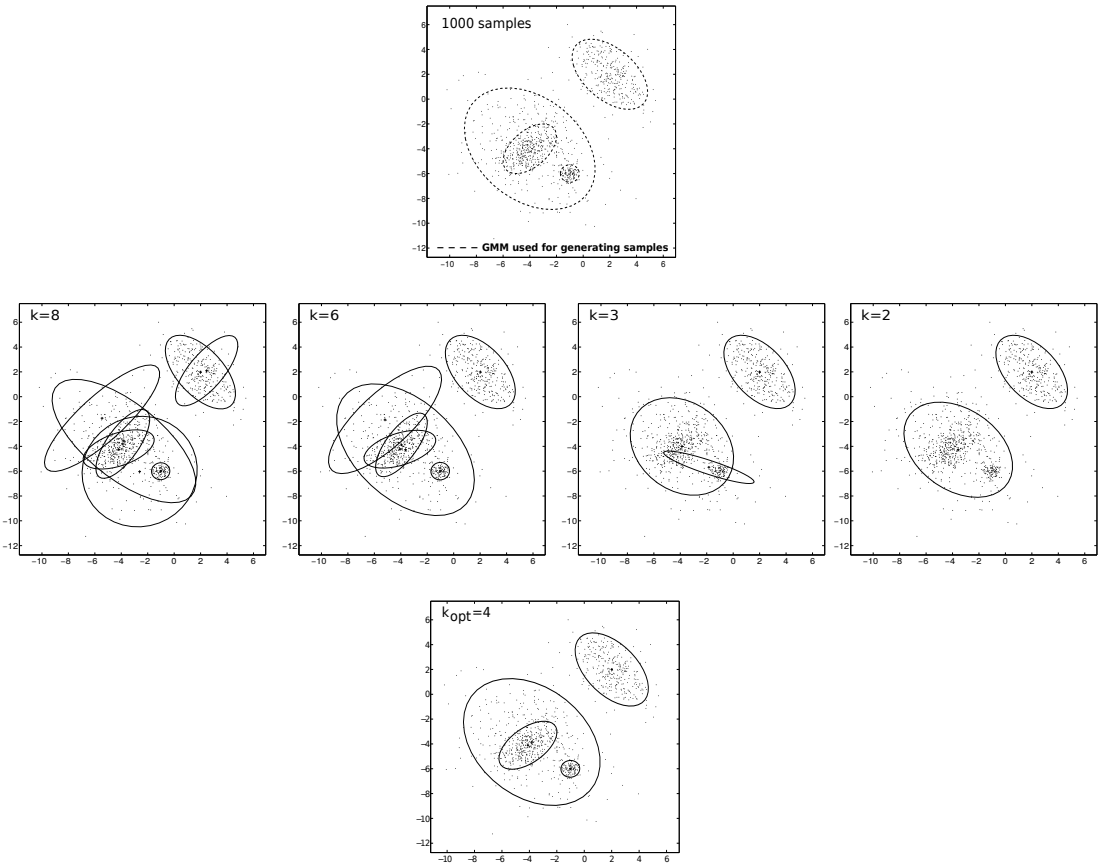


Figure 2.3.: Evolution of GMM for synthetic dataset.

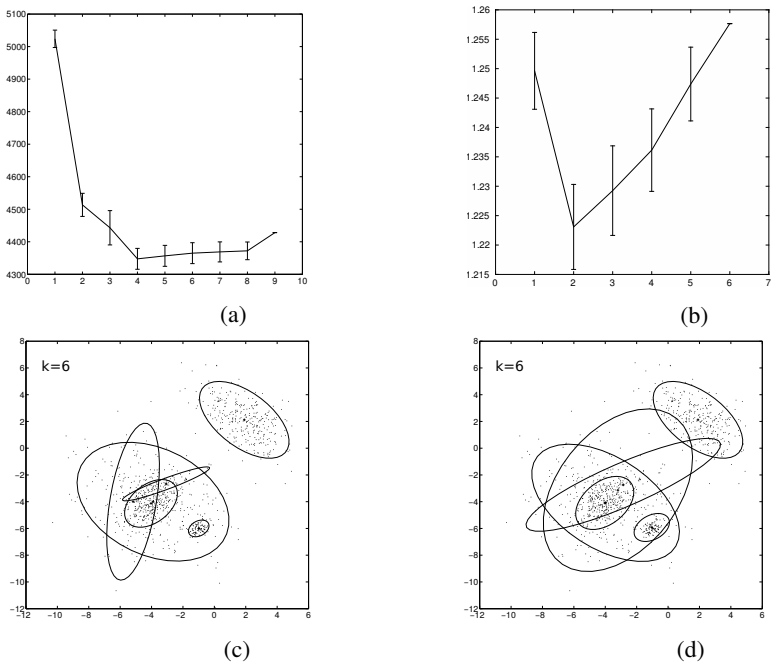


Figure 2.4.: (a) Result of 50 Monte Carlo simulations on synthetic Dataset 1 showing mean MMDL value (of 50 experiments) for components, with vertical bars depicting standard deviation in each value. (b) Same as (a) but for Dataset 2. (c) GMM just before and (d) after annealing step.

2.5.2. Results of CSAEM^2

All the experiments (except synthetic Dataset 2) described below uses the following parameter values $k_{start} = 150, k_{max} = 15, k_{min} = 1, MaxIter = 600, AnnIter = 400, \tau = 0.05, \varepsilon = 10^{-5}, \alpha = 2, \lambda = max(10^{-4}, min(D > 0))$ where D is a $n \times n$ matrix with $D(Q,R) = \|\mathbf{y}^{(Q)} - \mathbf{y}^{(R)}\|^2$.

Experiment with synthetic dataset

Dataset 1: 1000 samples were drawn from the four component bivariate GMM with

$$\pi_1 = \pi_2 = \pi_3 = 0.3, \pi_4 = 0.1, \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \begin{bmatrix} -4 \\ -4 \end{bmatrix}, \boldsymbol{\mu}_3 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \boldsymbol{\mu}_4 = \begin{bmatrix} -1 \\ -6 \end{bmatrix}$$
$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \boldsymbol{\Sigma}_2 = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}, \boldsymbol{\Sigma}_3 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \boldsymbol{\Sigma}_4 = \begin{bmatrix} 0.125 & 0 \\ 0 & 0.125 \end{bmatrix}.$$

This GMM has also been used by [31], [113] and [64]. It provides a challenging scenario of overlapping components with two components having a common mean. The evolution of the GMM for an experiment can be visualized in Figure 2.3. A Monte Carlo (MC) simulation of 50 experiments is performed. The results are shown in Figure 2.4a. In all experiments the presented algorithm identified the right four component GMM.
Dataset 2: 800 samples were drawn from the two component 10 dimensional GMM with $\pi_1 = \pi_2 = 0.5, \boldsymbol{\mu}_1 = [0,\ldots,0]^\top, \boldsymbol{\mu}_2 = [2,\ldots,2]^\top, \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \mathbf{I}$. The GMM contain high dimensional fused components. Since it is high dimensional data, $k_{start} = 250$ is used for this experiment. Figure 2.4b contains the result of 50 MC experiments and the presented approach correctly identified the two component GMM in all of them.
Effect of annealing: The effect of annealing during CSAEM^2 can be observed in Figure 2.4(c-d). The annealing increases the coverage and thus the survival probability of the weak components while the components with high prior values are unaffected by annealing. Another interesting property of annealing is depicted in Figure 2.5. The maximum value of $f = \sum_{i=1}^{k} \log(\pi_i)$ is attained when $\pi_i = \frac{1}{k}$. This can easily be proven by solving the constrained maximization problem as $\sum_{i=1}^{k} \log(\pi_i) + \nu(1 - \sum_{i=1}^{k} \pi_i)$ where ν is the Lagrange multiplier. Similarly the minimum value of $g = \sum_{i=1}^{k} (\frac{1}{k} - \pi_i)^2$ is attained when $\pi_i = \frac{1}{k}$. This can also be proven by solving the constrained maximization problem as $\sum_{i=1}^{k} (\frac{1}{k} - \pi_i)^2 + \nu(1 - \sum_{i=1}^{k} \pi_i)$. Moreover since the Hessian of f is negative-definite while that of g is positive-definite, the functions are strictly concave and strictly convex respectively. Thus the more equally π values are distributed, the higher would be f. It can be observed in Figure 2.5 that there is a sharp decrease in the value of g and a sharp increase in the value of f after annealing cycles (Dataset 1). Thus annealing has a tendency of redistributing the values of π more equally.

Experiment with Real datasets:

Now Acidity and Enzyme datasets are considered as they have skewed gaussians. They have been extensively studied by [86] and their optimal number of components are three and four respectively [66, 86]. A Monte Carlo simulation of 50 experiments is performed and in all the experiments, CSAEM^2 detected the same three and four component GMMs as shown in Figure 2.6(a-b). Then a well known relatively higher (four) dimensional Iris dataset [7] is also considered. The

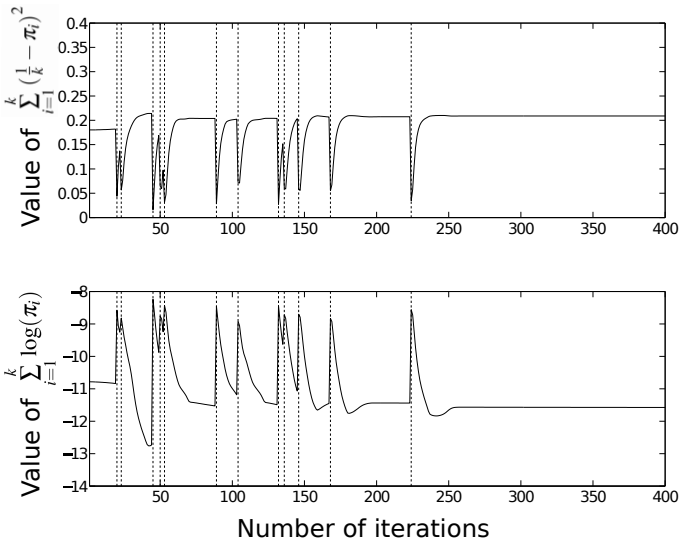
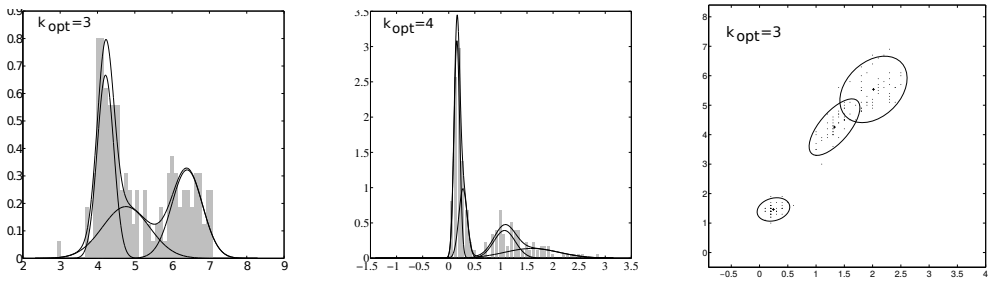


Figure 2.5.: The value of $\sum_{i=1}^{k}(\frac{1}{k}-\pi_i)^2$ (upper figure) and $\sum_{i=1}^{k}\log(\pi_i)$ (lower figure) during the annealing iterations of CSAEM^2. Vertical dashed lines depict the instances when annealing was applied.



(a) Optimal univariate GMM with three components

(b) Optimal univariate GMM with four components

(c) Optimal bivariate GMM with three components

Figure 2.6.: Results of CSAEM^2 on (a) Acidity, (b) Enzyme and (c) Iris datasets. In (a-b), data is encoded in histograms while (c) contains the projection of data on the two axis with highest variances

Table 2.4.: Percentage frequency of choosing k clusters for 50 experiments by the proposed approach and the approaches presented by Figueiredo et al. (1999,2002).

k	Presented method	Figueiredo et al. (2002)	Figueiredo et al. (1999)
3	0	0	1
4	47	1	17
5	3	13	16
6	0	14	7
7	0	10	4
8	0	8	2
9	0	2	2
10	0	2	1

dataset contains three classes with 50 samples for each class. Again a Monte Carlo simulation of 50 experiments is performed and with hundred percent success rate detected the three component GMM as shown in Figure 2.6(c).

Comparison with similar EM approaches: [30, 31] has presented similar EM based approaches where the components count starts from k_{max} and are brought down to k_{min}. These algorithms explicitly target the components with low prior values for switching to a $(k-1)$ component GMM. For e.g. in [30] a component is forced to merge if its weight value decreases below $\frac{5d}{n}$. Similarly in [31] a component with minimum prior value is removed for proceeding to a $(k-1)$ component GMM. Due to this reason these algorithms often fail when there is a component with very low prior value [31] while the presented approach increases the survival probability of the weak components and thus overcomes this problem.

Now a Monte Carlo simulation of 50 experiments is performed. 710 samples were drawn from the four component bivariate GMM with

$$\pi_1 = \frac{10}{71}, \pi_2 = \frac{20}{71}, \pi_3 = \frac{40}{71}, \pi_4 = \frac{1}{71}, \boldsymbol{\mu}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \boldsymbol{\mu}_3 = \begin{bmatrix} 3 \\ -3 \end{bmatrix}$$
$$\boldsymbol{\mu}_4 = \begin{bmatrix} 8 \\ 8 \end{bmatrix}, \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_4 = \mathbf{I}, \boldsymbol{\Sigma}_3 = \begin{bmatrix} 2 & 1.9 \\ 1.9 & 2 \end{bmatrix}, \boldsymbol{\Sigma}_2 = \begin{bmatrix} 2 & -1.5 \\ -1.5 & 2 \end{bmatrix}.$$

All algorithms used same $k_{max}, k_{min}, \varepsilon$ and λ values. Although the components are quite well separated from each other, the approaches presented in [30, 31] performed very poorly while the presented algorithm outperformed these methods as can be seen in Table 2.4.

Robustness against k_{max}: Choosing $k_{max} \gg k_{optimal}$ provides robustness against initialization issues. Since the value of k_{max} has to be specified by the user, it can be underestimated. Now the performance of the presented approach is tested with relatively smaller values of k_{max} for the simulated dataset with a weak component. For comparison, the initial k_{max} component GMM is obtained with two methods: the random initialization procedure presented in Figueiredo et al. (2002) and the presented initialization procedure. The results are summarized in Table 2.5. It can be observed that better results are obtained with the described procedure as compared to random initialization. Furthermore the presented algorithm has high frequency of detecting right number of components, even when starting with relatively smaller k_{max}. This algorithm clearly surpassed the approaches presented in [30, 31], whose results are mentioned in Table 2.4, even with random initialization and smaller k_{max} values. As k_{max} is increased, the frequency of detecting $k_{opt} = 4$ components is also increased for both random and my initialization procedures.

Table 2.5.: Percentage frequency of choosing $k_{opt} = 4$ clusters for 100 experiments by proposed approach with random initialization and stochastic exploration initialization.

k_{max}	Random initialization	Initialization by stochastic exploration approach
6	62	88
7	84	93
8	91	95
9	93	97

2.6. Discussion

This chapters introduces a novel GMM initialization and fitting algorithm. In the initialization procedure, each Gaussian applies search strategy to explore its surrounding. Instead of just utilizing a distance measure for merging Gaussians, this work proposes a merging criterion which utilizes the Gaussians. By using the means and the covariances, the proposed merging criterion provides an elegant way of detecting the overlap of the Gaussians. A minor drawback of utilizing the merging criterion based on the overlap of the Gaussians is that if two Gaussians have a common mean, they are often immediately combined as a single component. It is due to their high overlap during the initialization phase. This however is usually corrected by the EM steps applied later, as shown in the results of synthetic dataset 1 where the correct GMM is always identified.

Although the goal of EM is to increase the likelihood value, likelihood alone cannot be used as an indicator for selecting a model. This is due to the fact that the complex or overfitted models usually tend to have higher likelihood value. That is why model selection criterion are utilized to penalize models with high complexity. Results of Stochastic exploration initialization approach indicate that the initialization approach improves the model fit when analysed by using a model selection criteria. Apart from only testing for overlap, another way to merge components can be to also utilize the model selection criteria, as this can be helpful to preserve the components which share a common mean.

CSAEM^2 performs better than the approaches presented by Figueiredo et al. (1999) and Figueiredo et al. (2002). This is due to the fact that in their approaches they explicitly target the Gaussian with smallest prior to switch from a k component GMM to a $(k-1)$ component GMM. Due to this reason these approaches can often eliminate a Gaussian with low prior which is actually present in the real GMM. The presented approach firstly utilize the GMMs likelihood value to select the best fitting GMM. Secondly the simulated annealing cycles in the CSAEM^2 constantly redistributes the the weights of the components more evenly. This results in an increase in survival likelihood of weak Gaussians, as their weight are increased. In our approach a Gaussian with low prior is only eliminated if its weight becomes too low, which is detected if it falls below a predifined threshold.

One may wonder that when starting from k_{max} number of components, why the GMM first go down to only one component and then the GMM with lowest MMDL value is selected. Since the MMDL value can be readily calculated, one can immediately terminate the EM when it increases in a $(k-1)$ components GMM. This can reduce the number of iterations and thereby the overall computational time. The reason it is not done like this is that the plot of MMDL values do not always yield a convex curve. This means that its value can start to increase but can again decrease for the next model complexity. Hence the k_{min} is set to 1 so that all models of upto one component can be analysed.

2.7. Summary and Future works

In this chapter a novel nature inspired initialization approach for fitting a GMM is proposed. It utilizes a search strategy where each component looks for its nearby component. Two components are merged when they have a high overlap. Starting from a large number of components, the proposed approach reduces the complexity of a GMM and provides a transformation from an irregular to a smoother density function. CSAEM^2 is applied when the components count reaches k_{max}. A component is annihilated if it becomes too weak. $(k-1)$ components GMM is obtained by selecting the one which yields highest likelihood value. MMDL criterion is used to select the optimal model complexity. The proposed approach has shown promising results on challenging simulated and real datasets.

Following are the future working directions of the work presented in this chapter. Firstly the proposed initialization approach has been devised for GMMs. Ideas about how the exploration step and merging criterion for the initializtion can be applied to other forms of probability distributions have yet to be inquired. This involve deriving appropriate equations for those distributions. Secondly this work only considers merging steps to reduce the model complexity. If the k_{max} is underestimated to be less than k_{opt}, incorporating splitting steps by splitting a Gaussain into two Gaussians and increasing model complexity, if indicated by the model selection criterion, can also be investigated. The main challenge in implementing splitting of a Gaussian is that unlike merging two Gaussians, splitting a Gaussian does not have a closed form solution.

CHAPTER 3

Task parameterized skill learning

3.1. Motivation

Humans are very good at learning and reproducing complex tasks, but it is often tedious and cumbersome to program a robot for performing them. Programming by Demonstration (PbD) alleviates this problem, giving the possibility to teach a skill to a robot through demonstrations. The skill can be acquired from an expert in the corresponding field and removes the bottleneck for the teacher to have knowledge of robotics or programming. This also provides a great potential for industrial applications as PbD can significantly reduce the setting up time of an assembly line. Since PbD aims at speeding the setting up time, collecting a lot of demonstrations can be an expensive and time consuming task. Thus it is a desirable attribute to learn from as few demonstrations as possible. Moreover, since the demonstrations are finite, the learned controller should not only be able to generate motions within the demonstrated ranges, but also beyond them.

There are skills in which multiple demonstrations can look very different due to the underlying task specific variations [65, 103, 104]. As an example, Figure 3.1 presents a sweeping task. In this task, the trash position can completely modify the trajectory of the broom, even for a fixed starting and collection point. For this task, the trash position can be interpreted as a *task parameter*, governing the variations in different demonstrations. Task-parameterized skill learning aims at adaptive motion encoding to new situations. While existing approaches for task parameterized skill learning have demonstrated good adaptation within the demonstrated region, the extrapolation problem of task parameterized skills has not been investigated enough. In this chapter, with the aim of good adaptation not only within the demonstrated region but also outside of the region, a generative model is combined with a Dynamic Movement Primitive (DMP) by formulating learning as a density estimation problem. Moreover, for efficient learning from relatively few demonstrations, the training data is augmented with additional incomplete data. The proposed method is tested and compared with existing works in simulations and real robot experiments. Experimental results verified its generalization in the extrapolation region.

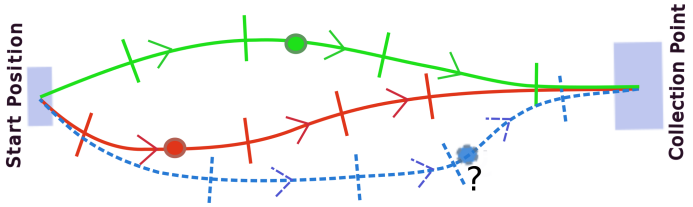


Figure 3.1.: This figure shows the illustration of a sweeping task. The position of the trash (colored circles) can be considered as the task parameter, governing variations in the demonstrations. For a new trash position (blue circle), which is away from the demonstrated region, the robot should be able to generate trajectory for moving trash to the collection point.

3.2. Movement Primitives

3.2.1. Dynamic Movement Primitive

DMP is a way to learn motor actions [99]. It can encode discrete as well as rhythmic movements. I consider the DMP formulation presented in [72], as it overcomes the numerical problems which arises when changing the goal position in the original formulation [99]. A separate DMP is learned for each considered degree of freedom (DOF). A canonical system acts as a clock and for synchronization each DMP is driven by the common clock signal.

$$\tau \dot{s} = -\alpha_s s \tag{3.1}$$

The parameter s is usually initialized to one and it monotonically decays to zero, τ is the temporal scaling factor while α_s determines the duration of the movement. From Equation (3.1), the time t and s are related as $s(t) = \exp(\frac{-\alpha_s t}{\tau})$. The canonical system drives the second order transformed system:

$$\begin{aligned} \tau \dot{v} &= k(g-x) - dv - k(g-x_0)s + sk\mathcal{F}(s) \\ \tau \dot{x} &= v \end{aligned}$$

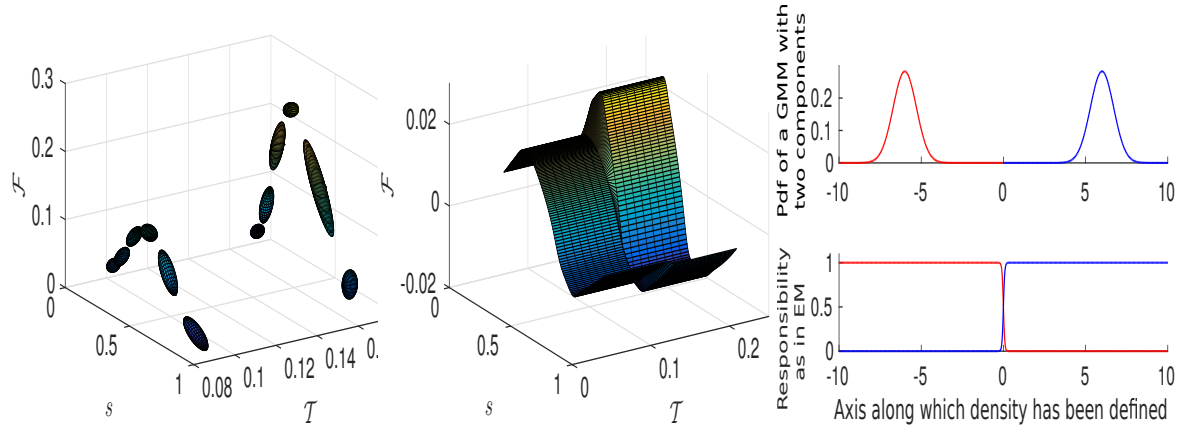
where g and x_0 are goal and start positions respectively, k acts like a spring constant while the damping term d is set such that the system is critically damped. The learning of forcing term $\mathcal{F}(s)$ allows arbitrarily complex movements. $\mathcal{F}(s)$ is defined as $\frac{\sum_{i=1}^{K} \psi_i(s)\omega_i}{\sum_{i=1}^{K} \psi_i(s)}$ where $\psi_i(s) = \exp(-h_i(s-c_i)^2)$ are Gaussian basis functions with spread h_i, centers c_i and adjustable weights ω_i. To encode a movement, $x(t)$ and its first and second derivatives $v(t)$ and $\dot{v}(t)$ are registered respectively at each time step $t = 0, \ldots, T$. Then for a suitable value of τ, the canonical system is integrated to calculate the target value $\mathcal{F}_{tar}(s)$ for each time step.

$$\mathcal{F}_{tar}(s) = \frac{\dot{v}\tau - k(g-x) + dv + k(g-x_0)s}{sk}$$

Now learning is performed to minimize the error criterion $J = \sum_s (\mathcal{F}_{tar}(s) - \mathcal{F}(s))^2$ which is a linear regression problem and the weights ω_i are learned with weighted least squares.

3.2.2. DMP learning with a Gaussian Mixture Model

The forcing term $\mathcal{F}(s)$ can be encoded with a Gaussian Mixture Model (GMM) [6] or any other suitable function approximator [103]. When using a GMM, the manual specification of the meta



(a) (b) (c)

Figure 3.2.: TP-DMP learning using mixture of GMMs. (a) The mixture of GMMs, (b) the underlying regression surface and (c) the intuitive reasoning for such a response.

parameters related to the basis functions (means and spread) is not required as the means and covariances of the GMM components are learned using EM. That is why the GMM based encoding also requires less number of components as compared with the number of basis functions. The number of GMM components can also be optimized by using an appropriate model selection criterion [77]. A GMM with K components is parameterized by $\boldsymbol{\theta}_{(K)} = \{\boldsymbol{\pi}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$, where $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K$ are mixing coefficients with constraints $\boldsymbol{\pi}_k > 0 \quad \text{and} \quad \sum_{k=1}^{K} \boldsymbol{\pi}_k = 1$, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ are means and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$ are covariance matrices. The learning scheme is as follows. First a dataset is created

$$\mathbf{x} = \begin{pmatrix} s_1 & \ldots & s_T \\ \mathcal{F}_{tar}(s_1) & \ldots & \mathcal{F}_{tar}(s_T) \end{pmatrix} \tag{3.2}$$

and then a GMM is fitted to the data with EM [28]. Eigenvalues of covariance matrices are regularized to avoid singularity during EM [16]. Now for retrieving $\mathcal{F}(s)$ for a given s value, Gaussian Mixture Regression (GMR) [10] is utilized. In GMR, input and output variables in each component are represented separately

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^I \\ \boldsymbol{\mu}_k^O \end{bmatrix}, \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^I & \boldsymbol{\Sigma}_k^{IO} \\ \boldsymbol{\Sigma}_k^{OI} & \boldsymbol{\Sigma}_k^O \end{bmatrix}$$

For a given input variable $\mathbf{x}^I$, the expected value of $\mathbf{x}^O$ is calculated as:

$$\begin{aligned} E(\mathbf{x}^O|\mathbf{x}^I) &= \sum_{k=1}^{K} h_k \hat{\mathbf{x}}_k \\ \text{with} \quad h_k &= \frac{\boldsymbol{\pi}_k \mathcal{N}(\mathbf{x}^I; \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I)}{\sum_{l=1}^{K} \boldsymbol{\pi}_l \mathcal{N}(\mathbf{x}^I; \boldsymbol{\mu}_l^I, \boldsymbol{\Sigma}_l^I)} \\ \hat{\mathbf{x}}_k &= \boldsymbol{\mu}_k^O + \boldsymbol{\Sigma}_k^{OI} (\boldsymbol{\Sigma}_k^I)^{-1} (\mathbf{x}^I - \boldsymbol{\mu}_k^I) \end{aligned}$$

3.3. Task Parameterized-DMP

The DMP parameters can be separated into two types: (1) the shape parameters ω_i associated with the basis functions and (2) the DMP meta parameters which are all parameters other than the shape

parameters, i.e. τ, g, K, etc. DMP presented in Section 3.2 does not consider external parameters $\mathcal{T}$, which are referred to as *task parameters* in this work (e.g. the trash position in Figure 3.1). Also the only input to a DMP is the clock signal. In the *Task Parameterized-DMP* (TP-DMP), the goal is to firstly learn from multiple demonstrations executed for different task parameters. Secondly for adapting the motion to a new task, the task parameters should also be passed as an input along with the clock signal. Following are the preprocessing steps that are considered in the presented TP-DMP framework:

1. Since a common clock (canonical system) drive all DMPs, a common time duration is assigned to all demonstrations.

2. All demonstrations are linearly resampled to have an equal number of samples. A sample inbetween two data points is created by linear interpolation of the neighboring data points.

3.3.1. TP-DMP learning using mixture of GMMs

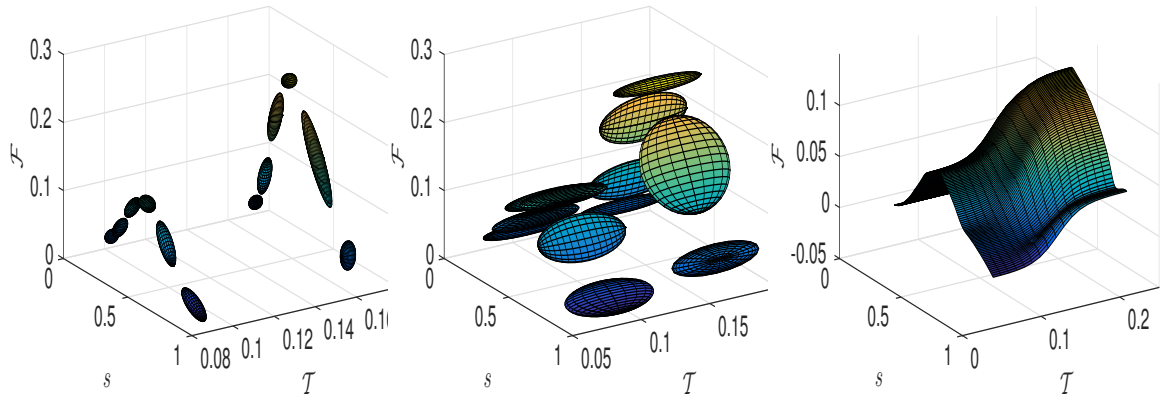
I consider learning as a density estimation problem where the joint distribution of $(s, \mathcal{T}, \mathcal{F})$ is learned. Learning a single GMM over all demonstrations encoding $(s, \mathcal{T}, \mathcal{F})$ can suffer from the curse of dimensionality in higher dimensional space. That is why a separate GMM is learned for each demonstration in a lower dimensional space i.e. $(s, \mathcal{F})$, by first creating the datasets as in Equation (3.2) and then applying EM as mentioned in section 3.2.2.

$$\boldsymbol{\mu}_{o,m} = \begin{pmatrix} \boldsymbol{\mu}^{s}_{o,m} \\ \boldsymbol{\mu}^{\mathcal{F}}_{o,m} \end{pmatrix}, \boldsymbol{\Sigma}_{o,m} = \begin{pmatrix} \boldsymbol{\Sigma}^{ss}_{o,m} & \boldsymbol{\Sigma}^{s\mathcal{F}}_{o,m} \\ \boldsymbol{\Sigma}^{\mathcal{F}s}_{o,m} & \boldsymbol{\Sigma}^{\mathcal{F}\mathcal{F}}_{o,m} \end{pmatrix}$$

Here the subscript o and m denote the indexes of demonstrations and components respectively while the terms s and $\mathcal{F}$ (defined in section 3.2.1) in superscript denote the dimensions corresponding to s and $\mathcal{F}$ respectively. Since the task parameters remain constant during a demonstration, their values can be simply concatenated in the learned means of the GMM components. The diagonal values corresponding to the task parameters in the covariances are not learned and are set to a small value ε:

$$\boldsymbol{\mu}_{o,m} = \begin{pmatrix} \boldsymbol{\mu}^{s}_{o,m} \\ \mathcal{T}_{o} \\ \boldsymbol{\mu}^{\mathcal{F}}_{o,m} \end{pmatrix}, \boldsymbol{\Sigma}_{o,m} = \begin{pmatrix} \boldsymbol{\Sigma}^{ss}_{o,m} & 0 & \dots & 0 & \boldsymbol{\Sigma}^{s\mathcal{F}}_{o,m} \\ 0 & \varepsilon & \ddots & & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \ddots & \varepsilon & 0 \\ \boldsymbol{\Sigma}^{\mathcal{F}s}_{o,m} & 0 & \dots & 0 & \boldsymbol{\Sigma}^{\mathcal{F}\mathcal{F}}_{o,m} \end{pmatrix}.$$

If the task parameters are varying during the demonstrations, the GMMs can simply be learned over all of the variables, i.e. $(s, \mathcal{T}, \mathcal{F})$, and there will be unconstrained covariances. In fact it is the fixed task parameters for each demonstration that makes the learning challenging. The idea of combining separately learned HMMs has also been introduced in [41] but in the approach I present now, an additional EM cycle over the separately learned GMMs is applied for achieving task specific generalization. The separately learned GMMs are mixed to achieve generalization for novel task parameter values. Similar to the mixing coefficients $\boldsymbol{\pi}$ which represent the weights of the components within a GMM, a mixing coefficient ϕ is introduced, representing the mixing weight of each GMM, having the same constraints as that of π, i.e. $\pi_i > 0$ and $\sum_{i=1}^{K} \pi_i = 1$. With these settings EM is applied to learn a mixture of GMMs. The mixing weights and the means of the components are not updated within each GMM, as they are important for preserving the local



(a) (b) (c)

Figure 3.3.: Result of manually setting the variance along task variable. (a) The mixture of GMMs, (b) the mixture of GMMs after manually setting the variance along task variable and (c) the underlying regression surface.

behavior of each demonstration. As EM maximizes the likelihood locally, it can converge to a local maxima. To overcome local maxima problem Deterministic Annealing EM (DAEM) [105] is utilized. DAEM uses the ideas from statistical mechanics, by applying the concept of maximum entropy. The objective function is considered as the thermodynamic free energy, which is regulated by a temperature parameter β. The temperature has a high value in the beginning, which is represented by a small β value. The temperature is incrementally decreased by gradually raising the β to one. EM is applied deterministically for each temperature value and the estimated parameters become initialization for the next temperature value. At higher temperature values DAEM suppresses poor local optima and increases the likelihood of convergence to the global maximum. For M demonstrations with T data points in each demonstration, a single dataset is created, containing all demonstrations (define $N = TM$). Then the DAEM is applied for learning mixture of GMMs, whose update equations can be written as:

E-step:

$$
\begin{aligned}
p_{i,o,m} &= \left(\phi_o^t \boldsymbol{\pi}_{o,m} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{o,m}^t, \boldsymbol{\Sigma}_{o,m}^t)\right)^{\beta} \\
b_{i,o} &= \frac{\sum_{l=1}^{K} p_{i,o,l}}{\sum_{r=1}^{M}\sum_{l=1}^{K} p_{i,r,l}}, \quad q_{i,o,m} = \frac{p_{i,o,m}}{\sum_{r=1}^{M}\sum_{l=1}^{K} p_{i,r,l}}
\end{aligned}
$$

M-step:

$$
\begin{aligned}
\phi_o^{t+1} &= \frac{\sum_{i=1}^{N} b_{i,o}}{N}, \\
\boldsymbol{\Sigma}_{o,m}^{t+1} &= \frac{\sum_{i=1}^{N} q_{i,o,m}(\mathbf{x}_i - \boldsymbol{\mu}_{o,m})(\mathbf{x}_i - \boldsymbol{\mu}_{o,m})^{\top}}{\sum_{i=1}^{N} q_{i,o,m}}
\end{aligned}
$$

where $p_{i,o,m}$ is calculated for utilizing in the calculations of $b_{i,o}$ and $q_{i,o,m}$. $b_{i,o}$ represents the responsibility that the o^{th} GMM takes for explaining the i^{th} data point while $q_{i,o,m}$ represents the

responsibility that the m^{th} component of the o^{th} GMM takes for explaining the i^{th} data point. ϕ_o represent the mixing weight of the o^{th} GMM. It is a common practice to apply regularization on learned covariances for avoiding singularities during EM. As a regularization measure, if any of the eigenvalues of the learned covariances becomes lower than a predefined threshold ε, it is reset to ε. The described approach is applied to the mixture of GMMs learned for encoding the forcing terms of a DMP, with variations along a scalar task parameter. The learned mixture of GMMs, which is also the estimated density function, is shown in Figure 3.2a. The GMMs remain unchanged. The regression for the mixture of GMMs is calculated as:

$$
\begin{aligned}
E(\mathbf{x}^O|\mathbf{x}^I) \quad &= \quad \sum_{r=1}^{M}\sum_{l=1}^{K} v_{r,l}\hat{\mathbf{x}}_{r,l} \\
\text{with} \quad v_{o,m} \quad &= \frac{\phi_o \boldsymbol{\pi}_{o,m}\mathcal{N}(\mathbf{x}^I;\boldsymbol{\mu}^I_{o,m},\boldsymbol{\Sigma}^I_{o,m})}{\sum_{r=1}^{M}\sum_{l=1}^{K}\phi_r\boldsymbol{\pi}_{r,l}\mathcal{N}(\mathbf{x}^I;\boldsymbol{\mu}^I_{r,l},\boldsymbol{\Sigma}^I_{r,l})} \\
\hat{\mathbf{x}}_{o,m} \quad &= \boldsymbol{\mu}^O_{o,m} + \boldsymbol{\Sigma}^{OI}_{o,m}(\boldsymbol{\Sigma}^I_{o,m})^{-1}(\mathbf{x}^I - \boldsymbol{\mu}^I_{o,m}).
\end{aligned}
$$

For evaluation, linearly spaced samples of $\mathcal{T}$ and s are generated within the demonstrated ranges. Then GMR is used to predict the value of $\mathcal{F}$ for each sample, i.e. $\{s \times \mathcal{T}\} \mapsto \mathcal{F}$. The surface plot of this data can be visualized in Figure 3.2b. It can be seen that it has a step along the task parameter. This shows that as there is sparse data in the task space (only two trajectories), each GMM in the mixture kept concentrated at regions of demonstrations. Due to data sparsity, the density estimate is overfitted in its current form. The reason for the step in surface plot can be explained by Figure 3.2c. If there are well separated Gaussians (Figure 3.2c-top) then their activation functions, calculated as the responsibility term in EM, switch in a very narrow region (Figure 3.2c-bottom). The same phenomenon occurred in the regression surface where regions close to a GMM are mostly influenced by it. As one moves away from a GMM, the activation transits sharply to a nearby GMM. This also means that this model is overfitted and can only be useful for exact reproduction of task parameter values in training dataset and it fails to generalize for novel task parameter values. Instead of using EM, a simple trick to avoid overfitting problem could have been to manually specify a reasonable value of variance ε for the task variables, as in the LWR based approaches [104]. This is analogous of applying a regularization term along task variables. Although this can provide interpolation behavior, it fails to extrapolate beyond the demonstrated regions, as the underlying regression surface changes its behavior beyond the demonstrated interval as shown in Figure 3.3.

3.4. Generalizing from incomplete data via the EM

Due to the few demonstrations, the challenge of data sparsity is posed in task space. Now it is shown that the data sparsity problem can be solved by augmenting the training data with the additional incomplete data spanning the input space. This can subsequently be used for getting a better estimate of the underlying distribution and thus improving the generalization behavior of the already learned GMMs. Since the mixture of GMMs is a generative model, it can benefit from incomplete data [12].

3.4.1. Defining input data distribution

For a task parameterized DMP, the input variables are $(s,\mathcal{T})$ for which the prediction of output value $\mathcal{F}$ is desired. Without loss of generality, it can be assumed that the input variables are inde-

pendent of each other but conditionally dependent for a given output value, as shown in Figure 3.4. Now the distribution of each input variable is separately modeled at first. Among the input variables, the clock signal s is generated by an exponentially decaying function (canonical system) and has uneven distribution of samples in different regions. Since a GMM can model any arbitrarily complex density function, the distribution of s is modeled by fitting a univariate GMM with W components (through EM) to its samples $\boldsymbol{\theta}^s_{(W)} = \{\pi^s_w, \mu^s_w, (\sigma^s_w)^2\}^W_{w=1}$. The same procedure cannot be used for the very limited samples of task parameters $\mathcal{T}$, which are equal to the number of demonstrations. For simplicity, each dimension of task variables is assumed to follow a univariate normal distribution, i.e. for the d^{th} dimension of $\mathcal{T} = [\mathcal{T}^1 \dots \mathcal{T}^d \dots \mathcal{T}^D]^\top$, $\mathcal{T}^d \sim \mathcal{N}(\mu_d, \sigma^2_d)$ where $\mu_d = \frac{\mathcal{T}^d_1 + \mathcal{T}^d_2 + \dots + \mathcal{T}^d_M}{M}$ and $\sigma^2_d = \frac{1}{M-1} \sum_{i=1}^{M} (\mathcal{T}^d_i - \mu_d)^2$.

Since the inputs are provided in a regression problem, they are referred as the observable variables. Similarly, the output variables that need to be predicted are termed as missing variables. Using the assumption of independence, the resultant distribution of the input variables is defined by concatenating all the distribution learned separately to form a multivariate GMM with W components. $\boldsymbol{\theta}^{obs}_{(W)} = \{\pi^{obs}_w, \boldsymbol{\mu}^{obs}_w, \boldsymbol{\Sigma}^{obs}_w\}^W_{w=1}$ with $\pi^{obs}_w = \pi^s_w$, $\boldsymbol{\mu}^{obs}_w = \begin{bmatrix} \mu^s_w & \mu_1 & \dots & \mu_D \end{bmatrix}^\top$ and

$$\boldsymbol{\Sigma}^{obs}_w = \begin{bmatrix} (\sigma^s_w)^2 & 0 & \dots & 0 \\ 0 & \sigma^2_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma^2_D \end{bmatrix}.$$

As the output dimension is not considered for now, the input data distribution is termed as Incomplete Data GMM (IDGMM). Figure 3.5 shows a dataset with a mixture of complete and incomplete data. The incomplete data still provides useful information when applying EM for fitting a GMM [28]. As mentioned earlier, the regression using mixture of GMMs encountered problem due to the data sparsity in task space. What would be the effect of filling regions inbetween GMMs with incomplete data, i.e. without the outputs $\mathcal{F}$? The EM applied with additional incomplete samples provide smooth activation of responsibilities when switching from one GMM to a nearby GMM. Since the learning is treated as a density estimation problem, due to the curse of dimensionality the amount of incomplete data required to fill the empty regions can increase drastically with the increase in the dimensions of task parameters. The computational burden of EM also increases with the increase in size of training data. To avoid these problems, the IDGMM is directly used instead.

To use the IDGMM, a weighting parameter analogous to the number of data points represented by the IDGMM has to be specified by the user. The training dataset consists of N data points. The

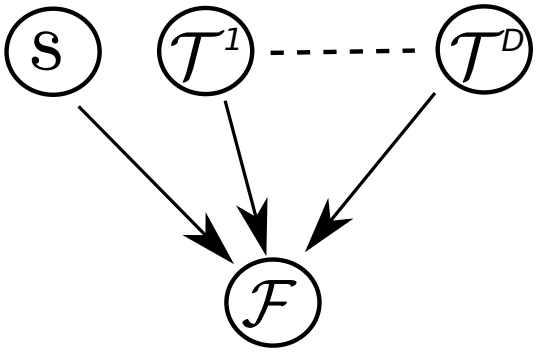


Figure 3.4.: Graphical model illustrating dependence of input and output variables.

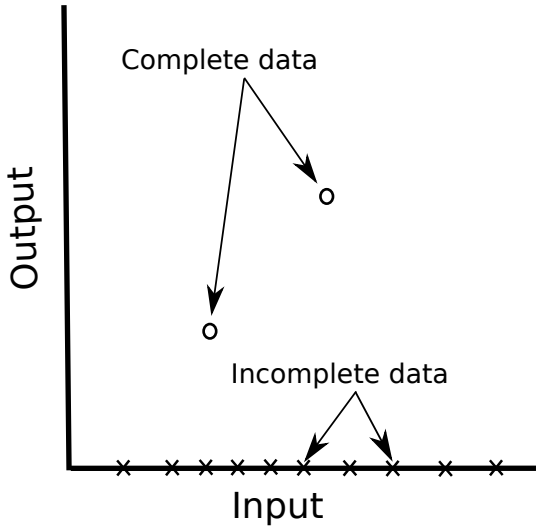


Figure 3.5.: If there are data points whose certain dimensions are missing then they can still be used when applying EM for fitting a GMM. For instance, the output value of the data plotted on the input axis is missing. For this incomplete data, the expected values of the missing output values are also estimated within EM.

weighting parameter is also set equal to N and thus each IDGMM component represent $\boldsymbol{\pi}_w^{obs} \times N$ data points. It has to be noted that the EM applied with this additional data still provides a maximum likelihood estimate of the model parameters as the IDGMM is defined on data and not on the parameters of the model. Now, for benefiting from this incomplete data, the current mixture of GMMs is used for calculating the Expectation of incomplete terms appearing in likelihood maximization [36]. It turns out that, for a data point $\mathbf{x}_i$ with observable (input dimensions) and missing (output dimensions) parts $\begin{bmatrix} \mathbf{x}_i^{obs} \\ \mathbf{x}_i^{miss} \end{bmatrix}$, one has to calculate three expectations [36], i.e. $E[z_{i,k}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$, $E[z_{i,k}, \mathbf{x}^{miss}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$ and $E[z_{i,k}, \mathbf{x}^{miss}\mathbf{x}^{miss^\top}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$, where $z_{i,k}$ is an indicator variable defining association of $\mathbf{x}_i$ to the k_{th} cluster. These expectations can be directly used in M-step [36].

In the M-step, the value of $d_{i,o,m}^{t+1}$ calculated only on the observable dimensions can be directly used for updating ϕ_o^{t+1}.

E-step:

$$
\begin{aligned}
p_{i,o,m}^{t+1} &= \left(\phi_o^t \boldsymbol{\pi}_{o,m} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{o,m}^t, \boldsymbol{\Sigma}_{o,m}^t)\right)^\beta \\
c_{w,o,m}^{t+1} &= \left(\phi_o^t \boldsymbol{\pi}_{o,m} \mathcal{N}(\boldsymbol{\mu}_w^{obs}; \boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_w^{obs} + \boldsymbol{\Sigma}_{o,m}^{obs})\right)^\beta \\
b_{i,o}^{t+1} &= \frac{\sum_{l=1}^{K} p_{i,o,l}^{t+1}}{\sum_{r=1}^{M}\sum_{l=1}^{K} p_{i,r,l}^{t+1}}, \quad q_{i,o,m}^{t+1} = \frac{p_{i,o,m}^{t+1}}{\sum_{r=1}^{M}\sum_{l=1}^{K} p_{i,r,l}^{t+1}} \\
d_{w,o,m}^{t+1} &= \frac{c_{w,o,m}^{t+1}}{\sum_{r=1}^{M}\sum_{l=1}^{K} c_{w,r,l}^{t+1}} \times \boldsymbol{\pi}_w^{obs} \times N
\end{aligned}
$$

M-step:

$$\begin{aligned}
\phi_o^{t+1} &= \frac{\sum_{i=1}^{N} b_{i,o}^{t+1} + \sum_{w=1}^{W}\sum_{l=1}^{K} d_{w,o,l}^{t+1}}{2N}, \\
\boldsymbol{\Sigma}_{o,m}^{t+1} &= \frac{\sum_{i=1}^{N} q_{i,o,m}^{t+1}(\mathbf{x}_i - \boldsymbol{\mu}_{o,m})(\mathbf{x}_i - \boldsymbol{\mu}_{o,m})^\top + \sum_{w=1}^{W} \boldsymbol{A}_{w,o,m}}{\sum_{i=1}^{N} q_{i,o,m}^{t+1} + \sum_{w=1}^{W} d_{w,o,m}^{t+1}}
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{\mu}_{o,m} &= \begin{bmatrix} \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}, \boldsymbol{\Sigma}_{o,m} = \begin{bmatrix} \boldsymbol{\Sigma}_{o,m}^{obs} & \boldsymbol{\Sigma}_{o,m}^{obs.miss} \\ \boldsymbol{\Sigma}_{o,m}^{miss.obs} & \boldsymbol{\Sigma}_{o,m}^{miss} \end{bmatrix} \\
\boldsymbol{A}_{w,o,m} &= d_{w,o,m}^{t+1} \begin{bmatrix} \boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}^\top \\
&= \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix}
\end{aligned}$$

with

$$\begin{aligned}
\boldsymbol{A}_{11} &= d_{w,o,m}^{t+1} \left(\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \right) \\
\boldsymbol{A}_{21} &= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \mathcal{A}_{w,o,m} \\
\boldsymbol{A}_{12} &= {A_{21}}^\top \\
\boldsymbol{A}_{22} &= d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss} + d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} \\
&\quad (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} \mathcal{A}_{w,o,m} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} {\boldsymbol{\Sigma}_{o,m}^{miss.obs}}^\top \\
&\quad - \; d_{w,o,m}^{t+1} \boldsymbol{\Sigma}_{o,m}^{miss.obs} (\boldsymbol{\Sigma}_{o,m}^{obs})^{-1} (\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top
\end{aligned}$$

where $\mathcal{A}_{w,o,m} = \boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top$. The term $c_{w,o,m}$ is calculated for utilizing in the calcualtion of $d_{w,o,m}$ while the term $d_{w,o,m}$ is calculated for utilizing in the calculation of ϕ_o. Since the means are not updated, the additional data cannot pull the already learned GMMs away from the demonstrated regions and the components cannot get to a saddle point during DAEM. Fixing the means also results in a fast rate of convergence during EM.

One may wonder about the result of fitting a single GMM instead of using the mixture of GMMs. As the data is concentrated at discrete regions of input space (task parameters), without the incomplete data, the components of a single GMM will also get attracted to those regions. An appropriate regularization term must be used for a single GMM, to avoid singularity issues of covariance matrices, as the data is concentrated at discrete regions in higher dimensional space. The reason for using the mixture of GMMs instead of a single GMM is that now the weight ϕ of each GMM is also optimized separately, without disturbing the weights π within each GMM, as they are important for preserving the local behavior of each GMM. This also results in a smaller number of parameters update during the second EM cycle, in contrast to updating the mixing weights π of a single GMM.

3.4.2. Computational complexity

The computational complexity of the presented approach during motion execution is $\mathcal{O}(n)$ for the number of DMPs, the number of demonstrations and the number of GMM components. Since

GMR involves matrix inversion over input variables, the computational complexity is $\mathcal{O}(n^3)$ for the dimensions of task parameters. For the special case of fixed task parameters throughout the trajectory, one can find conditional GMMs for the fixed task parameters. This makes the computational complexity irrelevant of task parameters, i.e. for the fixed task parameters $\mathcal{T}$, the conditional parameters for the regression are calculated as:

$$\begin{aligned}
\hat{\phi}_o \hat{\pi}_{o,m} &= \frac{\phi_o \pi_{o,m} \mathcal{N}(\mathcal{T}; \mu^{\mathcal{T}}_{o,m}, \Sigma^{\mathcal{T}}_{o,m})}{\sum_{r=1}^{M} \sum_{l=1}^{K} \phi_r \pi_{r,l} \mathcal{N}(\mathcal{T}; \mu^{\mathcal{T}}_{r,l}, \Sigma^{\mathcal{T}}_{r,l})} \\
\hat{\mu}_{o,m} &= \mu^{\{s,\mathcal{F}\}}_{o,m} + \Sigma^{\{s,\mathcal{F}\}.\mathcal{T}}_{o,m} (\Sigma^{\mathcal{T}}_{o,m})^{-1} (\mathcal{T} - \mu^{\mathcal{T}}_{o,m}) \\
\hat{\Sigma}_{o,m} &= \Sigma^{\{s,\mathcal{F}\}}_{o,m} - \Sigma^{\{s,\mathcal{F}\}.\mathcal{T}}_{o,m} (\Sigma^{\mathcal{T}}_{o,m})^{-1} \Sigma^{\mathcal{T}.\{s,\mathcal{F}\}}_{o,m}
\end{aligned}$$

where the terms $s, \mathcal{F}$ and $\mathcal{T}$ in superscript denote the dimensions corresponding to s, $\mathcal{F}$ and $\mathcal{T}$ respectively.

Table 3.1.: Computational complexity during motion generation with respect to the variables of interest.

The number of DMPs	$\mathcal{O}(n)$
The number of demonstrations	$\mathcal{O}(n)$
The number of GMM components	$\mathcal{O}(n)$
The number of task parameters	$\mathcal{O}(n^3)$
For constant task parameters	$\mathcal{O}(1)$

3.5. Comparison with State-of-the-Art

For all experiments the temperature schedule (β) which is used for annealing (DAEM) is $[0.1\ 0.2\ \ldots\ 1.0]$. The regularization term ε for eigenvalues of the GMM covariances is set to 10^{-6}. For initializing GMMs the trajectories are equally segmented in time domain and then the Gaussians are calculated from the samples of each segment. The parameters of all models are empirically set.

3.5.1. Simulation of variable height obstacle avoidance

The first experiment consists of a planar point to point reaching task with the variable height of the obstacles. If there is an obstacle in the way, the trajectory has to change according to the height of the obstacle. In this experiment the task parameter is defined as *the maximum height to avoid the obstacle*.[1] The goal of learning is to adapt a motion trajectory *for a new desirable height*. The demonstrations can be visualized in Figure 3.6a. There are only two demonstrations with 200 samples in each demonstration. The task parameters associated with these demonstrations are $[0.0903,\ 0.1598]$. Two DMPs are learned for generating motion in the x and y axis. The number of components in each GMM is set to 6. The two GMMs $(\varepsilon = 10^{-6})$ corresponding to the forcing terms of DMPs for motion on the y-axis are shown in Figure 3.6b. Now the IDGMM is applied as described in the prescribed approach, with the components in the IDGMM empirically set to 15. This transforms the complete data GMMs as shown in Figure 3.6c. A single computer with Ubuntu 14.04, Intel Core i7 4790K QuadCore 4.0GHz and 16GiB memory took approximately 13s for fitting the the GMM in Matlab R2015b. The forcing terms of all DMPs can be predicted in less than 1ms during the reproduction in simulation.

[1] This chosen task parameter can be directly used for evaluation for the task performance. But the alternative choice of task parameter is also possible (such as the height of the obstacle) without any algorithmic changes.

Table 3.2.: Mean absolute Error (ME) and its Standard Deviation (SD) for simulations.

	GMR^{1S}	**GMR**^{1S} (without DAEM)	**LWR**^{2S}	**GPR**^{1S}	**GPR**^{2S}	**TP-GMM**
			Interpolation			
ME (m)	0.0027	0.0067	0.0072	0.0015	0.0074	0.0055
SD (m)	0.0012	0.0046	0.0045	0.0010	0.0052	0.0028
			Extrapolation			
ME (m)	0.0113	0.0539	0.0541	0.0407	0.0835	0.0159
SD (m)	0.0033	0.0272	0.0313	0.0265	0.0385	0.0034

Table 3.3.: Task errors (simulation) of different approaches when the height is changed during the trajectories.

	GMR^{1S}	**LWR**^{2S}	**GPR**^{1S}	**GPR**^{2S}	**TP-GMM**
ME (m)	0.0038	0.0329	0.0239	0.0532	0.0129
SD (m)	0.0029	0.0253	0.0221	0.0342	0.0035

Comparison: The proposed approach is compared with **LWR**^{2S} [104], **GPR**^{1S} [103], **GPR**^{2S} [33] and **TP-GMM** [16]. Since Gaussian Mixture Regression has been used for direct prediction of forcing terms, I refer to my approach as **GMR**^{1S}. For **LWR**^{2S}, [104] and [103] used tricubic kernel and Gaussian kernel respectively. I use Gaussian kernel with 5 equally spaced kernels placed along the task space. The choice of kernel is seldom important for the performance of LWR [104]. In **LWR**^{2S} and **GPR**^{2S}, the number of basis functions for shape parameters are set to 10, as the smaller values do not correctly capture the demonstrations. On the other hand, the GMMs in the prescribed approach require fewer number of Gaussians (i.e. 6) in this experiment. As in [103], GPR is used with the covariance function of the Matérn form, with isotropic distance measure and hyperparameters optimization [85]. Three frames of reference are defined for **TP-GMM**, one at the starting point, one above the starting point with height equal to the desired height and one at the ending point. The number of components in the **TP-GMM** is empirically set to 4, as the higher values yield spiky motions. When performing learning with few demonstrations, if the GMMs in TP-GMM are learned with the high number of Gaussains then they converge to the data-points of the individual trajectories. Calculating the product of GMMs with Gaussians concentrated at regions of individual trajectories results in sudden activation of the responsibility term from one Gaussian to another Gaussain of a different trajectory and thus results the spiky behavior and wrong motion reproduction. Also the GMMs in TP-GMM is formed by encoding the relationship inbetween the clock signal and the spatial data with GMMs in different frame of references and their products afterwards while the GMMs in a TP-DMP model encode the relationship inbetween the clock signal, task parameters and the forcing term of a DMP. Since both models operate differently and encode different set of variables, setting the same number of Gaussians in each is not needed for fair comparison.

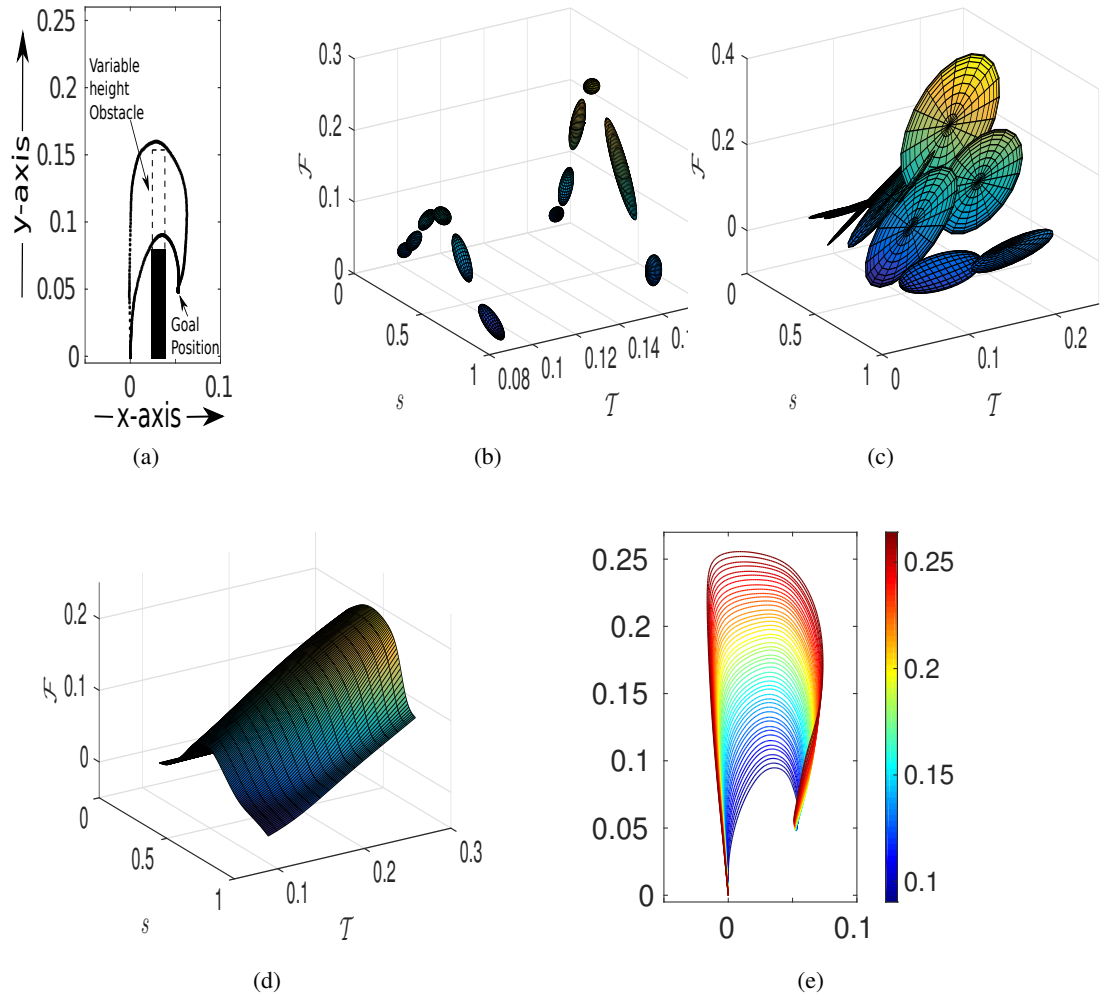


Figure 3.6.: A step by step illustration of TP-DMP learning with additional incomplete data and DAEM. (a) Planar demonstrations, (b) initial GMMs encoding forcing terms of DMPs for y-axis, (c) transformed GMMs using incomplete data and DAEM, (d) learned regression surfaces for y-axis, (e) multiple generated movements. The color encoding in (e) is used to indicate the desired height for each trajectory.

The errors can be defined as the difference inbetween the maximum desired height of the trajectory (i.e. the input task parameter value used for the regression) and the actual achieved height values by the reproduced trajectories. 50 linearly spaced task parameter values are generated in the range $[\min(\mathcal{T}), \min(\mathcal{T}) + 2.5 \times (\max(\mathcal{T}) - \min(\mathcal{T}))]$ ([0.0903,0.2641]). The generated trajectories for these task parameters are shown in Figure 3.6e and 3.8. Table 3.2 presents the Mean absolute Error (ME) and its Standard Deviation (SD) when interpolating (task parameter in the range [0.0903,0.1577]) and extrapolating (task parameter in the range [0.1612,0.2641]). All approaches produce small errors when interpolating while **GMR**^{1S} outperformed all other approaches when extrapolating beyond the demonstrated task parameters. The use of DAEM is critical for the performance of **GMR**^{1S} as the performance degrades substantially without annealing. The **TP-GMM** fails to preserve the shape information of the demonstrations, as it can be seen in Figure 3.8e.

Figure 3.6d and Figure 3.7 present the surface plots of the generated forcing terms $(\{s \times \mathcal{T}\} \mapsto \mathcal{F})$ of the DMP based approaches for y-axis. The regression surfaces of all the approaches ex-

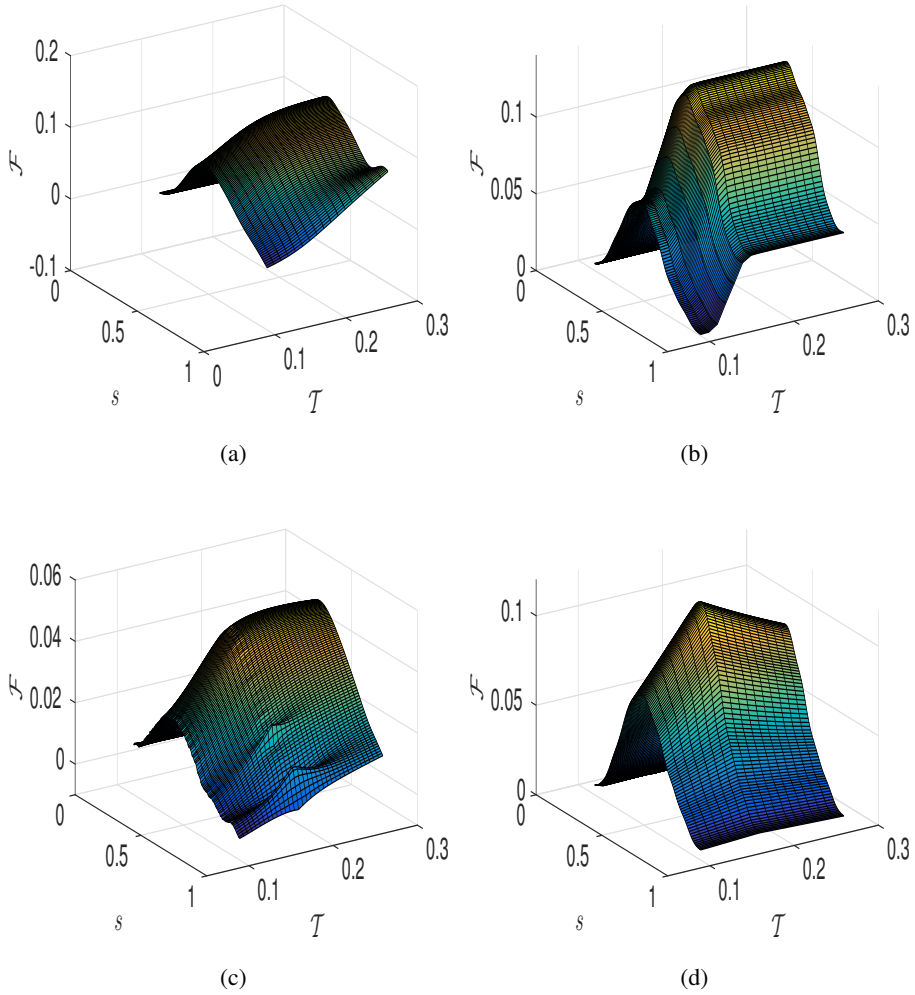


Figure 3.7.: Learned regression surfaces for y-axis (a) with **GMR**^{1S} without DAEM, (b) with **LWR**^{2S}, (c) with **GPR**^{1S} and (d) with **GPR**^{2S}.

cept **GMR**^{1S} change their response in the extrapolation range, which is also the reason for their large errors when extrapolating. The regression surface of **GMR**^{1S} without DAEM shows that the EM converged to a poor local optima without annealing. The kernels in **LWR** and **GP** based approaches predict by mostly using the nearby data. Thus, their performance degrades when trying to extrapolate. The emphasis in **TP-GMM** model is to strictly pass through certain frames of reference and thus it can lose the shape information. Additionally, with the clock signal as the only input, the starting point of the reproductions with the **TP-GMM**, which is marked by a $'-'$ sign in Figure 3.8e, moves quite far away from the starting point of the demonstrations. This problem can happen when the clock signal is the only input in the **TP-GMM** without consideration of the current point of a trajectory. A remedy to such a problem can be to encode the current point as an input.

Since a generative model is used for encoding the forcing terms of the DMP, this approach can benefit from the incomplete data. The IDGMM spans beyond the demonstrated range and thus retrieves a better underlying function when compared with the supervised learning approaches, which only rely on training data.

Varying task parameter: The task parameter is not necessary to be fixed during the repro-

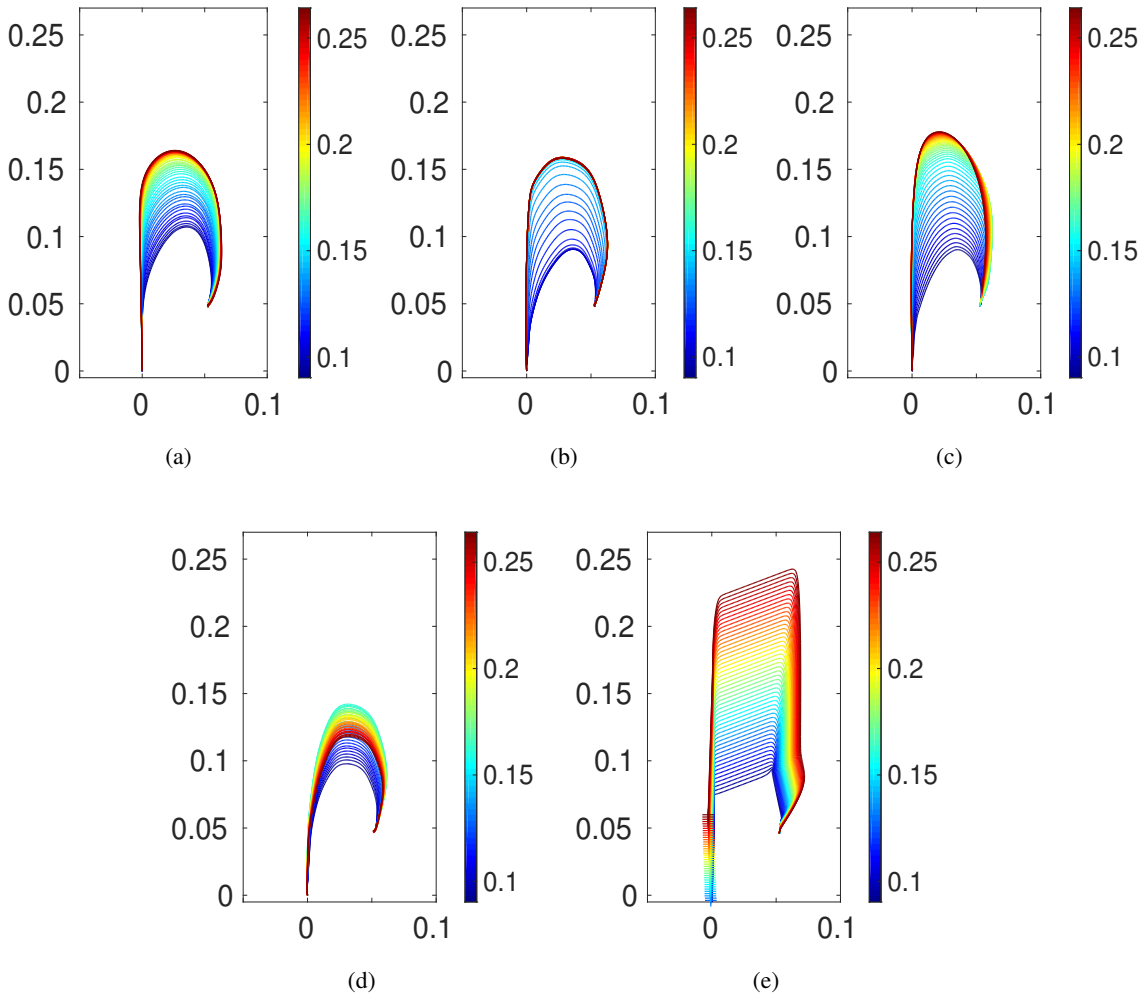


Figure 3.8.: Motion reproductions (a) with **GMR**^{1S} without DAEM, (b) with **LWR**^{2S}, (c) with **GPR**^{1S}, (d) with **GPR**^{2S} and (e) with **TP-GMM**. The color encoding is used to indicate the desired height for each trajectory.

duction phase and can vary if needed. Now 20 trajectories are generated with the initial desired task parameters linearly sampled from the interval $[0.0903, 0.1577]$. After one third of the executed motion, the desired task parameters are multiplied with 1.5. They now lie in the interval [0.1355,0.2366]. Figure 3.9 contains the plot of the generated trajectories. A benefit of using a DMP based approach is that the output trajectories are always smooth, even though the change in the task parameters is discontinuous. The reproduction errors are given in Table 3.3. Again, due to the aforementioned reasons, the **TP-DMP** outperforms all other approaches by producing least amount of error. Care should be taken in a real robot experiment, as an instantaneous change in the value of desired task parameters can cause a high acceleration at the end-effector. The high accelerations can be avoided by smoothly changing the task parameters when required.

3.5.2. Real robot experiments

Experimental setup: The experiments are conducted using a KUKA lightweight robot IV+. For collecting demonstrations, the robot is set to gravity compensation mode. With **GPR**^{1S}, offline

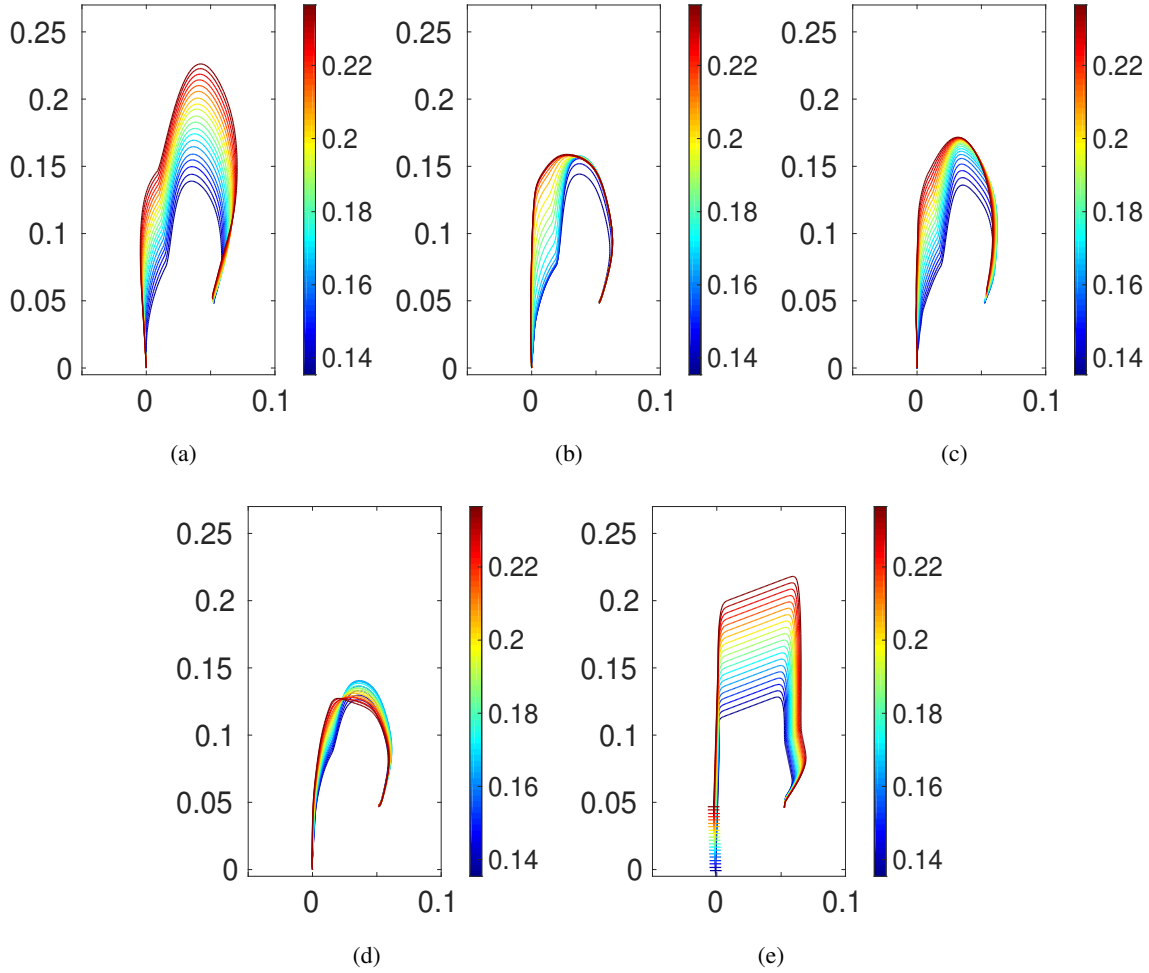


Figure 3.9.: Motion reproductions for changing the height during execution (a) with **GMR**^{1S}, (b) with **LWR**^{2S}, (c) with **GPR**^{1S}, (d) with **GPR**^{2S} and (e) with **TP-GMM**. The color encoding is used to indicate the changed desired height for each trajectory.

trajectories are generated due to its high computational cost. This also means that with **GPR**^{1S}, the task parameters should remain fixed during the motion reproduction. The markers are tracked with Kinect RGB-D camera by using ROS wrapper for Alvar, an open source augmented reality tag tracking library[2]. More specifically there is a fixed marker with respect to robot's frame of reference and a moving marker. The fixed marker is used for localizing the camera while the moving marker is used for tracking objects, as shown in Figure 3.10. A low pass filter is applied to remove high frequency noise in the vision system. The GMM model is always learned offline in Matlab. A single computer with Ubuntu 12.04 32-bit, Intel Core i5-2500 quad core 3.3GHz and 16GiB memory is used for marker tracking as well as online motion generation. For motion reproduction with **GMR**^{1S}, the forcing terms of all DMPs are predicted within 7 ms. A Cartesian impedance controller with a control frequency of 100Hz is used for motion generation.

[2] `http://wiki.ros.org/ar_track_alvar`

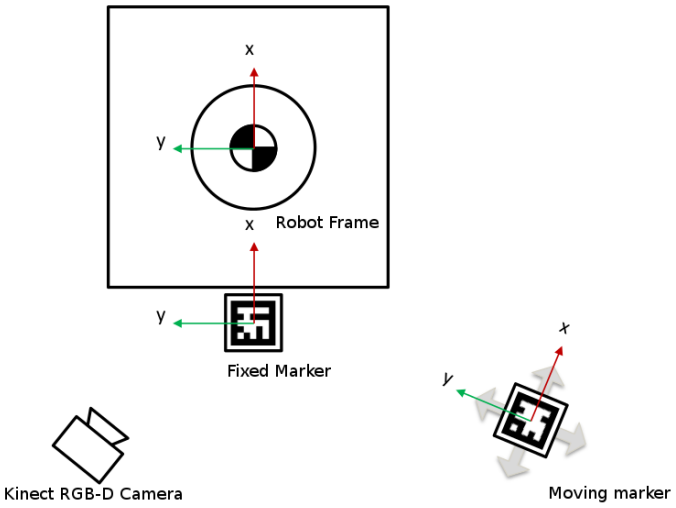


Figure 3.10.: Schematic of the vision system.

Sweeping task:

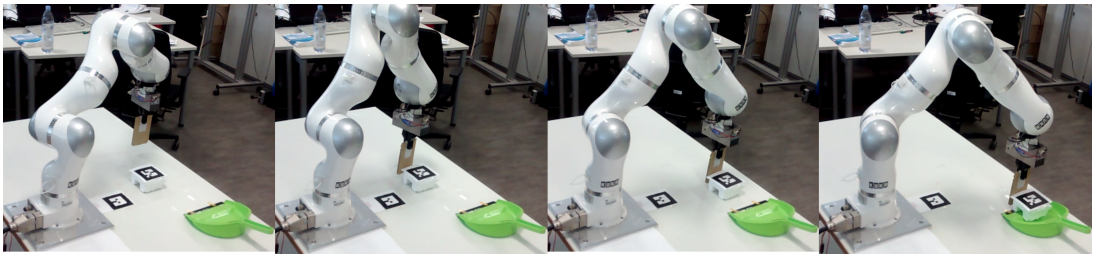
This experiment considers a sweeping task, which consists of moving trash to a collection point. The task parameters are defined as *the planar coordinates of the trash*. A teacher physically holds the end-effector for various trash positions and demonstrates the required motions for moving the trash to the collection point, as shown in Figure 3.11a. Learning is performed in task space (position and orientation of the end-effector). Each demonstration has a duration of approximately 5 seconds with a sampling rate of 10ms.

Table 3.4.: Task errors (KUKA) with different approaches.

	GMR^{1S}	**LWR^{2S}**	**GPR^{1S}**	**GPR^{2S}**	**TP-GMM**
			Interpolation		
ME (m)	5.4×10^{-3}	13×10^{-3}	7.5×10^{-3}	14.5×10^{-3}	11×10^{-3}
SD (m)	4.3×10^{-3}	7.1×10^{-3}	4.3×10^{-3}	10.3×10^{-3}	5.3×10^{-3}
			Extrapolation		
ME (m)	9.9×10^{-3}	68.3×10^{-3}	43×10^{-3}	84.6×10^{-3}	33.8×10^{-3}
SD (m)	9.6×10^{-3}	6.3×10^{-3}	5.7×10^{-3}	13×10^{-3}	8.2×10^{-3}

Three DMPs are learned, two for generating a planar motion and one for encoding the planar orientation of the end-effector. As shown in Figure 3.12a, four demonstrations are provided for different positions of the trash. In the demonstrations, the x and y values of the trash position lies between $[-0.0216m$,$0.0350m]$ and $[-0.54m$, $-0.454m]$ respectively (drawn as a rectangle). 25 new trash positions (as a grid) are selected for evaluation in the extended x and y ranges of $[-0.0358m$,$0.0491m]$ and $[-0.6045m$, $-0.3895m]$ respectively, which can be visualized in Figure 3.12b. The blue samples, which are close to the bounding box of the demonstrated region,

(a)



(b)

Figure 3.11.: Demonstrations are collected by setting the robot in gravity compensation mode while a Cartesian impedance controller is used for motion generation. (a) A human provides the demonstration for moving the trash to the dustpan. (b) The robot generates motion for a new trash position.

are used for evaluating interpolation performance. The red samples, which are far away from the bounding box, are used for evaluating extrapolation performance.

Comparison: The error is defined as the minimum distance inbetween the trash position and the generated trajectory. Table 3.4 contains the ME and its SD when using the presented approach, **LWR**^{2S}, **GPR**^{1S}, **GPR**^{2S} and **TP-GMM**. Three frames at starting point, ending point and at the trash location are defined for **TP-GMM**. The number of components in the **TP-GMM** is empirically set to 6, as the higher values yield spiky motion. The basis functions in **LWR**^{2S} and **GPR**^{2S} are set to 60. For the presented approach the components in each GMM, as well as the IDGMM, are set to 40. The remaining settings are same as in previous experiment. Like in the previous experiment, the presented approach requires less components as compared with the basis functions in the other approaches. Again, the presented approach produces less errors for interpolation as well as extrapolation as shown in Figures (3.12c-3.12f) and Table 3.4. The error for other approaches increases considerably as one moves away from the demonstrated interval. Additionally, some of the trajectories generated by **TP-GMM** surpassed the collection points as the **TP-GMM** model does not have the notion of a goal position. Interestingly, no demonstrations are provided for the trash positions in the upper half of the sweeping area, which makes it an interesting region for comparing different approaches. **GMR**^{1S} also successfully generates the motion for the trash position in this region, as shown in Figure 3.12f.

Striking task:

This experiment considers a task which involves striking a ball such that it hits a desired target position. The ball is always placed at the same point and the task parameter is defined as the *the x-coordinate of the target*. The y-coordinate of the target is fixed in the two demonstrations. Similar to the previous experiment, a teacher physically holds the end-effector and demonstrates

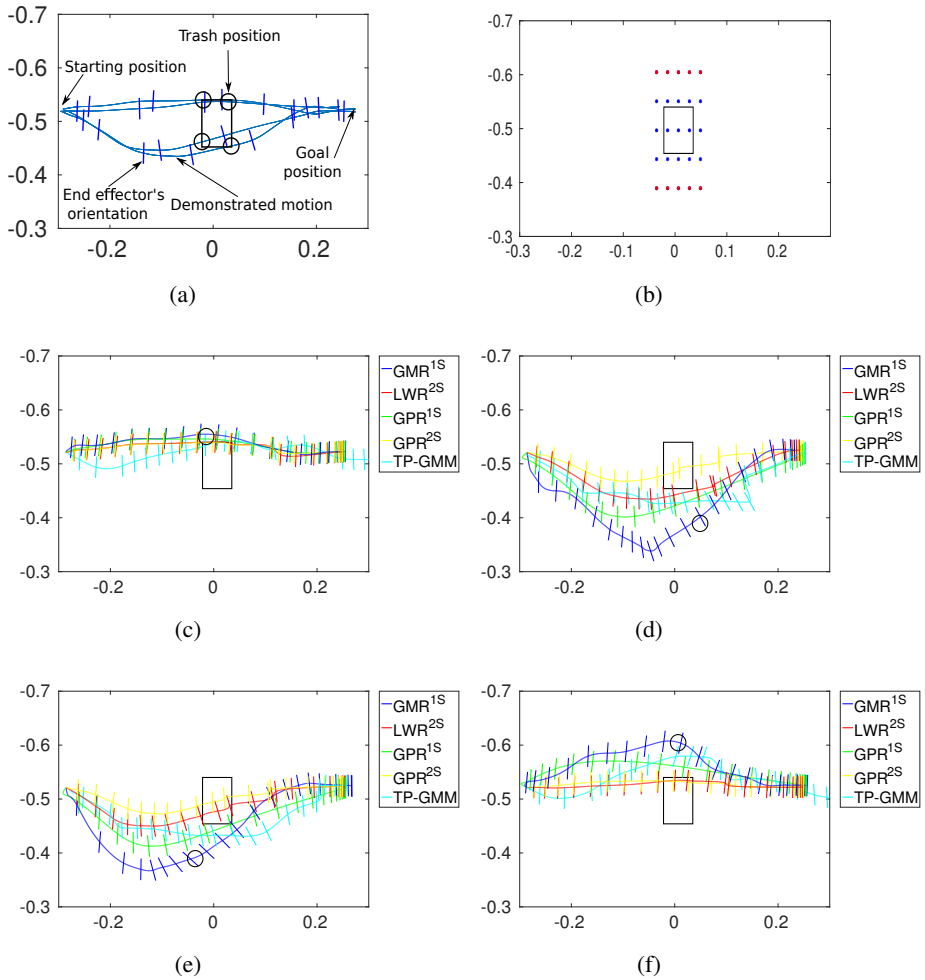
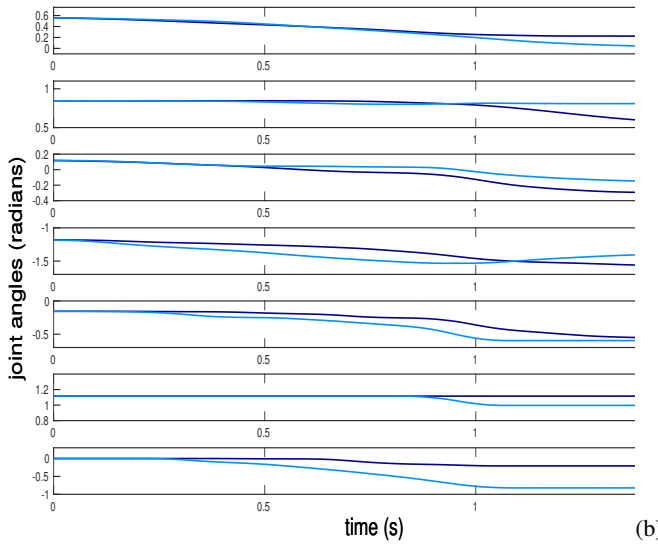


Figure 3.12.: Comparison of **GMR**^{1S}, **LWR**^{2S}, **GPR**^{1S}, **GPR**^{2S} and **TP-GMM** for the sweeping task. (a) Demonstrations for the sweeping task. Circles represent the trash positions while the rectangle represents the bounding box enclosing these trash positions. (b) Blue dots represent trash positions for interpolation evaluation while the red ones are for extrapolation evaluations. (c,d,e,f) Generated movements for new trash positions.

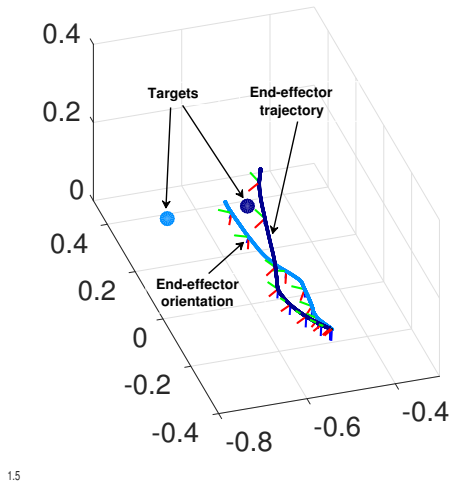
the required motion for hitting the target. The demonstrations have a duration of appropriately 1.5 seconds with a sampling rate of 10ms. Due to the small duration of the motions the size of data is increased ten times (to approximately 1500) by inserting samples in between the adjacent data points of the trajectories by using linear interpolation. Now the learning is considered in joint space. Different demonstrations can produce completely different final joints configuration during the demonstrations. The final joint configurations are measurable in the demonstrations but are unknown when reproducing motion for a new target position. Thus, it is necessary to predict the final joints configuration during the reproduction phase. Seven DMPs are learned with one DMP for each joint of the robot and thus utilizing all DOFs.

The presented approach can also easily incorporate the learning of meta parameters in a DMP. For **GMR**^{1S}, the distribution of $(s, [\mathcal{T}\ g], \mathcal{F})$ is learned. Similar to task parameters, the goal terms g (meta parameters) are constants for a single demonstration. Thus they are simply interpreted as additional task parameters. This also means that different DOF in each demonstration will now have different task parameters. The final joint configurations (goal positions), which is set as an

additional task parameter, cannot be known in advance. As there is no distinction inbetween input and output variables when fitting a GMM and during GMR, any set of variables can be selected as input, to retrieve the expected value of remaining variables. Thus with GMR, the observable task variable can be used for predicting the expected value of missing task variables and for motion generation. So now, with GMR, not only the goal terms g of each DMP is predicted but its forcing term is also generated. For **GMR**^{1S} the components in each GMM ($\varepsilon = 10^{-4}$), as well as in the IDGMM, are set to 60. The number of basis functions in **LWR**^{2S} and **GPR**^{2S} is set to 100. The goal positions for **LWR**^{2S} and **GPR**^{2S} are predicted in the first step along with the DMP parameters by using LWR and GPR respectively. The goal position in **GPR**^{1S} is predicted at each time step along with the forcing terms by using GPR. Two frames of reference are defined for **TP-GMM**: one at the start of the trajectory and one at the end of the trajectory. As mentioned before, the final joint configurations are not known and hence the **TP-GMM** cannot be used directly in this experiment. To use **TP-GMM**, the final joint configuration is first predicted with GP and then the second frame of reference is placed at that final joint configuration for motion reproduction i.e. the offset vector for second frame of reference looks like $b_2 = [t_f \; j_{f1} \; j_{f2} \; j_{f3} \; j_{f4} \; j_{f5} \; j_{f6} \; j_{f7}]'$ where t_f is the final time value and j_{fn} is the predicted final joint angle for the n^{th} joint. The transformation matrices for the two frames of references are set to identity matrices. The number of components in the **TP-GMM** is empirically set to 4. The remaining settings are same as in the previous experiments.



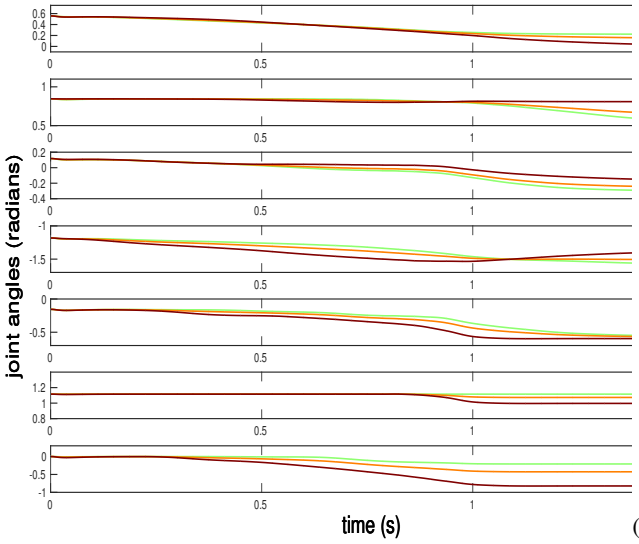
(a) Demonstrated joint angles in radians and time in seconds.



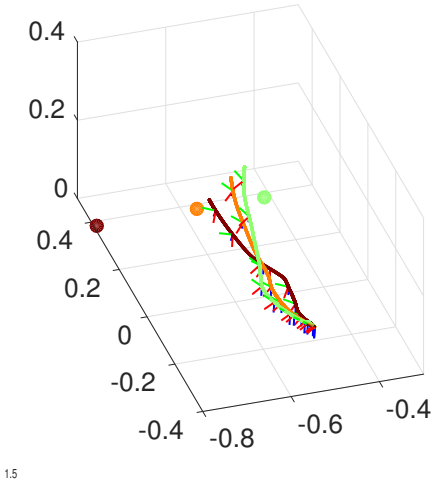
(b) Demonstrated end-effector trajectories in Cartesian space.

Figure 3.13.: Joint angles and end-effector trajectories of the demonstrated motions. Color encoding is used to indicate the correspondence inbetween the motions of joint space and the cartesian space.

Comparison: A binary evaluation criterion is defined as a success if the robot is able to hit the target and failure otherwise. Demonstrations are provided for hitting a target with x-coordinate of $-0.4891m$ and $-0.6703m$, as shown in Figure 3.13a. Figure 3.13b shows the end-effector pose. Interpolation performance is evaluated for the target x-coordinates of $-0.5663m$ while extrapolation performance is evaluated for the target x-coordinates of $-0.4146m$ and $-0.7933m$. When generalizing for novel goal positions, the DMPs can produce high accelerations at the beginning of the movement [47]. This is due to the initial interaction inbetween the linear dynamics and forcing

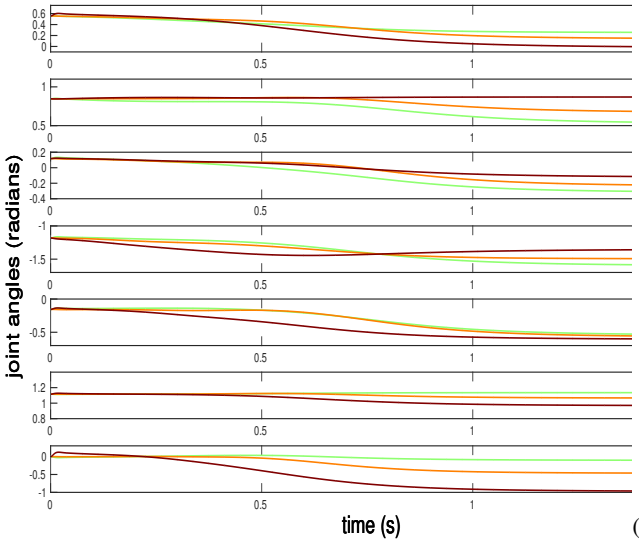


(a) Reproduced joint angles.

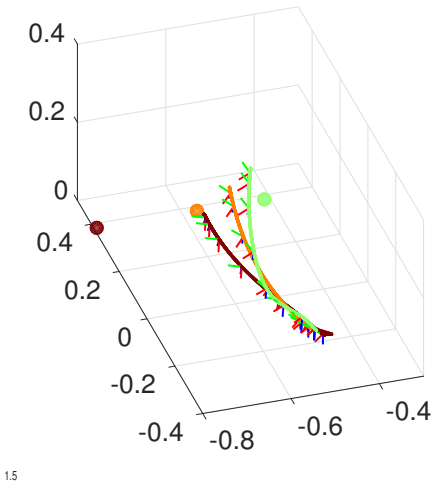


(b) End-effector trajectory in Cartesian space.

Figure 3.14.: Reproductions with **LWR**^{2S}



(a) Reproduced joint angles.

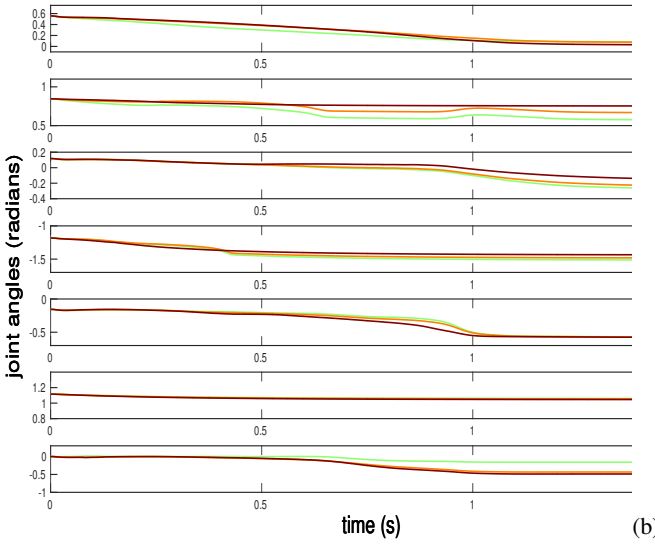


(b) End-effector trajectory in Cartesian space.

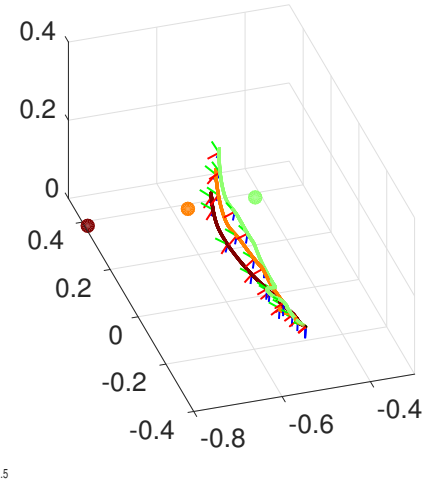
Figure 3.15.: Reproductions with **GPR**^{1S}

terms. This undesirable behavior was slightly observed in this experiment. A simple solution to solve this problem is to gradually activate the forcing terms. Thus the predicted forcing terms are multiplied by $(1-s^{10})$. As s decays exponentially, the effect of this term vanishes very quickly.

Figure 3.14 contains the reproduction results with **LWR**^{2S}. It produces a good trajectory for interpolation performance as it lies inbetween the demonstrated trajectories. The trajectories for the extrapolation fails to reproduce the task as they are similar to the demonstrations. The **LWR**^{2S} also encountered the same problem during the first experiment where successful trajectories were

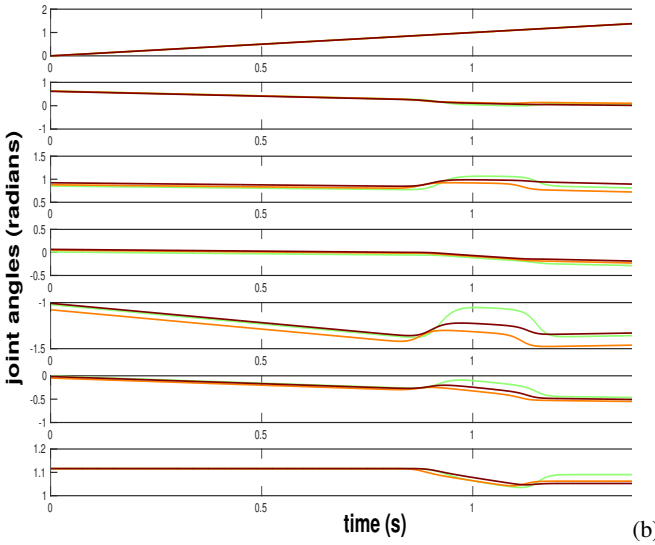


(a) Reproduced joint angles.

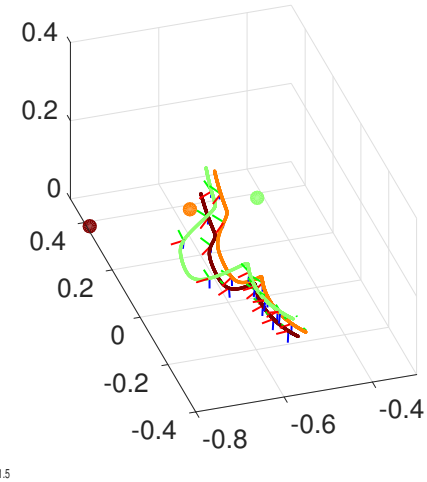


(b) End-effector trajectory in Cartesian space.

Figure 3.16.: Reproductions with **GPR**^{2S}



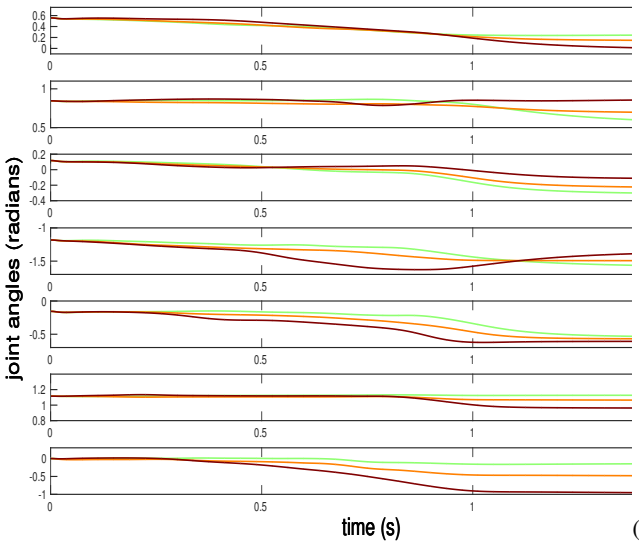
(a) Reproduced joint angles.



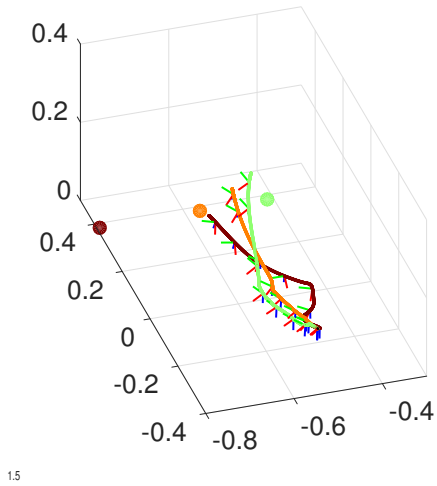
(b) End-effector trajectory in Cartesian space.

Figure 3.17.: Reproductions with **TP-GMM**

generated for interpolation intervals but it failed to reproduce during the extrapolation intervals. Figure 3.15 contains the reproduction results with **GPR**^{1S}. It can easily be observed that the initial end-effector trajectory required to approach the ball (and critical for the execution of the task) is very similar for the three reproductions. The trajectories lose the important shape information required for the successful execution of the task. Also the green and brown trajectories of one of the joints are almost overlapping. Since the reproduction is in joint space, an incorrect motion of even a single joint can lead to the failure of the task. Figure 3.16 contains the reproduction results

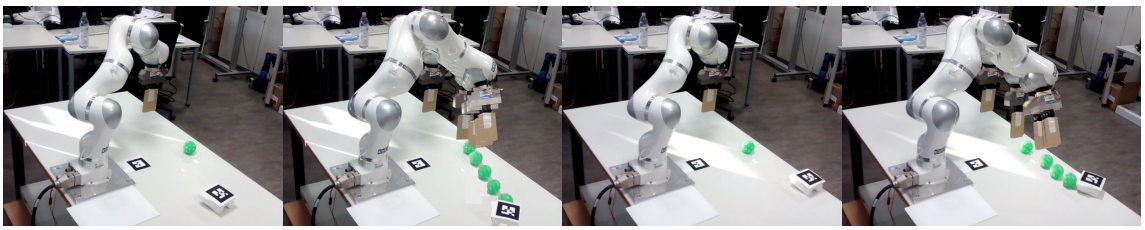


(a) Reproduced joint angles.



(b) End-effector trajectory in Cartesian space.

Figure 3.18.: Reproductions with **GMR**^{1S}



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

Figure 3.19.: Executed motions for the two extrapolation evaluations. (a,c) Robot initial configuration and different target positions. (b) Executed striking motion for hitting the target in (a). (d) Executed striking motion for hitting the target in (c).

with **GPR**^{2S}. Some of the reproduced trajectories are very different from the demonstrated ones. As with **GPR**^{1S}, the trajectories generated with **GPR**^{2S} lose the initial shape information which is required for the successful execution of the task. The failure of both of the GP based approaches can be attributed the small amount of training data, i.e. only two trajectories. Figure 3.17 contains the reproduction results with **TP-GMM**. The reproduced trajectories do not capture the demonstrated motions and it fails to learn anything useful for this task. The joint distribution of all the eight variables (one phase and seven joint angles) is encoded in the **TP-GMM**. As there are only two trajectories, the product of GMMs in **TP-GMM** suffers from a severe curse of dimensionality.

Figure 3.18 contains the reproduction results with **GMR**^{1S}. The shape information is preserved in the reproduced trajectories and the DMPs goal parameters are correctly inferred. The generated joint angles trajectories extend further away from the demonstrated trajectories for extrapolation. The joint angles trajectories for interpolation are inbetween the extrapolation trajectories. Executing motion trajectories generated by **GMR**^{1S} on KUKA show the presented approach always yields success in the extended range of $[-0.4146, -0.7933]$. For the two extrapolation evaluations with the presented approach, the ball trajectories, as well as the executed motions on KUKA, are visualized in Figure 3.19, where the different final joint configurations of the reproduced motions

are also observed.

3.6. Discussion

This chapter presents an approach for learning a TP-DMP by utilizing incomplete data. The existing works extending DMPs to include task parameters have used discriminative approaches for learning [33, 65, 103, 104]. Discriminative models are used for modeling the dependence of an unobserved variable y on an observed variable x. This is done by modeling the conditional probability distribution $P(y|x)$, which is then used for predicting y from x. On the contrary, the presented method combine a generative approach with the DMP, as it can make use of incomplete/unlabeled training data. This results in an 1-Step learning procedure similar to [103]. Generative models encode the joint distribution of $P(x,y)$ of the variables of interest. The conditional distribution can later be inferred from the joint distribution i.e. $P(y|x) = P(x,y)/p(x)$ where $p(x)$ is obtained by marginalizing out y from $P(x,y)$. The aim of this work is to learn from very few demonstrations which implies sparsely distributed data. The data sparsity problem is solved by augmenting training data with additional incomplete data while poor local optima are avoided with Deterministic Annealing Expectation Maximization (DAEM).

In the presented approach, the use of a generative model with a DMP may seem like a strange choice at first, as existing approaches use discriminative models [33, 65, 103, 104]. The discriminative models have been shown to yield lower asymptotic errors compared to their generative counterparts, but they also require higher amount of training data to reach those levels [45]. More specifically, a discriminative model requires $\mathcal{O}(n)$ training examples for reaching its asymptotic error while a generative model require $\mathcal{O}(\log n)$ [45]. This implies that, for few training examples, the generative model might have already reached its lowest asymptotic error, and thus performing better than a discriminative model. Since PbD focuses on learning from as few examples as possible, a generative model is indeed a useful choice.

The experiments demonstrates that the TP-GMM model did not perform well. TP-GMM model relies on learning GMMs in multiple frames of references. For inferring the motion for new settings, these GMMs are placed to new positions and multiplied to retrieve a resultant GMM. The resultant GMM is then used for motion reproduction. The TP-GMM model performs well when there are relatively large amount of demonstrations, as it has to fit a GMM model in each frame of reference for the demonstrations data. In our experiments, there are not enough demonstrations for the TP-GMM model to learn the task properly while in our approach we resolve the data scarcity problem by utilizing the incomplete data. Additionally the TP-GMM model is mostly successfully applied for tasks where the trajectory has to pass through certain locations and hence the idea of observing the demonstrations from multiple frames of references arisen. The TP-GMM did not perform well in the striking task because the generated motion did not have to actually pass through the target location. For the obstacle avoidance and the sweeping tasks it loses the shape information, as its main aim is to pass through the frames and in doing so compromises the shape of the trajectories.

3.7. Summary and Future works

In this chapter it is shown that how the task specific generalization of a DMP can be achieved by formulating learning as a density estimation problem. The proposed approach captures the local behavior of each demonstration by using a GMM. These GMMs are then mixed to get the task specific generalization. The data sparsity along task parameters is handled tby introducing

additional incomplete data filling the input space. Deterministic Annealing EM is used to avoid the local maxima problem. The local behavior of each GMM is retained by keeping the means and mixing weights within the GMMs fixed. The task specific generalization is achieved by just adapting covariances and mixing coefficient of the already learned GMMs. The TP-DMP framework can perform learning in task space as well as in joint space and can even handle the learning of meta parameters of a DMP. As shown in the experiments, the proposed approach requires very few demonstrations for learning and it outperforms the existing approaches specially when extrapolating beyond the demonstrated ranges of the task variables.

There are a couple of directions in which the presented TP-DMP approach can be extended. PbD approaches are often utilized for initializing models for reinforcement learning. Hence improving the performance of a TP-DMP model with reinforcement learning can also be investigated. Similar to reinforcement learning, a reproduced motion with some reproduction errors can be relabelled with the correct task parameters and included as a new demonstration. This idea is called sample reuse. For instance in the obstacle avoidance task, a generated motion for a wrong height can be relabelled with the height for which it would have been a successful execution and included in the training data as a new demonstration. Apart from reinforcement learning, the TP-DMP model can also be extended with the sample reuse approach [27] for improving the model's performance. Additionally, the scalability of the presented approach to more complex tasks, with higher dimensional task parameters should also be tested.

CHAPTER 4

Deep Movement Primitives

4.1. Motivation

Dynamic Movement Primitives (DMPs) are widely used for encoding motion data. As shown in the previous chapter, Task Parameterized DMP (TP-DMP) can adapt a learned skill to different situations. Task parameters are any additional environment variables that can influence the generated motion. Examples include positions of an object or a target in the environment. The main limitation of the task parameterized DMP approach is that the task parameters have to be provided during the motion reproduction. Customized vision systems are usually deployed to extract task specific variables. These dedicated vision systems are usually specifically designed for the targeted problems. For instance, a blob detection algorithm is used for tracking the position of a ball in [48]. In [78] markers are used for extracting positions of objects in the environment. The use of such dedicated systems limits the use of these approaches for real world scenarios.

This chapter proposes a method for combining the DMP with a Convolutional Neural Network (CNN). CNN is a special type of feed forward neural network which can process high dimensional inputs and has been successfully applied to analyse visual data [26, 55, 56, 81, 102]. The CNN's architecture also draws inspiration from animals visual cortex, which responds to the small regions of the visual field [44]. To mimic this property, CNN's utilize weight sharing which reduces the complexity (number of parameters) of the model and at the same time alleviate the overfitting problem commonly associated with the fully connected neural networks. The presented approach preserves the generalization properties associated with a DMP, while the CNN learns the task specific features from the camera images. This eliminates the need to extract the task parameters, by directly utilizing the camera image during the motion reproduction. The performance of the developed approach is demonstrated with a real robot trash cleaning task, which is also presented in the previous chapter. It is also shown that by using the data augmentation, the learned sweeping skill can be generalized for arbitrary real objects. As compared with the work presented in the previous chapter, the use of a CNN eliminates the need to use markers for extracting the positions of the trash. The experiments show the robustness of the presented approach for several different settings.

4.2. Deep-DMP

As stated earlier, the forcing term of a DMP can be modeled with any suitable function approximator. In this chapter, it is modeled with a CNN. By doing so all the useful properties associated with a DMP model are preserved, i.e. temporal and spatial rescaling properties as well as the guaranteed convergence to a goal position, while the generalization capabilities associated with feature learning are exploited by using a CNN. Figure 4.1 presents an overview of the presented approach. Since deep learning is employed here, this formulation is called as Deep-DMP (**D-DMP**). Compared with TP-DMP, the camera images are directly processed by the **D-DMP**, eliminating the need to extract the task parameters during motion reproduction, as shown in Figure 4.2.

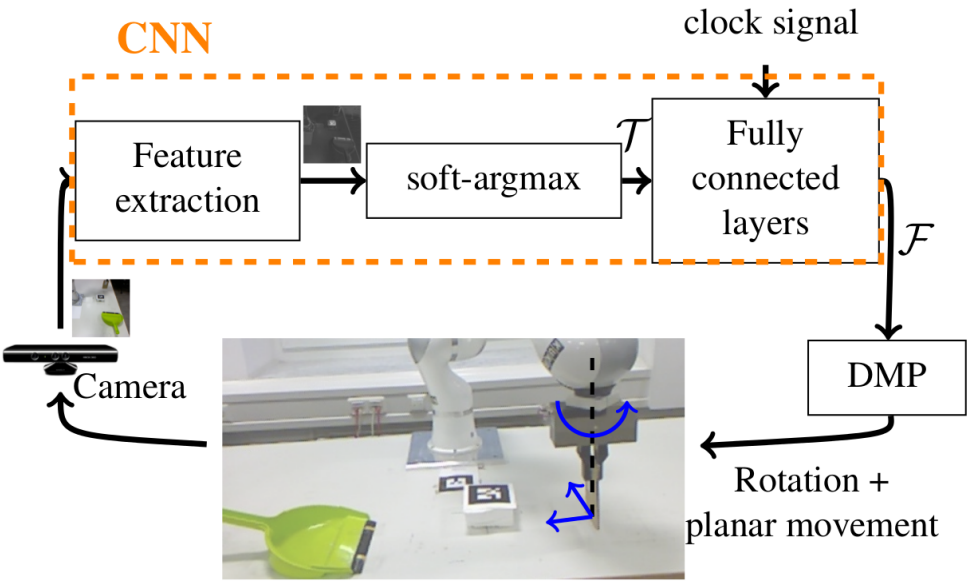


Figure 4.1.: An overview of the proposed Deep-DMP (**D-DMP**) architecture.

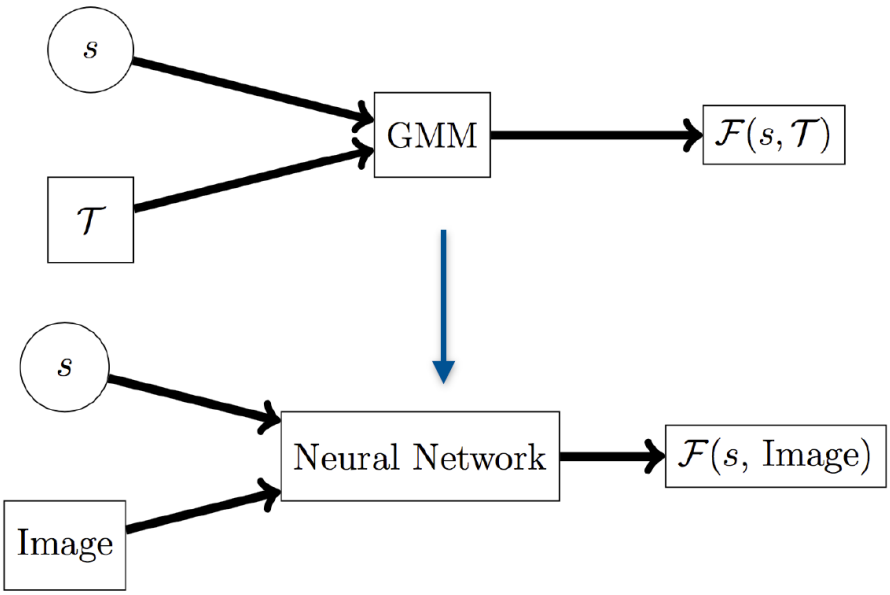


Figure 4.2.: Schematic of TP-DMP (above) and **D-DMP** (below).

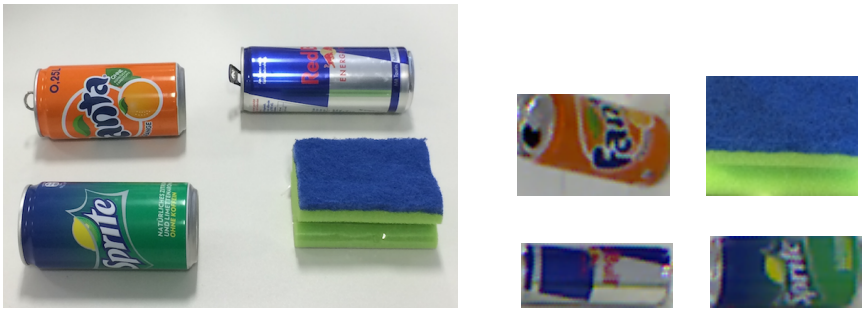


Figure 4.3.: Selected real objects (left) for generalization and their cropped images (right) for data augmentation .

(a) Background (b) Addition (c) Replacement

Figure 4.4.: Data augmentation process.

Data collection

Kinesthetic teaching is often used for data collection in PbD [59, 97]. In kinesthetic teaching, a teacher physically holds the robot's end-effector for generating the required motion. Since generating demonstrations via kinesthetic teaching can be a time consuming and a tedious task for a human operator, the already trained TP-DMP model introduced in the previous chapter is utilized for collecting the demonstrations. For the task considered in this paper, see [78] for details about data collection and training process of the TP-DMP. The dataset consists of 50 motions executed for different task settings. The entire data collection process by using TP-DMP took less than one hour. Each motion has 480 samples, which contain the clock signals, the forcing terms of the DMPs, the RGB images and the task parameters. 45 motions (21600 data-points) are selected for the training set while the remaining 5 motions (2400 data-points) are used for validating the learned model. RGB images were captured with a Kinect Xbox 360 camera. It has a resolution of 480×640 pixels (height$\times$width). The left parts of the images contain the robot's base and are irrelevant for the task. The right 480×480 pixels of the images are kept, as they are relevant for the task and then they are resized to 200×200 pixels for computational efficiency.

Data augmentation

In this work, the position of an object in an image is considered as a task parameter. Position of objects are often important when performing manipulation or reaching tasks. Here it is shown that if the task parameters are defined as the position of an object, then by using the proposed

data augmentation approach, the learned CNN can generalize for various different objects. For TP-DMP, this position is extracted by placing a marker on the object. The data augmentation process consists of first removing the marker from the image. This is done by covering the marker with a white patch of similar color as that of the table, as shown in Figure 4.4a. These marker-less images are now used as the background images for data augmentation. Now the data augmentation process is used for two purposes. Firstly for pretraining the network to predict the position of the objects in the images, by inserting the RGB image of a randomly selected real object, at a randomly selected position, as shown in Figure 4.4b. Secondly for learning to generate the motions for the real objects, by replacing the marker with a real object in the image, as shown in Figure 4.4c. The images of the four objects that are being used for data augmentation can be visualized in Figure 4.3. With data augmentation, the learned model generalizes for multiple real objects, without any need of detecting their positions. The objects positions have to be only provided during the training phase while the learned CNN directly uses the camera images during the motion reproduction, eliminating the need to extract the task parameters.

Network architecture

The architecture of the CNN is illustrated in Figure 4.5. It consists of nine layers, namely the input layer for image, three convolutional layers, a reduced layer of feature maps, input to fully connected layers, two fully connected layers and a linear output layer. The neural network takes a RGB image and a clock signal as inputs and predict the forcing terms of the DMPs. The input image has 120000 dimensions ($200 \times 200 \times 3$) as compared with a single value of the clock signal. The image vector can dominate over the clock signal due to its high dimensionality. To avoid this domination, the clock signal is concatenated with the extracted task parameters and then passed as an input to the fully connected layers. This is similar to what Levine et al. did with robot's state in their approach [62]. In order to avoid the vanishing gradient problem and for learning a good feature representation, the convolutional layers are pretrained to predict the object's position in the image.

Smaller sized kernels as in [40] can capture fine details from the image. Hence smaller kernels of size 3×3 are used, with the stride length of 1 and SAME padding, as shown in Figure 4.5. After convolutional layers a variant of soft attention mechanism called soft-argmax is used, which is introduced in [62]. The soft-argmax 'softpicks' out the position or indices of the maximum value within a matrix. It transforms a feature map into a probability matrix, then the probability matrix is element wise multiplied with the position matrices. The position matrices for x and y directions are defined by Eq. (4.1) and (4.2)

$$M_x = \left(\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 199 & 199 & \cdots & 199 \end{bmatrix} - 100.0 \right) / 100.0 \tag{4.1}$$

$$M_y = \left(\begin{bmatrix} 0 & 1 & \cdots & 199 \\ 0 & 1 & \cdots & 199 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 199 \end{bmatrix} - 100.0 \right) / 100.0 \tag{4.2}$$

Now the output positions are calculated as $x = \sum_{k,l}(softmax(F) \circ M_x)_{k,l}$, $y = \sum_{k,l}(softmax(F) \circ M_y)_{k,l}$ where $\circ$ denotes element wise product or Hadamard product, $k,l = [0...199]$ and F is a

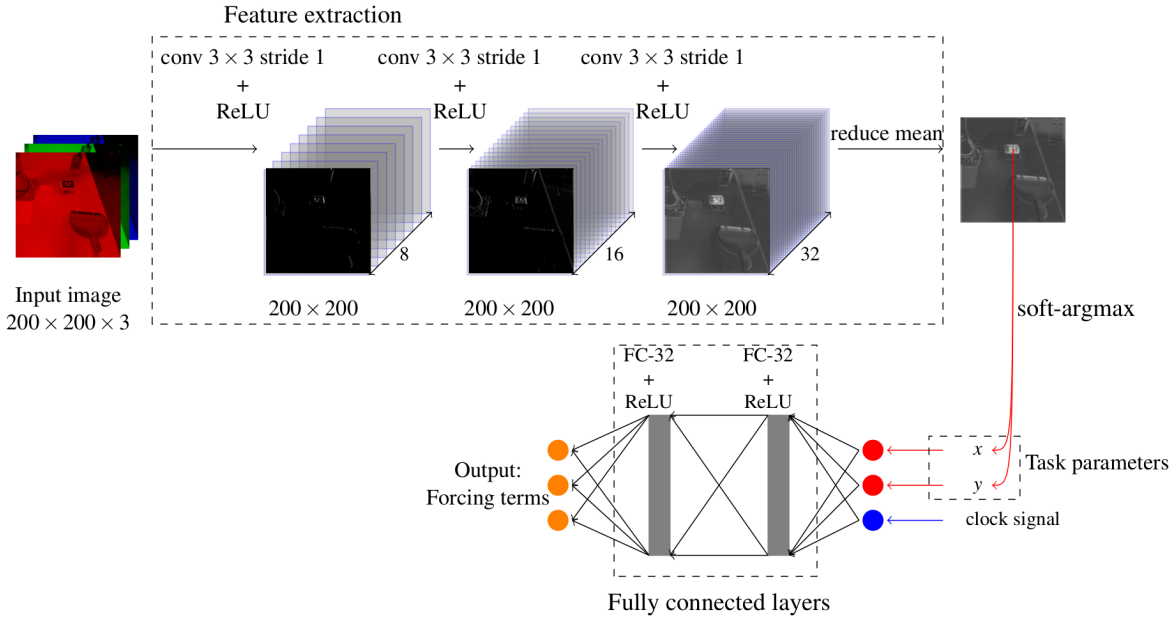


Figure 4.5.: Architecture of CNN.

single feature map, which has to be flattened before $softmax$, the result of $softmax$ has to be reshaped back to 200×200.

Unlike [62], an additional reduce_mean operator is added before the soft-argmax. The soft-argmax operator is executed on the single reduced feature map, so that it outputs only one position pair (x,y), which can be concatenated with a clock signal for further training. The reduce_mean operator is defined as $F(i,j) = (\sum_n F_n(i,j))/n$ where F denotes the reduced feature map, F_n is a single feature map in the last convolutional layer, $n = [1...32]$, and $i,j = [0...199]$ are the indices of pixels. All activation functions are Rectified Linear Unit (ReLU) in the neural network. The fully connected layers are pretrained with the positions of the marker and the clock signals as inputs and with the forcing terms as outputs. Once both the convolutional layers and the fully connected layers are pretrained, the pretrained weights are updated further with end-to-end training.

Adam optimizer is utilized for updating the weights of the CNN [46]. The first moment is set to be 0.9 and the second moment is set to be 0.999. The epsilon is set to be 1e-8 for numerical stability. The pixels intensities and the predicted task parameters are rescaled within the interval $[-1,1)$. The learning rate for pretraining convolutional layers of both object with marker and the real objects is 0.001. The learning rate for pretraining fully connected layers starts at 0.01 and ends at 0.0001, and is decayed by half after every 20 epochs. The learning rate for doing end-to-end training starts at 0.001 and ends at 0.0001. It is decayed by half after every 3 epochs. The loss function for all training steps is the mean squared error and the batch size for updating the weights is 100. The CNN is trained by using TensorFlow v1.0 on GPU GTX1060 6G and it takes about 4 hours to train one model. The forward pass of the CNN takes about 20ms with python 2.7 on GTX1060 GPU. For avoiding the overfitting problem, the model is validated after every 50 steps.

4.3. Experiments and Results

The proposed method is tested in a sweeping task. The goal of this task is to generate robot motion for moving a trash into the dustpan. The task parameters are defined as the position of the trash

in the image. The position of the trash governs the movement of the robot's end-effector. For a new trash position, the robot should be able to generate a trajectory for moving the trash to the collection point. Three DMPs are learned, two for generating a planar motion and one for encoding the rotary motion about the z-axis of the end-effector, as shown by the blue arrows in Figure 4.1.

4.3.1. Experimental setup

The experiments are conducted by using a KUKA light weight robot IV+. An already trained TP-DMP is used for collecting demonstrations for the **D-DMP**. For learning the TP-DMP, four demonstrations are collected via Kinesthetic teaching, by setting the robot to gravity compensation mode. For recording the position of the trash, a marker is attached on the object. The marker is tracked with Kinect RGB-D camera by using Robot Operating System (ROS) wrapper for Alvar, an open source augmented reality tag tracking library[1]. By using ROS a Server-Client communication interface is built between the CNN and the DMP as shown in Figure 4.6. The client PC (Personal Computer) on which the DMP model is running has to control the robot in real time. In order to avoid the computational burden on the client PC, the CNN is executed on another high performance server PC. The server PC receives the clock signal from the client PC. It then passes the current image and the received clock signal to the learned CNN, which then calculates the required forcing terms. The predicted forcing terms are then sent back to the client PC. Finally, the DMP uses the received forcing terms for the motion execution.

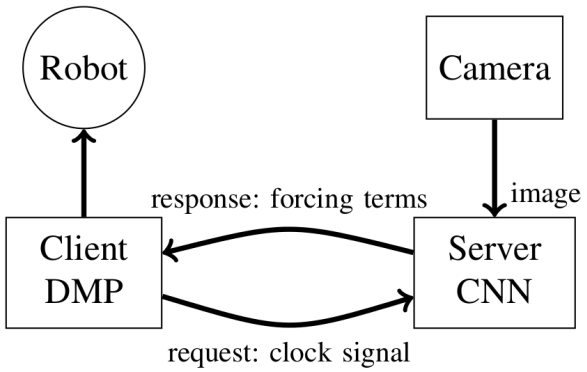
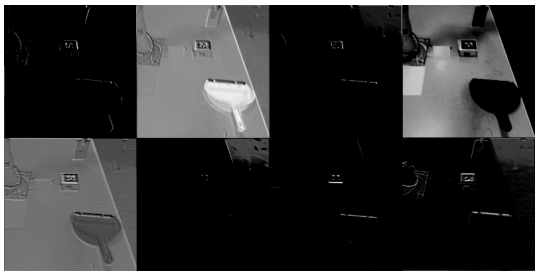


Figure 4.6.: Communication interface: interaction between DMP model and CNN using Server-Client pattern on ROS.

4.3.2. Feature maps

An example of the feature maps after ReLU operation of the three convolutional layers is shown in Figure 4.7. Since the size of feature maps is not decreased, the feature maps of each convolutional layers are human readable. Figure 4.7b shows one of the feature maps after the third convolutional layer. The target object (marker) has a strong activation in the reduced feature map. Visually all feature maps look very similar after the third convolutional layer and have only minor differences. Although these feature maps can be directly passed into the one of the soft-argmax operator for predicting the position of the object, the mean value of these feature maps is used for making the prediction more robust.

[1] `http://wiki.ros.org/ar_track_alvar`



(a) Feature maps after first convolutional layer.

(b) One feature map after third convolutional layer.

(c) Feature maps after second convolutional layer.

Figure 4.7.: Feature maps from the three convolutional layers.

4.3.3. Evaluation

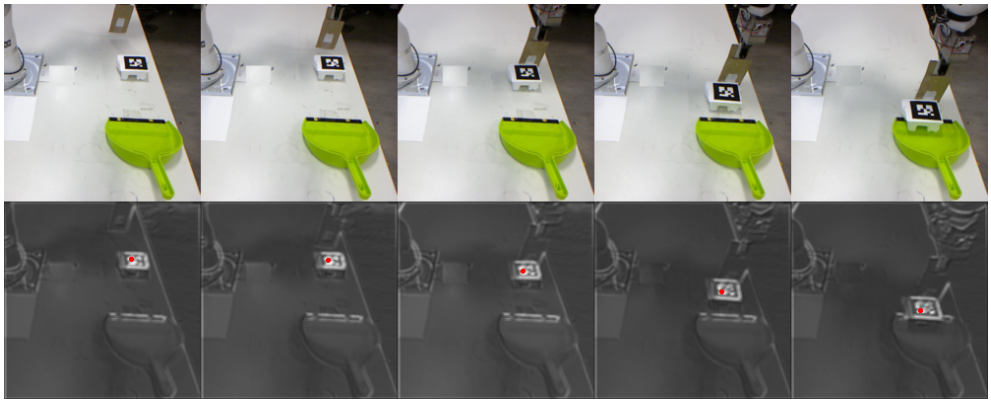
Now the errors and accuracies of predicted object's positions in the training and the validation set are calculated. The unit of error is pixel and the accuracy is stated in percentage. The (x, y) values are rescaled from range $[-1, 1)$ to $[0, 200)$ for calculating the error in pixels. Mean absolute error is calculated for the training set and the validation set. Errors in horizontal and vertical directions are evaluated separately. The resolution of the reduced feature map is 200×200, so 200 pixels are used as reference value for calculating the accuracy of predicted position by using Eq. (4.3). The errors and accuracies of the pretrained networks for predicting the positions of the marker and the objects are given in Tab. 4.1.

$$\text{Accuracy} = \big(1 - \text{error}/(200\text{pixels})\big) \times 100 \tag{4.3}$$

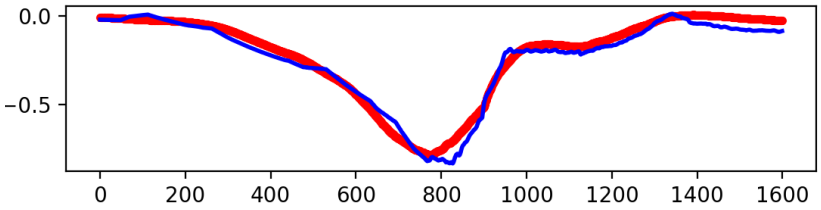
		Marker		**Real Objects**	
		horizontal	vertical	horizontal	vertical
Training	Error	1.87	4.56	3.23	8.28
	Accuracy	99.07	97.72	98.39	95.86
Validation	Error	1.83	4.39	3.44	8.36
	Accuracy	99.09	97.81	98.28	95.82

Table 4.1.: Errors (in pixel) and accuracies (%) of predicted positions in image coordinate system.

In order to check the performance of the CNN on real robot, the trained model is evaluated in different scenarios. Firstly it is tested that the CNN has learned the correct behavior from the collected dataset and can reproduce the results in [78]. The learned model is evaluated with the marker placed over the object. The successfully executed motion with images at different time steps and their corresponding reduced feature maps are visualized in Figure 4.8a. The feature maps have high activation at the marker. The red dot in the feature maps represents the predicted task parameters, which in this case is the object's position in the image. Since the TP-DMP can be



(a) The first row show the executed motion. The second row represents the reduced mean feature map of the images in the first row. The red dots in second row show the predicted marker position by the CNN.



(b) Generated forcing term for motion in x-axis by the TP-DMP (red curve) and the **D-DMP** (blue curve).

Figure 4.8.: Motion execution with a marker on the object.

Figure 4.9.: Motion execution for an object not used during the training. The red rectangle represents the bounding box of the initial trash positions in the collected dataset. The object is placed outside the range of training data for evaluating the extrapolation performance.

used for generating the forcing terms for an object with the marker, they are also being generated with the TP-DMP, while the motion is being executed by the **D-DMP**. Figure 4.8b shows the generated forcing term from the two models, for motion along x-axis. It can be seen that the graph of the predicted forcing terms from the **D-DMP** is smooth and closely matches to the graph of the TP-DMP.

Now the motions are executed for the objects without markers. The performance of the approach is also evaluated for an object which is not present in the training data. Since the object specific features are learned by the CNN, the learned skill can be generalized for the objects that look similar to the objects in the training data. The successfully executed motion for a new object can be visualized in Figure 4.9. The red rectangle in the most left image in Figure 4.9 depicts the range of the starting positions of the trash in the collected dataset. It can be seen in image sequences that the learned CNN can generalize for a real object, even when it is placed outside the range of the training data.

By combining DMP with a CNN, it inherits the desirable attributes associated with a DMP model. For instance Figure 4.10 shows that the motion can adapt for a change in the dustpan

Figure 4.10.: Motion execution for a different goal position.

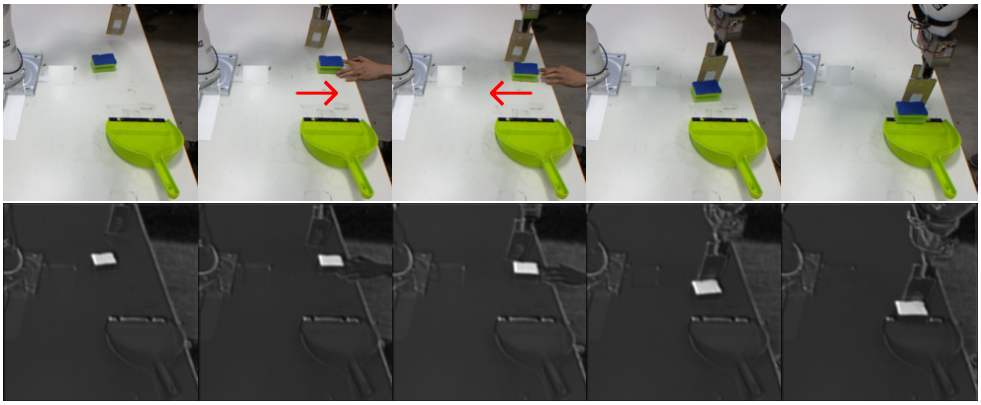


Figure 4.11.: Robust motion execution under perturbation (above). Corresponding reduced mean feature map (below).

position, by changing the goal value g in the DMP. The speed of the executed motion can be changed, by changing the decay rate α_s in Eq. (3.1).

The performance of the learned model against the disturbance in the image is also evaluated while changing the position of the sponge in real time. The results are shown in Figure 4.11. The sponge is pulled and pushed by a human hand during the task execution. The hand was never observed by the CNN in the training data and can be termed as a disturbance in the image. The end-effector of the robot follows the changing position of the sponge and neglects the disturbance in the image, by successfully pushing the sponge into the dustpan. A strong activation of the object of interest (sponge) can be visualized in the extracted feature maps. Since the shape and color of the hand are very different from the objects in the training data, the performance of the CNN remains unaffected by its introduction in the image.

4.4. Comparison with State-of-the-Art

In the presented approach, 45 motions are selected for the training set while the remaining 5 motions are used for validating the learned model. With 480 images in each demonstration, the training and testsets consist of 21600 and 2400 data-points respectively. This is lower than 800,000 grasp attempts in [62], at least 156 execution trials in [63] and 24,500 and 3500 data-points in training and validation sets respectively in [112]. The reason for the need of lower amount of training data for the **D-DMP** can be attributed to the hybrid DMP and CNN model where the CNN has to only learn the forcing terms of the DMP. Also the deep learning based motor primitive models have to be retrained for even a slight change in the execution speed [112]. In contrast, due to the underlying DMP model, the approach presented her can easily speed-up or slow-down the execution speed by simply changing the temporal scaling factor τ. Moreover, the learned motor primitive can also generalize for different objects, as shown in the experiments.

4.5. Discussion

This chapter proposes the **D-DMP** architecture, which is achieved by replacing the GMMs of a TP-DMP model with a CNN. As shown in the experiments, the training data for the **D-DMP** is collected by utilizing the already trained TP-DMP model introduced in the previous chapter. One can argue that why an already trained model is used for the collecting the training data. If a human teacher provides demonstrations for the **D-DMP**, then his presence will also be recorded in the training images. Since the camera images are utilized in the **D-DMP**, if a human is always present in the images during demonstrations, a CNN can learn human specific feature. Now during motion reproduction, if the human is not present in the image, then it can result in a failure of the task [62, 63, 112]. Alternatively a human can provide teleoperated demonstrations as in [76, 112]. In teleoperation mode a remote interface is utilized to control the robot. Since collecting teleoperated demonstrations can be a time consuming and a tedious task for a human operator, we use the already trained TP-DMP model to speed up the learning process.

In the presented approach task parameters are also recorded along with the training data. One may wonder that what would be the result of training the CNN model without utilizing the task parameters. Deep architectures usually have a high tendency of overfitting the training data. This is due to the vanishing gradient problem where the first few layers of a CNN get very little update as compared with the last few layers. This problem becomes more severe if the amount of training data is small. Due to this reason we utilize the collected task parameters to pretrain the convolutional layers of the CNN. In the pretraining, the initial convolutional layers are trained to predict the task parameters. Pretraining helps to initialize the weights of the initial layers for reducing the effect of vanishing gradient problem. Moreover the pretraining of inital layers is also helpful to decrease the number of learning iterations when performing end-to-end learning of the network. Due to all these reasons, the task parameters are recorded and utilized for pretraining the convolutional layers of the network. Once trained, the network comes up with an internal representation of the task parameters and thus do not require the extracted task parameters during the motion reproduction.

CNNs can learn task specific features. The experiments demonstrate that they can also reject disturbances in the camera images. Figure 4.11 shows that when a hand enters the image, the learned features did not activate that part of the image and treat it as the background. On the other hand the sponge in the image is highlighted. The reason is that the shape and color of the hand is very different from that of the objects which are learned. Another benefit of using image as an input is that the data augmentation can be applied on the images. This enables the network to generalize for arbitrary real objects, eliminating the need to use markers on the objects.

4.6. Summary and Future works

This work proposes **D-DMP**, which is learned by modeling the forcing terms of a TP-DMP with a CNN. Existing approaches for learning TP-DMP usually require specialized vision systems or markers for extracting task specific variables. In contrary, task specific features are learned by supervised learning in the presented approach. Also here the task parameters are only required during the training phase. After learning, the camera image is directly used for motion reproduction. The pretraining of the convolutional layers for extracting the task specific variables is also demonstrated. This reduces the effect of vanishing gradient problem as compared with directly doing end-to-end learning of the CNN. Since the features are learned, this eliminates the need to use any markers. Additionally by applying the data augmentation, the proposed approach can easily generalize the learned skill for various different objects. The generalization capability of

the **D-DMP** approach is demonstrated by executing the task for a novel object and against the background perturbations.

D-DMP requires relatively fewer training data for learning, as compared with other CNN based motor learning approaches [62,63,112]. Applying RL for learning a motor skill can require a lot of trials [63]. Thus the imitation strategy proposed in this work can be used to initialize the network parameters and for rapidly acquiring a motor skill, while the skill refinement can subsequently be done by using RL [98]. In this work the goal position g is provided by the user. Alternatively, it can be also predicted by the learned network, along with the forcing terms of the DMPs.

Currently the time signal and the image are the only inputs to the network. Additional information such as state of the robot can also be passed as an input to the network. This can enable the learned network to get an awareness about the robot's state, which might be helpful for the execution of the task. Another extension of the the proposed approach can be to eliminate recording of the task parameters, by applying the dimensionality reduction techniques, to come up with a suitable lower dimensional representation of the images. The lower dimensional representation can then be utilized directly as an input to the fully connected layers. An unsupervised deep learning architecture that can be investigated for this purpose is an autoencoder [110]. If the network can only utilize the camera images for learning then this will eliminate the need to identify the task parameters by a human and hence making the approach more self sustaining.

The presented work has used a CNN. Another interesting approach that can be investigated is to use a Recurrent Neural Network (RNN) [68] instead of a CNN. RNN can model the time dependence in a signal and have been shown to work well for speech recognition, text generation and forecasting applications. They can utilize a sequence of images and have the capability to model the dynamical systems [34]. By utilizing a sequence of images, the temporal structure in the tasks can be an exploited, eliminating the need to pass time signal as an additional input for motion reproduction.

CHAPTER 5

Learning in Teleoperation

5.1. Motivation

The last two chapters emphasized on learning task parameterized movement primitives by utilizing kinesthetic teaching for collecting demonstrations. Since a robot and a human cannot always be colocated, teleopeartion can be utilized to control a robot in such circumstances. While teleoperation provides a possibility for a robot to operate at extreme conditions instead of a human, teleoperating a robot still demands a heavy mental workload from a human operator. Programming by Demonstrations can reduce the human operator's burden by learning repetitive teleoperation tasks. Even though there have been several prior researches on learning via teleoperation, most of them were focused on applying PbD approaches developed for kinesthetic teaching, undermining the need to address the problems encountered when using teleoperation for collecting demonstrations. For instance teleoperation can produce demonstrations with large spatial and temporal variations, with different starting/ending phases and partial executions of the task. To the best of my knowledge, there has been no research on developing learning from demonstration approach which can handle inconsistent demonstration with large spatial and temporal variations, demonstration with different starting/ending phases and partial demonstrations of the task.

This chapter proposes a novel PbD method which aim at identifying and overcoming the challenges associated with using teleoperation as an input modality for PbD. It first compares kinesthetic teaching with teleoperation and highlights some inherent problems associated with teleoperation; specifically uncomfortable user interface and inconsistent teleoperated demonstrations. A novel approach for handling the unique features encountered during teleoperation is presented in this chapter. The approach is based on Dynamic Movement Primitives (DMP) for learning the demonstrations provided by an operator during teleoperation. As opposed to other papers which use Dynamic Time Wrapping (DTW) as a preprocessing step, the presented approach aligns and encodes the motion data simultaneously and incorporates the temporal and spatial variance presented by using an EM algorithm [28]. Although the task parameters are not explicitly considered in this chapter, the work presented here can be combined with the task parameterized skill learning approach presented in chapter 3, to simultaneously align and encode the task parameterized demonstrations. The utilization of the learned model to the shared teleoperation settings is also addressed in this chapter. In shared teleoperation, the learned model control some DOFs, while the remaining DOFs are controlled by the human operator/operators. The synchronization of the

generated motion of learned model and the human operators is the main challenge which is encountered in shared teleoperation. To address this challenge we propose two shared teleoperation architectures. The performance of the two architectures is tested and validated against human-only teleoperation arcitecture with real robot experiments of a peg-in-hole task conducted on a 3-Degree of freedom (DOF) master-slave teleoperation system.

5.2. Teleoperated demonstrations

While comparing kinesthetic teaching and teleoperation, the former is reported to be easier to use, as a faster way to provide the demonstrations and a more preferred way of demonstrating a task [4, 32]. Additionally kinesthetic teaching is also reported to provide more successful demonstrations in a shorter time duration. This is due to a lack of situational awareness and difficulty of controlling the robot with teleoperation. By using the setup shown in the Figure 5.1, an operator is asked to demonstrate four cycles of the peg-in-hole task. Figures 5.2a and 5.2b illustrate the demonstrations of the peg-in-hole task recorded via kinesthetic teaching and teleoperation. It can be seen in Figures 5.2c and 5.2d that the demonstrations obtained using kinesthetic teaching have a higher level of consistency as compared to the demonstrations recorded via teleoperation. Also trajectories obtained via teleoperation have a higher level of spatial-temporal variations.

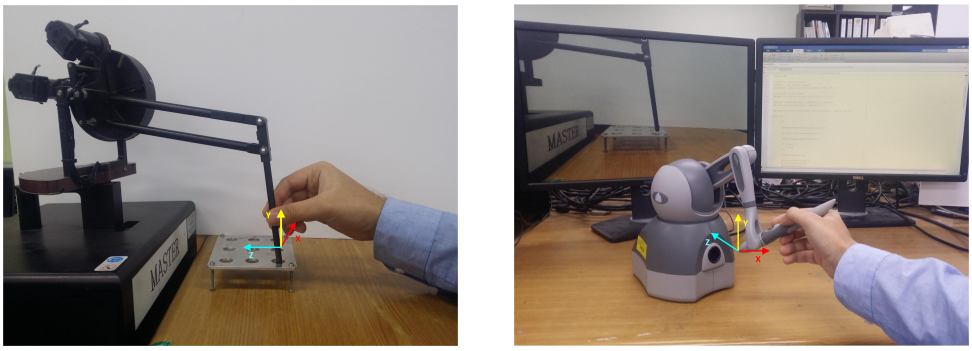
Apart from the higher level of spatial-temporal variations, demonstrations via teleoperation can pose additional challenges. For instance, due to the symmetric nature of the peg-in-hole task, a change in camera position can cause a user to lose track of a single starting region while providing the demonstrations. This can result in different initial and final configurations for different demonstrations. Also an operator might prematurely terminate his/her motion due to communication issues or a sensor failure, yielding an incomplete demonstration. All these issues make it challenging to apply existing PbD techniques directly for teleoperation.

Most of the existing PbD literature for teleoperation focuses on DTW [96] based motions alignment before encoding the demonstrations. DTW is an algorithm which measure the similarities between two temporal signals executed at different speeds. However, DTW does not provide a good solution to the aforementioned problems because it assumes the same initial and final states of the signals, executed at different speeds. Another limitation of GMM/HMM based PbD approaches for teleoperation is that unlike nonlinear dynamical system based approaches, such as DMP [99], the HMM/GMM based encoding with GMR based trajectory generation does not guarantee the stability properties associated with dynamical systems [18, 83]. The above mentioned reasons raise a scalability issue, when applying PbD approaches developed for kinesthetic teaching to teleoperation. Hence there is a need to develop PbD approaches for handling inconsistencies encountered during demonstrations via teleoperation.

5.3. Teleoperation Skills Learning

5.3.1. DMP learning with GMM/GMR

In the original DMP, locally weighted regression technique is applied in order to learn the nonlinear forcing term $\mathcal{F}(s)$. In this work, the forcing terms are encoded with a GMM. A GMM with k components is parameterized by $\boldsymbol{\Theta}_{(k)} = \{\pi_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}_{m=1}^{k}$ where $\pi_1, \ldots, \pi_k$ are mixing weights, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ are means and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$ are covariance matrices. The probability density function of $\mathbf{y}$ is said to follow the k-component GMM if it can be written as $\mathcal{P}(\mathbf{y}|\boldsymbol{\Theta}_{(k)}) = \sum_{m=1}^{k} \pi_m \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, subject to the constraints $0 < \pi_m < 1$ and $\sum_{m=1}^{k} \pi_m = 1$. After learning the GMM, the forcing term is synthesized by using the GMR. When human demonstrations are



(a) (b)

Figure 5.1.: (a) Kinesthetic teaching by directly holding the robot, (b) Teleoperation by using 3-DOF haptic device

collected, each demonstration trajectory is linearly re-sampled to have n number of samples. For each trajectory, the required forcing terms are calculated by rearranging the terms in the DMP equation i.e. $\mathcal{F}(s) = \dot{v}/(\tau a) - (\alpha_x(\beta_x(g-x) - v))/a$. Then, there is a sequence of a pair, consisting of position x and forcing terms $\mathcal{F}(s)$. Assuming that the corresponding phase variable s is known, now the joint distribution of these variables is encoded by using a GMM. During the reproduction phase, GMR is used for predicting the forcing term for a given phase signal s, which is plugged into the DMP equation to get the acceleration command. A separate controller is used for executing the motion.

5.3.2. EM algorithm for learning from asynchronous data

As discussed earlier, the demonstrations can have temporal variations and thus the phase signal cannot be attached with each trajectory. In order to handle such inconsistent and asynchronized teleoperated demonstration trajectories, the demonstrations are first separated into two parts: one reference trajectory and the rest. The reference trajectory should be a full execution of the motion and it is recommended to be a minimum jerk trajectory.

Assume that there are k demonstration trajectories and the first demonstration is the reference trajectory. From the demonstrations, there are two data sets. The first data set contains only the reference trajectory, its forcing terms and the concatenated phase signals. It is called as a complete data ($\mathbf{X}^{Com}$). The second data set contains the remaining trajectories and their forcing terms. However their phase variables s are unknown. Thus, it is termed as incomplete data ($\mathbf{X}^{InCom}$). The missing phase signals in the second data set will be estimated by synchronizing this data set with the first data set iteratively during EM. The notation $\mathbf{x}_i^{Com}$ and $\mathbf{x}_i^{InCom}$ is used to denote i^{th} column in complete ($\mathbf{X}^{Com}$) and incomplete data sets ($\mathbf{X}^{InCom}$) respectively.

$$\mathbf{X}^{Com} = \mathbf{X}_1 = \begin{bmatrix} \mathcal{F}_1(s_0) & x_{1,0} & s_0 \\ \vdots & \vdots & \vdots \\ \mathcal{F}_1(s_n) & x_{1,n} & s_n \end{bmatrix}^\top \quad \mathbf{X}^{InCom} = \begin{bmatrix} \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_k \end{bmatrix}^\top$$

Now a GMM is fitted to the data sets by using an EM algorithm. The GMM parameters and the missing phase signals in each trajectory should be estimated. These parameters are initialized and are then iteratively updated during EM. The values of phase signals are initialized as a linearly increasing value from 0 to 2π in each trajectory for a rhythmic DMP and an exponentially decreasing value from 1 to 0 in each trajectory for a discrete DMP. Now there are n data-points in complete data set and $n_2 = n * (k-1)$ data-points in incomplete data set.

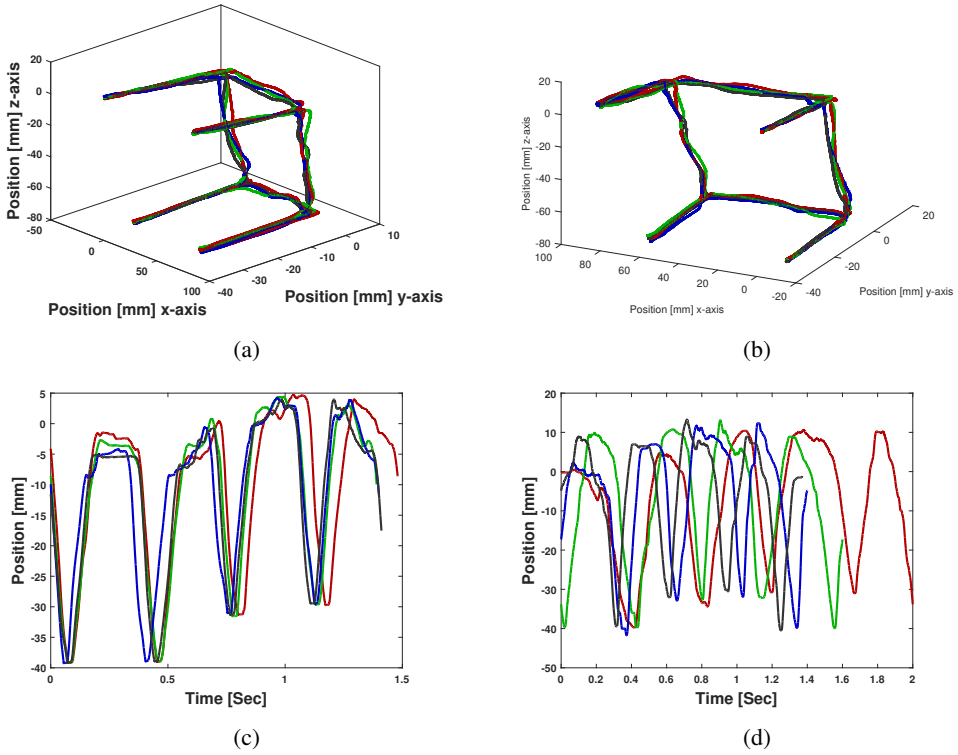


Figure 5.2.: Four demonstrations recorded via kinesthetic teaching and teleoperation for the peg-in-hole task (a) Recorded trajectories via kinesthetic teaching (b) Recorded trajectories via teleoperation (c) Motion in y-axis using kinesthetic teaching (d) Motion in y-axis using teleoperation.

E-step

First the variables are separated into two types, missing (*miss*) denoting the phase variable and observable (*Obs*) denoting all variables other than the phase variable.

$$\boldsymbol{\mu}_m = \begin{bmatrix} \boldsymbol{\mu}_m^{Obs} \\ \boldsymbol{\mu}_m^{miss} \end{bmatrix}, \boldsymbol{\Sigma}_m = \begin{bmatrix} \boldsymbol{\Sigma}_m^{Obs} & \boldsymbol{\Sigma}_m^{Obs,miss} \\ \boldsymbol{\Sigma}_m^{miss,Obs} & \boldsymbol{\Sigma}_m^{miss} \end{bmatrix}$$

In case of a rhythmic DMP, first map all the phase variable within the interval $[\mu_m^{miss} - \pi, \mu_m^{miss} + \pi]$, for calculating the valid probabilities for the m^{th} GMM component.

$$\begin{aligned} p_{i,m} &= \pi_m \mathcal{N}(\mathbf{x}_i^{Com}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \\ q_{j,m} &= \pi_m \mathcal{N}(\mathbf{x}_j^{InCom}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \end{aligned}$$

where the initialized (or updated during M-step) phase variables are used for probabilities calculation in incomplete data set. The responsibility terms for the i^{th} and j^{th} data points in complete and incomplete data sets are calculated as:

$$E[z_{i,m}|\mathbf{x}_i^{Com}, \boldsymbol{\theta}_t] = h_{i,m} = \frac{p_{i,m}}{\sum_{l=1}^{k} p_{i,l}} \quad , \quad h_m = \sum_{l=1}^{n} h_{l,m}$$

$$E[z_{j,m}|\mathbf{x}_j^{InCom}, \boldsymbol{\theta}_t] = r_{j,m} = \frac{q_{j,m}}{\sum_{l=1}^{k} q_{j,l}} \quad , \quad r_m = \sum_{l=1}^{n_2} r_{l,m}$$

In incomplete data set, the prediction of the j^{th} missing value with respect to the m^{th} GMM component is done as:
$\hat{x}_{j,m}^{miss} = \mu_m^{miss} + \Sigma_m^{miss,Obs}(\Sigma_m^{Obs})^{-1}(\mathbf{x}_j^{InCom,Obs} - \mu_m^{Obs})$
With this predicted value, two additional expectations are calculated for the incomplete data set:

$$E[z_{j,m}, x_j^{InCom,miss}|\mathbf{x}_j^{InCom,obs}, \theta_t] = q_{j,m}(\hat{x}_{j,m}^{miss})$$
$$E[z_{j,m}, x_j^{InCom,miss} x_j^{InCom,miss^\top}|\mathbf{x}_j^{InCom,obs}, \theta_t] =$$
$$q_{j,m}(\Sigma_m^{miss} - \Sigma_m^{miss.obs}\Sigma_m^{obs^{-1}}\Sigma_m^{miss.obs^\top} + \hat{x}_{j,m}^{miss}\hat{x}_{j,m}^{miss^\top})$$

M-step

The mixing weights are updated as:

$$\pi_m = \frac{h_m + r_m}{n + n_2}$$

while the GMM means are updated as:

$$\mu_m = \frac{\sum_{l=1}^{n} \mathbf{x}_l^{Com} h_{l,m} + \sum_{v=1}^{n_2} \mathbf{x}_v^{InCom} r_{v,m}}{h_m + r_m}$$

In case of a rhythmic DMP, the phase signal lies on a circular plane for whcih mean of cos and sin terms is needed:

$$\bar{cx} = \frac{\sum_{l=1}^{n} \cos(x_i^{Com,miss}) h_{l,m} + \sum_{v=1}^{n_2} \cos(x_i^{InCom,miss}) r_{v,m}}{h_m + r_m}$$
$$\bar{sx} = \frac{\sum_{l=1}^{n} \sin(x_i^{Com,miss}) h_{l,m} + \sum_{v=1}^{n_2} \sin(x_i^{InCom,miss}) r_{v,m}}{h_m + r_m}$$

Afterwards the phase variable for a rhythmic DMP is updated in the GMM means with these conditions and wrapped in the interval $[0, 2\pi]$:

$$if(\bar{cx}) < 0) \qquad \mu_m^{miss} = \tan^{-1}\frac{\bar{sx}}{\bar{cx}} + \pi$$
$$elseif(\bar{sx}) > 0) \qquad \mu_m^{miss} = \tan^{-1}\frac{\bar{sx}}{\bar{cx}}$$
$$else \qquad \mu_m^{miss} = \tan^{-1}\frac{\bar{sx}}{\bar{cx}} + 2\pi$$

As before, phase variable is mapped within the interval $[\mu_m^{miss} - \pi, \mu_m^{miss} + \pi]$ for a rhythmic DMP. Afterwards the covariances are updated as:

$$\Sigma_m = \frac{\sum_{i=1}^{n} h_{i,m}(\mathbf{x}_i^{Com} - \mu_m)(\mathbf{x}_i^{Com} - \mu_m)^\top + \sum_{l=1}^{n_2} \mathbf{A}_l}{h_m + r_m}$$

$$\mathbf{A}_l = r_{l,m} \begin{bmatrix} \mathbf{x}_l^{InCom,obs} - \mu_m^{obs} \\ x_l^{InCom,miss} - \mu_m^{miss} \end{bmatrix} \begin{bmatrix} \mathbf{x}_l^{InCom,obs} - \mu_m^{obs} \\ x_l^{InCom,miss} - \mu_m^{miss} \end{bmatrix}^\top$$
$$= \begin{bmatrix} \mathbf{A}_{11} & \mathbf{a}_{12} \\ \mathbf{a}_{21} & a_{22} \end{bmatrix}$$

where

$$\begin{aligned}
\mathbf{A}_{11} &= r_{l,m}(\mathbf{x}_l^{InCom,obs} - \boldsymbol{\mu}_m^{obs})(\mathbf{x}_l^{InCom,obs} - \boldsymbol{\mu}_m^{obs})^\top \\
\mathbf{a}_{21} &= (E[z_{l,m}, x_l^{InCom,miss}] - r_{l,m}\mu_m^{miss})(\mathbf{x}_l^{InCom,obs} \\
&\quad -\boldsymbol{\mu}_m^{obs})^\top \\
\mathbf{a}_{12} &= \mathbf{a}_{21}{}^\top \\
a_{22} &= E[z_{l,m}, x_l^{InCom,miss} x_l^{InCom,miss^\top}] + \\
&\quad r_{l,m}\mu_m^{miss}(\mu_m^{miss})^\top - 2E[z_{l,m}, x_l^{InCom,miss}](\mu_m^{miss})^\top
\end{aligned}$$

After updating the GMM parameters the next step is to maximize the phase signal values in the incomplete data set. This is done by using Gaussian Mixture Regression but with responsibilities calculated using all of the dimensions (phase values from initialization or from last M-step). For a given input variable $\mathbf{x}_i^{InCom,obs}$ and a given Gaussian distribution m, the expected value of $x_i^{InCom,miss}$ is defined by:

$$\mathcal{P}(x_i^{InCom,miss}|\mathbf{x}_i^{InCom,obs}, m) = \hat{x}_{i,m}$$

where

$$\hat{x}_{i,m} = \mu_m^{miss} + \boldsymbol{\Sigma}_m^{miss,obs}(\boldsymbol{\Sigma}_m^{obs})^{-1}(\mathbf{x}_i^{InCom,obs} - \boldsymbol{\mu}_m^{obs})$$

By considering the complete GMM

$$E(x_i^{InCom,miss}|\mathbf{x}_i^{InCom,obs}) = \sum_{l=1}^{k} r_{i,l}\hat{x}_{i,l}$$

$$\text{with } r_{i,m} = \frac{\pi_m \mathcal{N}(\mathbf{x}_i^{InCom}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(\mathbf{x}_i^{InCom}; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$$

where $E(x_i^{InCom,miss}|\mathbf{x}_i^{InCom,obs})$ is the updated phase value of the i^{th} data point in incomplete data set.

5.4. Shared teleoperation

The goal of skill learning is to decrease the effort which a human operator has to put in for executing the task. Although the autonomy of a task relaxes the workload of a human operator, the issue of correct task execution has to be addressed for effective assistance [29]. Alternatively multiple operators can be deployed for the task execution, which in comparison to a single operator, provides a decrease in cognitive workload of each operator and a reliable execution of the task [107]. The issue of dividing the control authority among multiple operators with multiple Field-of-Views (FOVs) is addressed by [107]. However, introducing multiple operators for controlling a single robot is an economically expensive solution. Also during control sharing among multiple operators, a camera failure or obstruction in FOV of one operator can make it impossible for that operator to execute the task.

There can be cases where some DOFs in a task are repetitive, while the remaining DOFs have non repetitive motion. In such cases, we can introduce shared teleoperation instead of either fully manual teleoperation or autonomous execution of the task. The DOFs which have repetitive motion, can be encoded by LfD and executed autonomously through the artificial agent, and the other DOFs, which are either non repetitive or highly uncertain, can be controlled by a human operator. That way we can expand the application area of the proposed skills learning method to more general tasks by removing the constraint that all DOFs should be fully repetitive.

However, in order to perform human-agent shared teleoperation, the main challenge is how to synchronize the generated motion of the learned agent and the motion command from the human operator. To address this problem, two human-agent control sharing strategies are proposed i.e. human-synchronized shared teleoperation and agent-synchronized shared teleoperation.

5.4.1. Human-synchronized Shared Teleoperation

In the human-synchronized shared teleoperation, the agent controls the motion of its DOFs, without taking into account the motion of DOFs controlled by the human. All the synchronization effort is performed by the human operator, where the agent does not depend on the former's spatial information for the execution of task. The control flow of the human-synchronized shared teleoperation architecture can be visualized in Figure 5.3, where the human operator utilizes the visual and haptic feedback coming from the slave's end and the agent, to synchronize his/her motion while sharing control with the learned artificial agent.

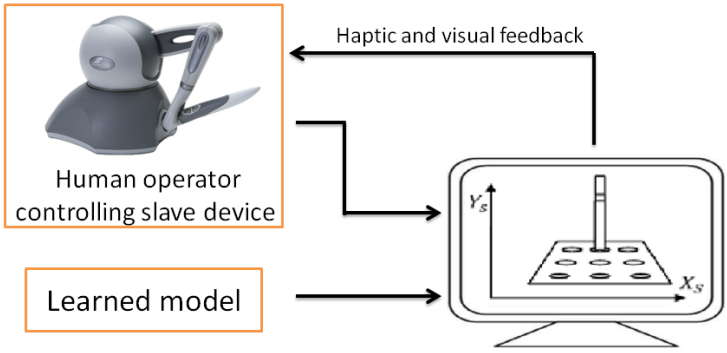


Figure 5.3.: Human-Synchronized Shared Teleoperation.

5.4.2. Agent-synchronized Shared Teleoperation

In agent-synchronized shared teleoperation, the learned agent (DMP) also actively participates in synchronizing its motion with the human operator, as shown in Figure 5.4. The basic idea in this architecture is to predict the DMP's clock signal, for the given robot configuration, which is then utilized for generating the motion of the DOF to be automated. Hence the learned agent receives task feedback to adjust its motion.

Table 5.1 presents the pseudo-code for the shared teleoperation in between the DMPs and the human operator/operators. The basic idea for shared teleoperation is to predict the clock signal for the given robot configuration, which is then utilized for generating the motion of the DOFs to be automated. The initial clock signal is predicted by a numerical maximum likelihood approach for the current end-effectors configuration. Linearly spaced samples of phase variable are generated and the likelihood value of the generated samples along with the current end-effector configuration is evaluated for the learned GMMs. The sample which yields the maximum likelihood value is selected as the predicted starting value for the clock signal.

For generating the motion of the autonomous DOFs, it is important to synchronize them with the motion of the DOFs to be controlled by the human. For that the current clock signal value is re-estimated after motion command at each time step. To achieve this the clock signal value is generated within a predefined window around the current estimated value. The generated clock

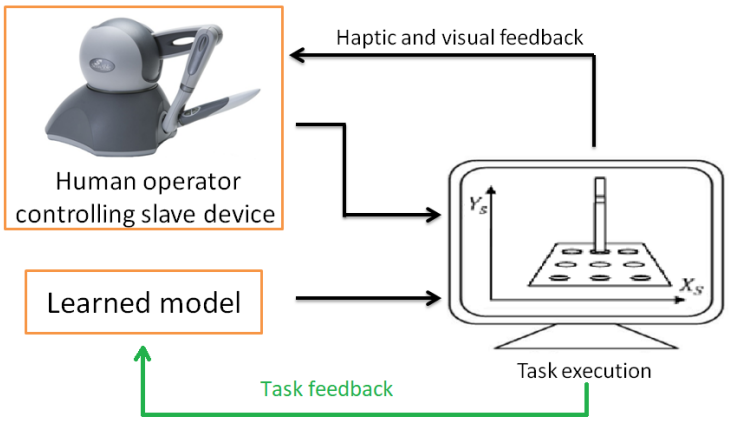


Figure 5.4.: Agent-Synchronized Shared Teleoperation.

signal values are combined with the current robot configuration and then again a numerical maximum likelihood approach is deployed along the DOFs controlled by the human. The sample that maximizes the likelihood value is now used for updating the clock signal. The clock signal is an increasing function for a rhythmic DMP. If the sample value is greater than the current estimate, the current estimate is updated with the sample maximizing the likelihood. The clock signal for the next time step is predicted by increasing the current estimate with an average increase of clock signal observed in the previous time steps. If the value of the sample maximizing the likelihood is less than the current estimate, the current estimate is kept as it is, as the clock signal is an increasing variable and increment the current estimate by a small predefined amount Δ for the next time step. The same approach for the discrete DMPs is applied but with decreasing values, instead of increasing values of the clock signal. A concise block diagram of the shared teleoperation controller is presented in Figure 5.5.

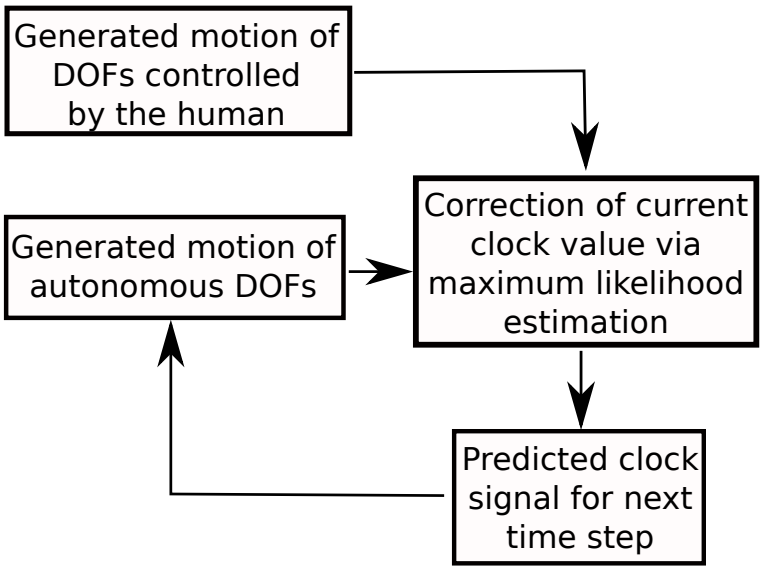


Figure 5.5.: Block diagram of agent-synchronized shared teleoperation.

Table 5.1.: Shared teleoperation for rhythmic motion by using the learned GMMs

Input: $\boldsymbol{\Theta 1} = \{\{\pi_{m,d_1}, \boldsymbol{\mu}_{m,d_1}, \boldsymbol{\Sigma}_{m,d_1}\}_{m=1}^{k}\}_{d_1=1}^{D_1}, \sigma_s, N_2, N, \Delta, \delta,$

$\boldsymbol{\Theta 2} = \{\{\pi_{m,d_2}, \boldsymbol{\mu}_{m,d_2}, \boldsymbol{\Sigma}_{m,d_2}\}_{m=1}^{k}\}_{d_2=1}^{D_2}$

Initialize: $\text{Step}_{\texttt{sum}} = 0, l = 0$

Predict initial clock value

$s_{\texttt{Samp}} = [0 \;\; \delta \; 2\delta \; \ldots \; 2\pi]'$ ▷ Generate N_2 samples

for $o = 1$ to N_2 **do**

  for $d_1 = 1$ to D_1 **do** ▷ DOFs controlled by operator

    $\text{Temp}_{d_1} = \sum_{m=1}^{k} \pi_{m,d_1} \mathcal{N}([x_{d_1} \; s^o_{\texttt{Samp}}]'; \boldsymbol{\mu}^{[2\ 3]}_{m,d_1}, \boldsymbol{\Sigma}^{[2\ 3]}_{m,d_1})$

  end for

  for $d_2 = 1$ to D_2 **do** ▷ DOFs to be automated

    $\text{Temp}_{d_2} = \sum_{m=1}^{k} \pi_{m,d_2} \mathcal{N}([x_{d_2} \; s^o_{\texttt{Samp}}]'; \boldsymbol{\mu}^{[2\ 3]}_{m,d_2}, \boldsymbol{\Sigma}^{[2\ 3]}_{m,d_2})$

  end for

  $\text{Prob}_o = \mathcal{N}(s^o_{\texttt{Samp}}; s^o_{\texttt{Samp}}, \sigma_s^2) \prod_{d_1=1}^{D_1} \text{Temp}_{d_1} \prod_{d_2=1}^{D_2} \text{Temp}_{d_2}$

end for

$\hat{o} = \underset{o}{\text{argmax}}\{\text{Prob}_o, o = 1, \ldots, N_2\}$

$s_{\texttt{cur}} = s^{\hat{o}}_{Samp}$

while (GenerateMotion==1) **do**

  Generate motion

  ForcingTerms $\leftarrow$ GMR$(s_{\texttt{cur}}, \boldsymbol{\Theta 2})$ ▷ For autonomous DOFs

  AccelerationCommand$\leftarrow$DMPs(ForcingTerms)

  Re-estimate clock signal

  $s_{\texttt{Samp}} = [s_{\texttt{cur}} - \delta \;\; s_{\texttt{cur}} \; \ldots \; s_{\texttt{cur}} + N\delta]'$ ▷ Generate $N+2$ samples

  for $o = 1$ to $N+1$ **do**

    for $d_1 = 1$ to D_1 **do** ▷ DOFs controlled by operator

      $\text{Temp}_{d_1} = \sum_{m=1}^{k} \pi_{m,d_1} \mathcal{N}([x_{d_1} \; s^o_{\texttt{Samp}}]'; \boldsymbol{\mu}^{[2\ 3]}_{m,d_1}, \boldsymbol{\Sigma}^{[2\ 3]}_{m,d_1})$

    end for

    $\text{Prob}_o = \mathcal{N}(s^o_{\texttt{Samp}}; s^o_{\texttt{Samp}}, \sigma_s^2) \prod_{d_1=1}^{D_1} \text{Temp}_{d_1}$

  end for

  $\hat{o} = \underset{o}{\text{argmax}}\{\text{Prob}_o, o = 1, \ldots, N+1\}$

  $l = l + 1$

  if $s^{\hat{o}}_{Samp} < s_{\texttt{cur}}$ **then**

    $\text{Step}_{\texttt{sum}} = \text{Step}_{\texttt{sum}} + \Delta$

    $s_{\texttt{cur}} = s_{\texttt{cur}} + \Delta$

  else

    $\text{Step}_{\texttt{sum}} = \text{Step}_{\texttt{sum}} + (s^{\hat{o}}_{Samp} - s_{\texttt{cur}})$ ▷ Correct clock signal

    $s_{\texttt{cur}} = s^{\hat{o}}_{Samp} + \frac{\text{Step}_{\texttt{sum}}}{l}$ ▷ Predict clock signal

  end if

end while

5.5. Comparison with State-of-the-Art

The proposed approaches are evaluated by using a peg-in-hole task with a master-slave teleoperation system. The setup consists of SensAble PHANToM Omni as a master device and PHANToM Premium 1.5A as a slave. An aluminum plate with 4 holes as shown in Figure 5.6 is used for the peg-in-hole task. The operator observed the visual feedback through video displayed on the monitor and controlled the slave with the same orientation as the view from the camera. One execution cycle constitutes of inserting the robot end-effector into the four holes in counter-clockwise direction, while starting and ending above the same hole. During the data collection phase, slave records the Cartesian positions, which are being used for calculating the velocities and accelerations through numerical differentiation. Separate DMPs are learned for each considered degree of freedom i.e. the x, y and z-axis. All the models used phase signal as input and 20 Gaussians in the GMMs. The number of Gaussians in the GMMs can also be optimized [77]. Since the amplitude modification is not required, the amplitude modifier term a is set in the DMP equation to be 1.

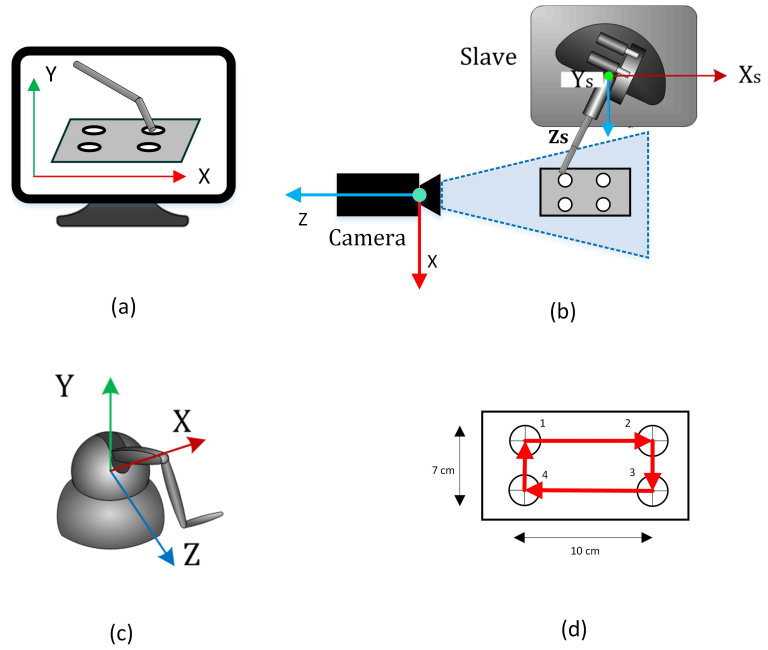
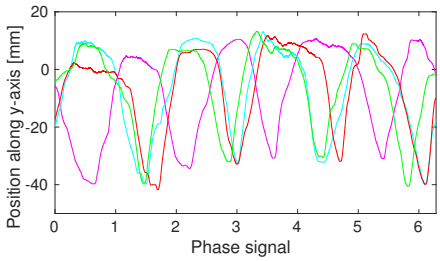


Figure 5.6.: (a) Visual feedback, (b) slave with peg-in-hole task, (c) haptic device and (d) peg-in-hole task.

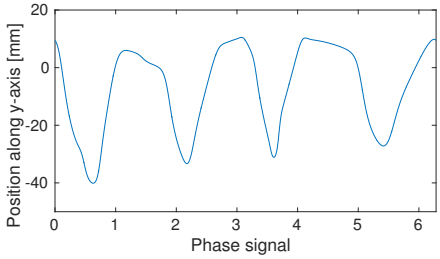
5.5.1. Experiment 1: Temporal and spatial variations

An operator demonstrated four counter-clockwise cycles of the peg-in-hole task as shown in Figures (5.8a-5.8d). The operator started and ended above the same hole. An operator needs some time to reach an adequate level of performance in teleopeartion and thus first few demonstrations can have large temporal and spatial variations [14], as visible in Figure 5.2d.

Figure 5.8e contains the result of PbD method presented in [3, 19] where the trajectories are first aligned by using DTW. Afterwards the phase signal and the spatial data are encoded by using a GMM. Finally GMR is used for motion retrieval. When applying GMR for a rhythmic task, the data for the circular dimension (phase signal) is always mapped in the interval $\mu - \mathbb{X}_i$ and $\mu + \mathbb{X}_i$ for calculating the valid responsibilities. The downsides of this approach are that firstly the value of phase signal cannot be inferred for arbitrary starting point of the end-effector using

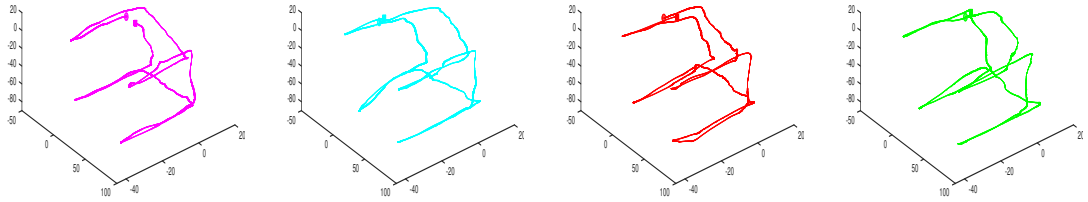


(a)

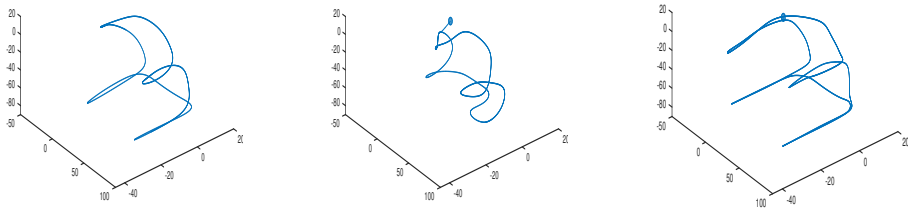


(b)

Figure 5.7.: Experiment 1: (a) Demonstrated motions and (b) learned trajectory for y-axis plotted against phase signal.



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



(e) (f) (g)

Figure 5.8.: All axis are in millimeters. (a-d) Recorded motions for experiment 1, with starting and ending points in each trajectory indicated by a circle and a square respectively. Same starting point and ending point representation is also used in the other two experiments. (e-g) Motion reproductions with different models (e) DTW+GMM based encoding of spatial data with GMR based motion reproduction, (f) DMP model with DTW+GMM based encoding of forcing terms, (g) Our approach.

GMR. This is because the mapping from an output to input of a function is not guaranteed to be one to one. Secondly with phase signal as the only input, the GMM-GMR based encoding does not consider the current position of the end-effector when generating the motion. The learned

trajectory partially completes the task by inserting peg in three out of four holes, as shown in Figure 5.8e.

The next model that is considered in the DMP based encoding of the demonstrations, with phase signal and forcing terms encoded by using a GMM [6]. Again the forcing terms of different trajectories are first aligned with DTW. This approach fails to reproduce the task, as shown in Figure 5.8f. Because of dissimilarity in the forcing terms of different trajectories, the DTW fails to align them properly.

Figure 5.8g shows the result of the proposed approach which successfully reproduces the task due to the temporal alignment performed during EM. The asynchronous nature of the motions and the learned motion along y-axis can be visualized in Figure 5.7. Another benefit of using the DMP based encoding is that it also considers the current position along with the phase signal when doing motion reproduction. The starting value of phase signal can easily be inferred by linearly generating samples of phase signal in between 0 and 2π and then calculating the acceleration value $\dot{v}$ for each of them by using the DMP equation. The sample which yields the lowest value of sum of absolute accelerations of all DMPs is used as the starting point for integrating the canonical system.

5.5.2. Experiment 2: Different initial and final conditions

In the second experiment, an operator was asked to complete four counter-clockwise cycles of the peg-in-hole task, each time starting and ending above a different hole, as illustrated in Figures (5.9a-5.9d). A major constraint when applying DTW is that it assumes similar starting and ending positions for a signal, with temporal variations in between. If this condition is violated, its performance degrades. The major advantage with the proposed approach is that it does not need to fulfill any such constraints. Since the holes are symmetric, a user can easily lose track of a single hole while demonstrating the peg-in-hole task in case of the change in camera view.

Figure 5.9e shows the result of DTW and GMM-GMR approach. The learned trajectory was only able to reach two holes due to the extreme misalignment of phase signal. Figure 5.9f shows the result of DMP encoding with DTW based alignment of forcing terms. This approach only reaches one hole and fails to learn the complete task due to the aforementioned reasons. The presented approach successfully executes the task as shown in Figure 5.9g, because in this approach the trajectories alignment is performed during the EM.

5.5.3. Experiment 3: Incomplete demonstrations

This experiment utilizes data from the first experiment. One full trajectory is used while the others are clipped to simulate the partial executions. The generated trajectories are shown in Figures (5.10a-5.10d). This experiment depicts the situation where a human operator may fail to provide a full execution of the motion and abort the execution before completing the full cycle. As shown in Figure 5.10e DTW aligned GMM-GMR based reproduction generates reaching motion for two out of four holes but did not go deep enough to insert the peg. DMP-GMM based encoding with DTW based alignment of forcing terms completely fails to execute the task as depicted in Figure 5.10f. Again the presented approach successfully reproduces the learned task as shown in Figure 5.10g. The success rates of the three experiments are summarized in Table 5.2.

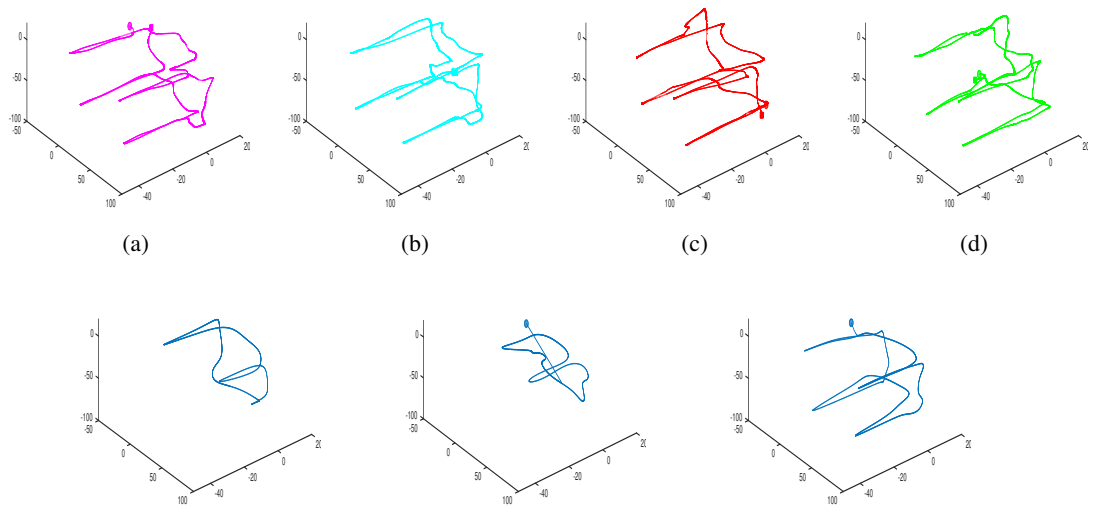


Figure 5.9.: All axis are in millimeters. (a-d) Recorded motions for experiment 2. The motions start and end above different holes in each demonstration. (e-g) Motion reproductions with different models (e) DTW+GMM based encoding of spatial data with GMR based motion reproduction, (f) DMP model with DTW+GMM based encoding of forcing terms, (g) Our approach.

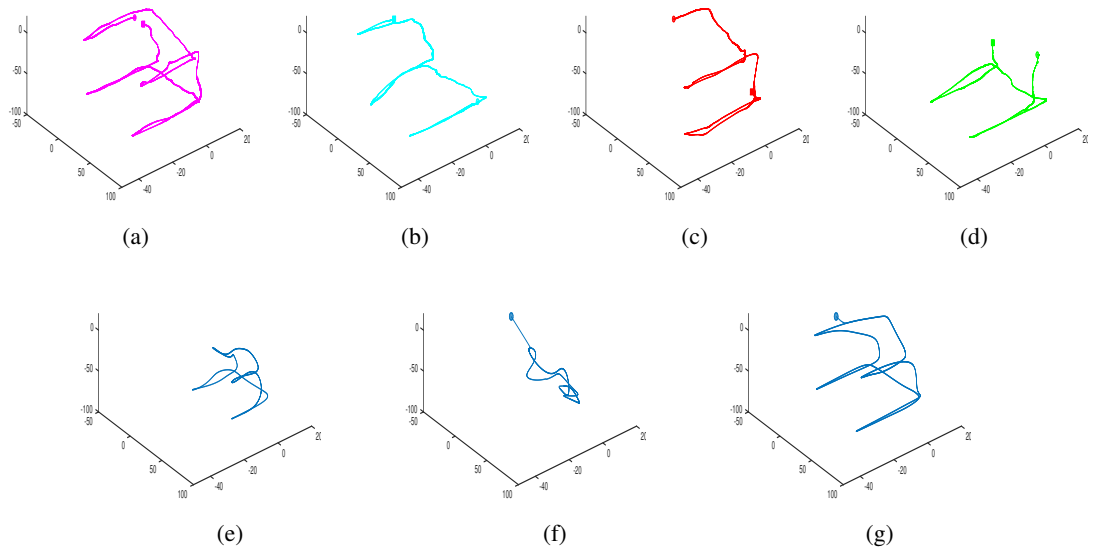


Figure 5.10.: All axis are in millimeters. (a-d) Demonstrations for experiment 3. (e-g) Motion reproductions with different models (e) DTW+GMM based encoding of spatial data with GMR based motion reproduction, (f) DMP model with DTW+GMM based encoding of forcing terms, (g) Our approach.

Table 5.2.: Success rate of different approaches shown as the number of holes reached.

	DTW GMM+GMR	DTW DMP+GMM	Our approach
Experiment 1	3	0	4
Experiment 2	2	1	4
Experiment 3	0	0	4

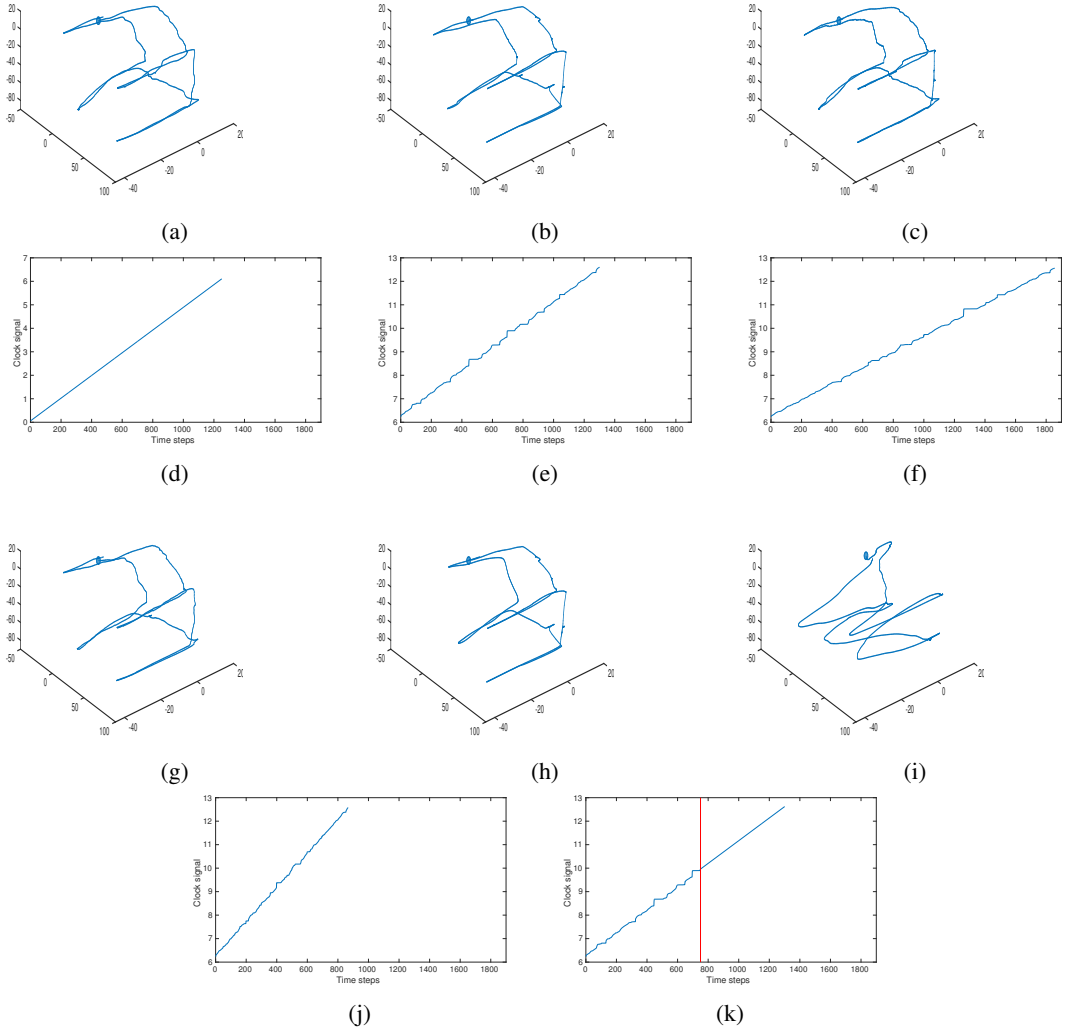


Figure 5.11.: **(a)** Recorded motion. **(b)** y-axis motion of the recorded motion is used as the operator motion while the motion of x and z-axis is autonomously generated. **(c)** x and y-axis motion of the recorded motion is used as the operator motion while the motion of z-axis is autonomously generated. The speed of operators motion is significantly slower than the demonstrated motions in this experiment. **(d)** Linearly increasing clock signal (without shared teleoperation) corresponding to experiment 1. **(e)** Generated clock signal corresponding to (b). **(f)** Generated clock signal corresponding to (c). **(g)** y and z-axis motion of the recorded motion is used as the operator motion while the motion of x-axis is autonomously generated. The speed of operators motion is significantly faster than the demonstrated motions in this experiment. **(h)** y-axis motion of the recorded motion is used as the operator motion while the motion of x and z-axis is autonomously generated during first half of the motion. The later half of the motion is generated in fully autonomous mode. **(i)** Control sharing without synchronization. The motion of x and y-axis comes from a human operator while that of z-axis is autonomously generated by the approach presented in Section 5.3.2. **(j)** Generated clock signal corresponding to (g). **(k)** Generated clock signal corresponding to (h). The switching point from shared teleoperation to full autonomy is indicated by the red line

5.5.4. Experiment 4: Shared teleoperation

Simulation Results of Agent-Synchronized Shared Teleoperation

For simulation based experiment, an operator is asked to demonstrate a single motion of the peg-in-hole task as a reference trajectory, as shown in Figure 5.11a. Now the reference trajectory is used along with the GMM model learned in experiment one to demonstrate the performance of the agent-synchronized shared teleoperation. For the simulation, the values of N, N_2 δ and Δ are set to 4000, 1300, 2×10^{-4} and 1×10^{-3} respectively. Firstly only the motion of y-axis is passed as the human input while that of x and z-axis are autonomously generated. The learned model successfully completes the peg-in-hole task as shown by the generated motion in Figure 5.11b while the evolution of clock signal is shown in Figure 5.11e.

Now the performance is evaluated for the temporal variation in the execution of the motion as well as for the control of multiple DOFs by the operator. The multiple degrees of freedom can be operated by a single operator or by more than one operator. Figure 5.11c shows the result of successful motion execution when x and y-axis of the motion comes from the reference trajectory as human input while the motion of z-axis is autonomously generated. The speed of execution of the motion of DOFs controlled by the reference trajectory is significantly decreased in this experiment and the generated clock signal also increases slowly. This can be observed by comparing Figure 5.11f with Figure 5.11e. Figure 5.11g shows the result of successful motion execution when the motion of y and z-axis comes from the reference trajectory while the motion of x-axis is autonomously executed. Now the speed of execution of the motion of DOFs controlled by the reference trajectory is significantly increased in this experiment. This generated clock signal adapts by increasing faster. This can be observed by comparing Figure 5.11j with Figure 5.11e. In all of the experiments of shared teleoperation, the generated clock signal does not simply increase linearly and small adjustments can be observed in the generated clock values for synchronizing the motion of autonomous DOFs with the DOFs controlled by the human. For comparison, Figure 5.11d show the result of generated clock signal for experiment 1, which only increases linearly, as the motion of all DOFs is autonomously generated.

The generated motion can also switch from shared teleoperation to fully autonomous mode and vice versa. If an operator wants to rest or is confident of correct motion execution then he can enable the fully autonomous mode. Figure 5.11h shows the result when half of the motion is generated in shared teleoperation setting while the second half of the motion is generated in fully autonomous mode. For full autonomy the increase of clock signal becomes linear, as shown in Figure 5.11k. Due to the use of a dynamical system in the DMP model, the control transitions smoothly from shared teleoperation to fully autonomous mode as shown in Figure 5.11h. Figure 5.11i shows the result of failed execution of the peg-in-hole task when the control is shared without synchronization in-between a learned model and an operator. For unsynchronized trajectory the motion of x and y-axis comes from the reference trajectory while the motion of z-axis is autonomously executed

Experimental Evaluation of Shared Teleoperation

Eight engineering students, both male and female, ages ranging from 23 to 33, participated in performing 4 trials for each of the three types of experiments - human-only teleoperation, human-synchronized shared teleoperation and agent-synchronized shared teleoperation. The subjects were given sufficient time to familiarize themselves with the setup of the three type of experiments. The subjects observe the visual feedback from the camera placed at the slave's end. During shared teleoperation experiments the agent and the operator have complete control authority of

their respective axis. In order to evaluate the performance of the two proposed shared teleoperation approaches against the human-only teleoperation mode, we recorded the execution time, the rate of collision and the overall workload index using NASA-TLX [38] for the three aforementioned settings.

Execution Time: Each execution cycle of the peg-in-hole task comprised of moving to the four holes of the rig and inserting the end-effector into them. Of the three teleoperation architectures, the human-synchronized shared teleoperation on average enables slightly faster motion execution, while the agent-synchronized shared teleoperation on average enables significantly faster motion execution, when compared with human-only teleoperation. Moreover, in the human-synchronized teleoperation, the execution time, measured in milliseconds, was almost constant throughout the task execution across all the 8 subjects' records, with only minor variations due to the noise introduced by the subjects (time taken by subjects to stop the recording manually). This is because the execution time in this setting is mainly governed by the time taken by the artificial agent to complete the task. As visible in Figure 5.12, the agent-synchronized shared control, with the same value for the temporal scaling factor τ, has the shortest execution time to complete one cycle of the task, due to the prediction of the next clock signal being based on the current motion of the human operator. This allows the execution speeds to be variable for different subjects, and hence, brisked up at points where human intended on moving faster, whereas slowed down (less increment in the clock value) where human wanted to take more time at critical points of the task. Finally, the human-only teleoperation posed the most difficulty and labour for the subjects, primarily due to the visual distortion and zero haptic assistance from any external agents, thereby making it the most slowest of the three teleoperation modes for the same task.

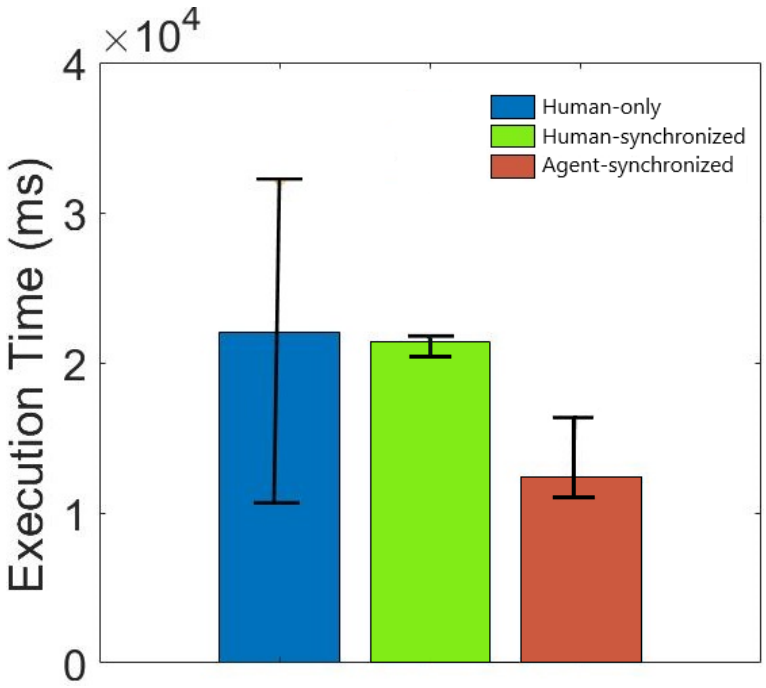


Figure 5.12.: Average execution time of the peg-in-hole task for the two shared teleoperation approaches compared with the human-only teleoperation experiments. Error bars indicate the minimum and maximum values.

Rate of Collision and Missed Holes: In this matric the three teleoperation approaches were evaluated based on the rate of collisions and the missed holes recorded while experimenting. By definition, 'collision' here means involuntarily hitting the rig with the slave's end-effector anywhere apart from the hole itself (including the walls surrounding the holes). It also incorporates dismantling the rig completely due to a delayed or brisk move of the human controlled axis. 'Missing' simply symbolizes losing the opportunity to insert the slave end-effector into the hole of the rig as per the task's requisite steps. Additionally, the maximum number of collisions incurred by each subject throughout the four trials of each teleoperation setting was averaged across all subjects. Figure 5.13 clearly shows that the subjects had much less collisions and missed holes with the task rig in the two shared teleoperation approaches, as compared to the highly error-prone human-only teleoperation. The reason being that in the human-synchronized control, the slave robot would traverse through the rig as per the prescribed steps of motion and would only station itself right above each hole, which gives the operator the time as well as the ease to just insert the end-effector when it is at that point. Similarly, the agent-synchronized shared teleoperation had no missed holes, since the slave only moved to the next step based on the master's current position, thereby synchronizing its speed with the human's speed at every instant. Whereas, the human-only teleoperation incurred the most collisions and holes' misses.

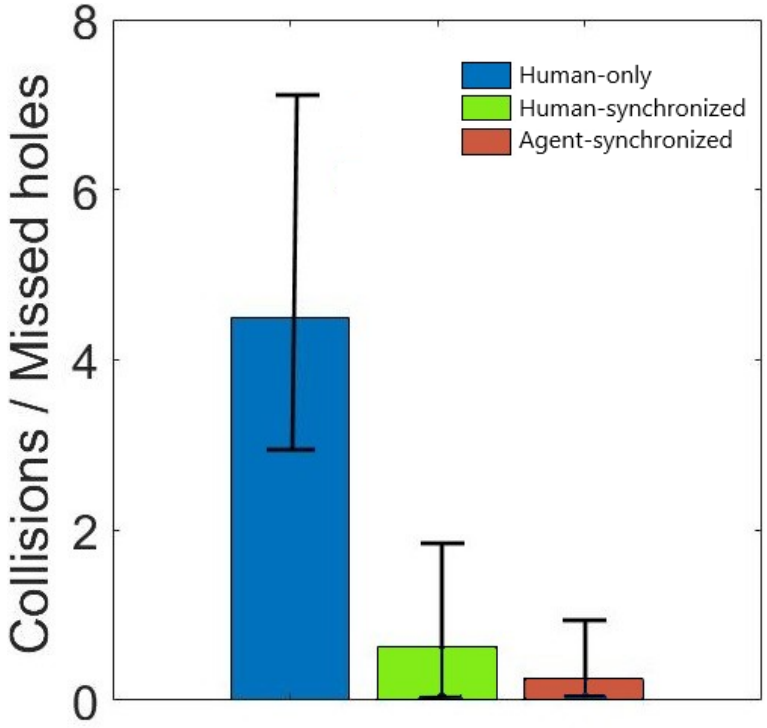


Figure 5.13.: Average rate of collision/missed holes of the peg-in-hole task for the two shared teleoperation approaches compared with the human-only teleoperation experiments. Error bars indicate the minimum and maximum values.

Workload Index: The NASA-TLX subjective assessment in terms of the overall workload is shown in Figure 5.14 for each of the teleoperation methods. As a consequence of all the satisfactory results in the previously mentioned metrics, both of the proposed shared teleoperation approaches have ranked subjectively to have a lower index of workload. The subjects reported to have felt complete ease at having to only care about inserting the end-effector into the holes

(y-axis motion) while the agent controlled the rest of the motion (x and z axes). Between the two shared teleoperation settings, the human-synchronized one takes preeminence over its agent-synchronized counterpart slightly. This precedence, even though by a slight margin, could perhaps be accounted for due to an obvious factor. The human subjects had their motion completely decoupled with that of the agent in the human-synchronized mode, hence had little to no dependency upon themselves regarding the agent's motion whilst they inserted the end-effector into the holes. Whereas, in the agent-synchronized mode, the human users had to be constantly in some minimal movement in order for the agent to deduce that the human motion was in fact non-zero, and thus the clock signal needs to jump to the next step in the task trajectory. Having only the task execution fed back into the agent for real-time reproduction as the only distinguishing feature of the agent-synchronized teleoperation, the human-agent shared teleoperation served as the least laborious of the two of them.

Hence, conclusively, the agent-synchronized based teleoperation mode was evaluated to be the one with the most accurate execution of task while taking the least bit of time to perform it across all subjects. Whereas, its sibling teleoperation, human-synchronized shared control, came out to be the least burden-some of all.

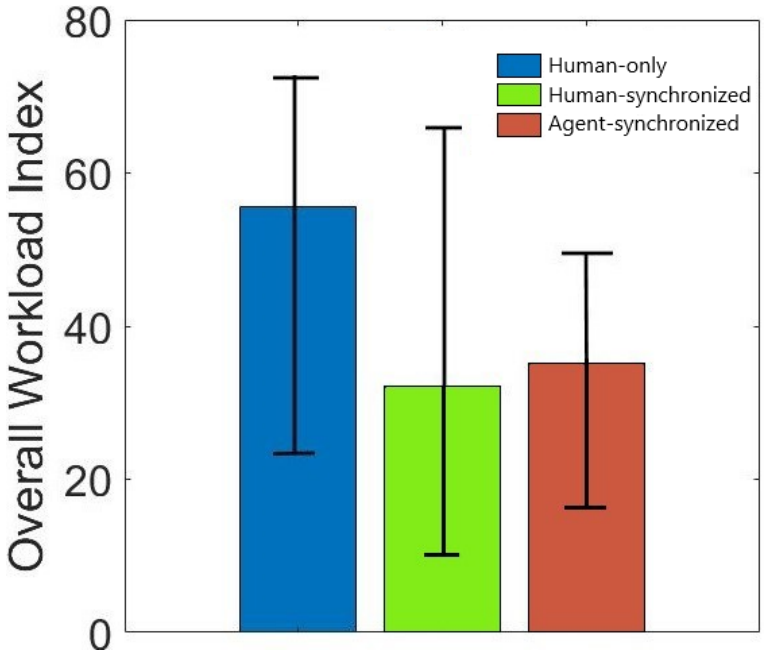


Figure 5.14.: Overall workload index of the peg-in-hole task for the two shared teleoperation approaches compared with the human-only teleoperation experiments. Error bars indicate the minimum and maximum values.

5.6. Discussion

This work introduces a way to simultaneously align and encode the teleopearated demonstrations via the EM algorithm. The proposed approach outperforms the approaches which first perform DTW based pre-alignment before encoding the trajectories. The reason for the failure of the other approaches is that the DTW assumes the signals/trajectories to have same initial and final configurations. If this condition is violated then the DTW cannot align the signal. This is particularly true in the second and the third experiments where demonstrations with different initial and final

conditions and incomplete demonstrations are considered. Due to this reason DTW failed to align those demonstrations.

The experiments show the advantage of using the EM based alignment and encoding approach. Although the presented approach align the data before encoding it, it cannot be separately used as a trajectories alignment algorithm. This is due to the fact that it does not guarantee to preserve the temporal structure of the trajectories, as each data point is considered independently in EM. To use EM for only aligning the trajectories, additional temporal constraints should be considered in the EM cycles. This can also be regarded as one of the future working directions for this work.

In the shared teleopeartion settings, human and the learned controller operate in conjunction. One can argue that if the model can operate autonomously then what is the point of including an operator in the loop. The idea behind shared teleopeartion setup is to execute tasks where some DOFs have repetitive motion while some have non-repetitive motions. It also guarantee a safe task execution. A wrong task execution by the learned controller can be dangerous for the robot as well as for the environment. If the human feels that the task is being correctly executed then he can let the learned model to execute autonomously. If the operator feels that the certain DOFs should be manually controlled then they can be taken over by him. As we showed in the experiments, the switching inbetween manual control and the autonomous mode can take place smoothly at any moment. Hence the degree of autonomy can be easily regulated by the operator.

5.7. Summary and Future works

This chapter discusses the typical problems encountered when applying existing PbD approaches to teleoperation. Although it explicitly focuses on utilizing teleopearated demonstrations, this does not limit the applicability of the presented approach to demonstrations collected via kinesthetic teaching. In this chapter an EM based approach for aligning and encoding the trajectories simultaneously is presented. This is achieved by simultaneous estimation of the GMM parameters and the phase variables. Although the results have been presented for a rhythmic peg-in-hole task, the approach can encode rhythmic as well as discrete motions, as it possesses all the desirable properties associated with a DMP model. Through multiple peg-in-hole experiments, it is demonstrated that the presented approach can handle large variabilities in the teleoperation demonstrations i.e. large spatial and temporal variabilities, demonstrations with different starting/ending phases and partial task executions. Additionally it is also shown that the learned model can also be used in the shared autonomy setup. This way the motion of DOFs which are delicate for the execution of the task can be controlled by the human operator/operators while the motion of the remaining DOFs can be autonomously executed. Adding human in the loop also ensures correct task execution and supervision during motion execution. The proposed method outperforms existing approaches based on combining DTW based time alignment and trajectories encoding algorithms.

Generally a separate DMP is learned for each considered DOF. Since the tasks are repetitive, as a future work it can be useful to consider the learning of correlations inbetween different degrees of freedom. This can be done by learning only a single DMP that can capture the correlations inbetween all DOF, instead of learning a separate DMP for each DOF. Learning of movement primitives by utilizing cameara images is considered in the last chapter. Since the camera feedback is also available in the teleopeartion setup, use of deep architectures such as a CNN can also be investigated for learning repetitive teleoperation tasks. This work only utilizes the position data for encoding the tasks. Haptic data is also often available in the teleoperation setups [1,71,84,89]. For the tasks which involve interaction with the environment, force feedback can be also encoded in the learned DMP for robust task execution. The EM approach presented in this chapter can be combined with the EM algorithm for task parameterized skill learning presented in chapter 3. This

can syncronize the phase variable, along with encoding of the task variables of the demonstrations, yielding better task reproductions.

CHAPTER 6

Conclusion

6.1. Conclusions

PbD is the main focus of the reserach presented in this thesis. More specifically I have presented PbD algorithms for robotics tasks. Although it is not a new research area in robotics, there are still many challenges which are associated with the existing PbD approaches. In this work, some of those challenges are identified and new PbD approaches are presented to address them.

In chapter 2, a novel approach for handling the problems encountered when fitting a GMM with EM is proposed, as most of the PbD approaches presented in this thesis require density estimation via GMM as a preliminary step. The proposed method considers the initialization and the model complexity issues. The initialization approach for fitting a GMM utilizes a search strategy where each component looks for its nearby component. Two components are combined when they have a high overlap. This provides a transition from an irregular density function to a smoother density function. CSAEM^2 is applied when the components count reaches k_{max}. A component is annihilated if its prior becomes very small. $(k-1)$ components GMM is obtained by selecting the one which yields highest likelihood value. A model selection criterion is utilized to select the optimal model complexity.

In chapter 3, it is shown that how the task specific generalization of a DMP can be achieved by formulating learning as a density estimation problem. The proposed approach captures the local behavior of each demonstration by using a GMM. These GMMs are then mixed to get the task specific generalization. The data sparsity along task parameters is handled by introducing additional incomplete data filling the input space. The task specific generalization is achieved by just adapting covariances and mixing coefficient of the already learned GMMs. The TP-DMP framework can perform learning in task space as well as in joint space and can also handle the learning of meta parameters of a DMP. The proposed approach requires very few demonstrations for learning and it outperforms the existing approaches specially when extrapolating beyond the demonstrated ranges of the task variables.

Chapter 4 introduces **D-DMP**, which is learned by modeling the forcing terms of a TP-DMP with a CNN. In **D-DMP** the task parameters are only required during the training phase. After learning, the camera image is directly used for motion reproduction. Since the features are learned, this eliminates the need to use any markers. Additionally by applying the data augmentation, **D-DMP** can easily generalize the learned skill for various different objects. The generalization

capability of is demonstrated by executing the task for a novel object and against the background perturbations. **D-DMP** requires relatively fewer training data for learning, as compared with other CNN based motor learning approaches [62, 63, 112].

Chapter 5 highlights the typical problems encountered when applying existing PbD approaches to teleoperation, as it is not always possible to utilize kinesthetic teaching for collecting the demonstrations. For solving the issues with teleoperated demonstration, an EM based approach for aligning and encoding the trajectories simultaneously is presented. This is achieved by simultaneous estimation of the GMM parameters and the phase variables. Additionally it is also shown that the learned model can also be used in the shared autonomy setup. This way the motion of DOFs which are delicate for the execution of the task can be controlled by the human operator/operators while the motion of the remaining DOFs can be autonomously executed. The proposed method outperforms existing approaches based on combining DTW based time alignment and trajectories encoding algorithms.

6.2. Future research directions

Most of the research presented in this thesis is based on developing mathematical models suitable for PbD. More research need to be carried out from a psychological perspective on about how humans actually perform imitation learning. As a future work for TP-DMP and **D-DMP**, the proposed approaches can be extended with sample reuse approach [27], where the reproduced motion is regarded as an additional demonstration for improving the performance of the task. Another more popular approach of skill refinement is reinforcement learning. Directly applying RL for learning a motor skill can require a lot of trials [63]. PbD is mostly utilized to initialize the model parameters of a movement primitive. RL is then usually applied as a next step to improve the performance of the learned model. As a future work for the PbD approaches presented in this thesis, they can be used to initialize the model parameters and for rapidly acquiring a motor skill, while the skill refinement can subsequently be done by using RL [98].

For **D-DMP**, the task parameters still have to be recorded during the data collection phase. As a future work, by utilizing the unsupervised learning methods, for instance an autoencoder, lower dimensional features from a camera image can be directly extracted. The extracted features can then be utilized for learning the motor actions, eliminating the need to utilize the task parameters for initializing the CNN parameters. The presented **D-DMP** approach utilizes a CNN. Since the robotics tasks usually have a temporal structure, utilizing a Recurrent Neural Network to capture those dependencies can be considered. The visual feedback is always available in the teleoperation setups. Using deep learning approaches for directly processing the visual feedback in the teleoperation scenarios can be investigated. Also for teleopeartion, additional information, for instance the haptic feedback, can be utilized in the presented skill acquisition approach.

Appendices

APPENDIX A

Appendix

Following properties have been used in this proof [80]:

- *Property 1:* If $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{b}$ a vector then $E\left[(\mathbf{X}+\mathbf{b})(\mathbf{X}+\mathbf{b})^\top\right] = \boldsymbol{\Sigma} + (\boldsymbol{\mu}+\mathbf{b})(\boldsymbol{\mu}+\mathbf{b})^\top$

- *Property 2:* The product of Gaussian densities is defined as: $\mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})\mathcal{N}(\boldsymbol{\mu_2}, \boldsymbol{\Sigma_2}) = e\ \mathcal{N}(\boldsymbol{\mu}_e, \boldsymbol{\Sigma}_e)$ where $e = \mathcal{N}(\boldsymbol{\mu_1}; \boldsymbol{\mu_2}, \boldsymbol{\Sigma_1} + \boldsymbol{\Sigma_2})$.

Expected pdf value:
The Expected pdf value of a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ evaluated at a Gaussian stochastic random variable $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ can be defined as:
$E[\mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})] \rightarrow E\left[\mathcal{N}(\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X); \boldsymbol{\mu}, \boldsymbol{\Sigma})\right]$ which is similar to $E[f(\boldsymbol{X})] = \int f(x)p(x)dx$. Now

$$\begin{aligned} E\left[\mathcal{N}\left(\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X); \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\right] &= \\ \int_{-\infty}^{\infty} \ \dots \ & \int_{-\infty}^{\infty} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X) \end{aligned}$$

By using property 2.

$$\begin{aligned} E\left[\mathcal{N}\left(\mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X); \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)\right] &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} e\mathcal{N}(x; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\ &= e\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathcal{N}(x; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\ &= e \end{aligned}$$

where

$$e = \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\mu}_X, \boldsymbol{\Sigma} + \boldsymbol{\Sigma}_X) \tag{A.1}$$

Incorporating Incomplete data via EM for mixture of GMMs:
In the E-step, the expectations $E[z_{i,k}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$, $E[z_{i,k}, \mathbf{x}^{miss}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$ and $E[z_{i,k}, \mathbf{x}^{miss}\mathbf{x}^{miss^\top}|\mathbf{x}^{obs}, \boldsymbol{\theta}_t]$ have to be calculated for incomplete data [36]. Let $\boldsymbol{\mu}_{o,m} = \begin{bmatrix} \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}$, $\boldsymbol{\Sigma}_{o,m} = \begin{bmatrix} \boldsymbol{\Sigma}_{o,m}^{obs} & \boldsymbol{\Sigma}_{o,m}^{obs.miss} \\ \boldsymbol{\Sigma}_{o,m}^{miss.obs} & \boldsymbol{\Sigma}_{o,m}^{miss} \end{bmatrix}$,

$\boldsymbol{X}_w = \begin{bmatrix} \boldsymbol{X}_w^{obs} \\ \boldsymbol{X}_w^{miss} \end{bmatrix}$ where $\boldsymbol{X}_w^{obs} \sim \mathcal{N}(\boldsymbol{\mu}_w^{obs}, \boldsymbol{\Sigma}_w^{obs})$ is the observable part representing the GMM components spanning the input space. Now the similar expectations of the missing dimensions (output) are calculated as:

$$\begin{aligned} E[z_{w,o,m}|\boldsymbol{X}_w^{obs}, \boldsymbol{\theta}_t] &= d_{w,o,m}^{t+1} \\ &= \frac{c_{w,o,m}^{t+1}}{\sum_{r=1}^{M}\sum_{l=1}^{K} c_{w,r,l}^{t+1}} \times \boldsymbol{\pi}_w^{obs} \times N \end{aligned}$$

with $c_{w,o,m}^{t+1} = \left(\phi_o \boldsymbol{\pi}_{o,m} \mathcal{N}(\boldsymbol{\mu}_w^{obs}; \boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_w^{obs} + \boldsymbol{\Sigma}_{o,m}^{obs})\right)^{\beta}$ where $\mathcal{N}(\boldsymbol{\mu}_w^{obs}; \boldsymbol{\mu}_{o,m}^{obs}, \boldsymbol{\Sigma}_w^{obs} + \boldsymbol{\Sigma}_{o,m}^{obs})$ is the expected pdf calculated only on the observed dimensions, as derived for Equation (A.1).

$$E[z_{w,o,m}, \boldsymbol{X}_w^{miss}|\boldsymbol{X}_w^{obs}, \boldsymbol{\theta}_t] = d_{w,o,m}^{t+1}(\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{d,m}^{obs}))$$

Define $\hat{\boldsymbol{X}}_{w,o,m}^{miss} = \boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})$
Now:

$$E[z_{w,o,m}, \boldsymbol{X}_w^{miss}|\boldsymbol{X}_w^{obs}, \boldsymbol{\theta}_t] = d_{w,o,m}^{t+1}\hat{\boldsymbol{X}}_{w,o,m}^{miss}$$
$$E[z_{w,o,m}, \boldsymbol{X}_w^{miss}\boldsymbol{X}_w^{miss^\top}|\boldsymbol{X}_w^{obs}, \boldsymbol{\theta}_t] = d_{w,o,m}^{t+1}(\boldsymbol{\Sigma}_{o,m}^{miss} - \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top + \hat{\boldsymbol{X}}_{w,o,m}^{miss}\hat{\boldsymbol{X}}_{w,o,m}^{miss^\top})$$

For updating $\boldsymbol{\Sigma}_{o,m}^{t+1}$, the term $d_{w,o,m}^{t+1}(\boldsymbol{X}_w - \boldsymbol{\mu}_{o,m})(\boldsymbol{X}_w - \boldsymbol{\mu}_{o,m})^\top$ for incomplete data is calculated as:

$$d_{w,o,m}^{t+1}\begin{bmatrix} \boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}\begin{bmatrix} \boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs} \\ \boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss} \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix}$$

where

$$\begin{aligned} \boldsymbol{A}_{11} &= d_{w,o,m}^{t+1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\ &\quad \text{by using Property 1} \\ &= d_{w,o,m}^{t+1}\left(\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top\right) \end{aligned}$$

$$\begin{aligned} \boldsymbol{A}_{21} &= d_{w,o,m}^{t+1}(\boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss})(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\ &= (E[z_{w,o,m}, \boldsymbol{X}_w^{miss}] - d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss})(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\ &= \left(d_{w,o,m}^{t+1}(\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})) - d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\right)(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\ &= d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top \\ &= d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}\left(\boldsymbol{\Sigma}_w^{obs} + (\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})(\boldsymbol{\mu}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^\top\right) \\ \boldsymbol{A}_{12} &= \boldsymbol{A}_{21}^\top \end{aligned}$$

$$\begin{aligned} \boldsymbol{A}_{22} &= d_{w,o,m}^{t+1}(\boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss})(\boldsymbol{X}_w^{miss} - \boldsymbol{\mu}_{o,m}^{miss})^\top \\ &= E[z_{w,o,m}, \boldsymbol{X}_w^{miss}\boldsymbol{X}_w^{miss^\top}] + d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}(\boldsymbol{\mu}_{o,m}^{miss})^\top - 2E[z_{w,o,m}, \boldsymbol{X}_w^{miss}](\boldsymbol{\mu}_{o,m}^{miss})^\top \\ &= d_{w,o,m}^{t+1}(\boldsymbol{\Sigma}_{o,m}^{miss} - \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{\Sigma}_{o,m}^{miss.obs})^\top + \hat{\boldsymbol{X}}_{w,o,m}^{miss}\hat{\boldsymbol{X}}_{w,o,m}^{miss^\top}) + d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}(\boldsymbol{\mu}_{o,m}^{miss})^\top \\ &\quad -2\left(d_{w,o,m}^{t+1}(\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}))\right)(\boldsymbol{\mu}_{o,m}^{miss})^\top \end{aligned}$$

$$
\begin{aligned}
\boldsymbol{A}_{22} &= d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss} - d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{\Sigma}_{o,m}^{miss.obs})^{\top} + d_{w,o,m}^{t+1}\big(\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})\big) \\
&\quad \big(\boldsymbol{\mu}_{o,m}^{miss} + \boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})\big)^{\top} + d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\boldsymbol{\mu}_{o,m}^{miss\top} - 2d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\boldsymbol{\mu}_{o,m}^{miss\top} \\
&\quad -2d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})\boldsymbol{\mu}_{o,m}^{miss\top} \\
&= d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss} - d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{\Sigma}_{o,m}^{miss.obs})^{\top} + \cancel{d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\boldsymbol{\mu}_{o,m}^{miss\top}} \\
&\quad \cancel{+2d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\big(\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})\big)^{\top}} + d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) \\
&\quad (\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^{\top}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}\boldsymbol{\Sigma}_{o,m}^{miss.obs\top} + \cancel{s_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\boldsymbol{\mu}_{o,m}^{miss\top}} - \cancel{2d_{w,o,m}^{t+1}\boldsymbol{\mu}_{o,m}^{miss}\boldsymbol{\mu}_{o,m}^{miss\top}} \\
&\quad \cancel{-2d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{X}_w^{obs} - \boldsymbol{\mu}_{o,m}^{obs})\boldsymbol{\mu}_{o,m}^{miss\top}} \\
&= d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss} - d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}(\boldsymbol{\Sigma}_{o,m}^{miss.obs})^{\top} + d_{w,o,m}^{t+1}\boldsymbol{\Sigma}_{o,m}^{miss.obs}(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}\Big(\boldsymbol{\Sigma}_{w}^{obs} + (\boldsymbol{\mu}_{w}^{obs} - \boldsymbol{\mu}_{o,m}^{obs}) \\
&\quad (\boldsymbol{\mu}_{w}^{obs} - \boldsymbol{\mu}_{o,m}^{obs})^{\top}\Big)(\boldsymbol{\Sigma}_{o,m}^{obs})^{-1}\boldsymbol{\Sigma}_{o,m}^{miss.obs\top}
\end{aligned}
$$

Bibliography

[1] David A Abbink, Mark Mulder, and Erwin R Boer. Haptic shared control: smoothly shifting control authority? *Cognition, Technology & Work*, 14(1):19–28, 2012.

[2] Jake Abbott, Panadda Marayong, and Allison Okamura. Haptic virtual fixtures for robot-assisted manipulation. *Robotics research*, pages 49–64, 2007.

[3] B. Akgun, Kaushik Subramanian, and A.L. Thomaz. Novel interaction strategies for learning from teleoperation. *AAAI Fall Symposium Series*, pages 2–9, 2012.

[4] Baris Akgun and Kaushik Subramaman. Robot Learning from Demonstration: Kinesthetic Teaching vs. Teleoperation. page 7, 2011.

[5] T. Alizadeh, S. Calinon, and D. G. Caldwell. Learning from demonstrations with partially observable task parameters. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3309–3314, Hong Kong, China, May-June 2014.

[6] Tohid Alizadeh. *Statistical Learning of Task Modulated Human Movements Through Demonstration*. PhD thesis, Istituto Italiano di Tecnologia, 2014.

[7] Edgar Anderson. The irises of the gaspe peninsula. *Bulletin of the American Iris society*, 59:2–5, 1935.

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

[9] Halima Bensmail, Gilles Celeux, Adrian E Raftery, and Christian P Robert. Inference in model-based cluster analysis. *Statistics and Computing*, 7(1):1–10, 1997.

[10] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. 2008.

[11] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[12] Christopher M Bishop, Julia Lasserre, et al. Generative or discriminative? getting the best of both worlds. *Bayesian statistics*, 8:3–24, 2007.

[13] Konstantinos Blekas and Isaac E Lagaris. Split–merge incremental learning (smile) of mixture models. In *Artificial Neural Networks–ICANN 2007*, pages 291–300. Springer, 2007.

[14] Joseph Bukchin, Ruth Luquer, and Avraham Shtub. Learning in tele-operations. *IIE Transactions*, 34(3):245–252, 2002.

[15] S. Calinon, F. D'halluin, D.G. Caldwell, and A. Billard. Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, pages 582–588, December 2009.

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

[17] Sylvain Calinon, Tohid Alizadeh, and Darwin G Caldwell. On improving the extrapolation capability of task-parameterized movement models. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 610–616. IEEE, 2013.

[18] Sylvain Calinon, Paul Evrard, Elena Gribovskaya, Aude Billard, and Abderrahmane Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *IEEE Int. Conf. on Advanced Robotics*, pages 1–6, 2009.

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

[20] Sylvain Calinon and Dongheui Lee. Learning control. In *Humanoid Robotics: A Reference*. Springer Netherlands, 2017.

[21] Sylvain Calinon, Zhibin Li, Tohid Alizadeh, Nikos G Tsagarakis, and Darwin G Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 323–329.

[22] Sylvain Calinon, Affan Pervez, and Darwin G Caldwell. Multi-optima exploration with adaptive gaussian mixture model. In *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6, 2012.

[23] Gilles Celeux, Stéphane Chrétien, Florence Forbes, and Abdallah Mkhadri. A component-wise em algorithm for mixtures. *Journal of Computational and Graphical Statistics*, 2012.

[24] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3381–3388, 2017.

[25] Jie Yang'Yangsheng Xu CS ChenZ. Hidden markov model approach to skill learning and its application to telerobotics. 1993.

[26] Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jurgen Schmidhuber. Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139. IEEE, 2011.

[27] Bruno Castro Da Silva, Gianluca Baldassarre, George Konidaris, and Andrew Barto. Learning parameterized motor skills on a humanoid robot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5239–5244. IEEE, 2014.

[28] Arthur P Dempster, Nan M Laird, Donald B Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[29] Anca D Dragan and Siddhartha S Srinivasa. Assistive teleoperation for manipulation tasks. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 123–124. ACM, 2012.

[30] Málrio AT Figueiredo, José MN Leitão, and Anil K Jain. On fitting mixture models. In *Energy minimization methods in computer vision and pattern recognition*, pages 54–69. Springer, 1999.

[31] Mario AT Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396, 2002.

[32] Kerstin Fischer, Franziska Kirstein, Lars Christian Jensen, Norbert Kr, Kamil Kukli, Maria aus der Wieschen, and Thiusius Savarimuthu. A comparison of types of robot control for programming by demonstration. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 213–220, 2016.

[33] Denis Forte, Andrej Gams, Jun Morimoto, and Aleš Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):1327–1339, 2012.

[34] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

[35] Andrej Gams, Auke J Ijspeert, Stefan Schaal, and Jadran Lenarčič. On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous robots*, 27(1):3–23, 2009.

[36] Zoubin Ghahramani and Michael I Jordan. Supervised learning from incomplete data via an em approach. In *Advances in neural information processing systems 6*. Citeseer, 1994.

[37] Jacob Goldberger and Sam T Roweis. Hierarchical clustering of a mixture model. In *NIPS*, 2004.

[38] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[39] Timothy C Havens, Christopher J Spain, Nathan G Salmon, and James M Keller. Roach infestation optimization. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pages 1–7. IEEE, 2008.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Dennis Herzog, Volker Krüger, and Daniel Grest. Parametric hidden markov models for recognition and synthesis of movements. In *BMVC*, volume 8, pages 163–172. Citeseer, 2008.

[42] Peter F Hokayem and Mark W Spong. Bilateral teleoperation: An historical survey. *Automatica*, 42(12):2035–2057, 2006.

[43] Kai Hu, Christian Ott, and Dongheui Lee. Online human walking imitation in task and joint space based on quadratic programming. In *IEEE Int. Conf. on Robotics and Automation*, pages 3458–3464, 2014.

[44] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[45] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[46] D Kinga and J Ba Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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

[48] Jens Kober and Jan Peters. Imitation and reinforcement learning. *IEEE Robotics & Automation Magazine*, 17(2):55–62, 2010.

[49] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. Incremental object learning and robust tracking of multiple objects from rgb-d point set data. *Journal of Visual Communication and Image Representation*, 25(1):108–121, 2014.

[50] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3237, Taipei, Taiwan, October 2010.

[51] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

[52] Jens Krause and Graeme D Ruxton. *Living in groups*. Oxford University Press, 2002.

[53] Jonas Krause, Jelson Cordeiro, Rafael Stubs Parpinelli, and Heitor Silverio Lopes. A survey of swarm algorithms applied to discrete optimization problems. *Swarm Intelligence and Bio-inspired Computation: Theory and Applications. Elsevier Science & Technology Books*, pages 169–191, 2013.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[55] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[56] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[57] Dongheui Lee and Yoshihiko Nakamura. Mimesis model from partial observations for a humanoid robot. *The International Journal of Robotics Research*, 29(1):60–80, 2010.

[58] Dongheui Lee and Christian Ott. Incremental motion primitive learning by physical coaching using impedance control. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4133–4140, 2010.

[59] Dongheui Lee and Christian Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31(2):115–131, 2011.

[60] Dongheui Lee, Christian Ott, and Yoshihiko Nakamura. Mimetic communication with impedance control for physical human-robot interaction. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1535–1542. IEEE, 2009.

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

[62] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[63] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 2016.

[64] Bin Luo, Sui Wei, et al. Estimation for the number of components in a mixture model using stepwise split-and-merge em algorithm. *Pattern Recognition Letters*, 25(16):1799–1809, 2004.

[65] Takamitsu Matsubara, Sang-Ho Hyon, and Jun Morimoto. Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Networks*, 24(5):493–500, 2011.

[66] Clare A McGrory and DM Titterington. Variational approximations in bayesian model selection for finite mixture distributions. *Computational Statistics & Data Analysis*, 51(11):5352–5367, 2007.

[67] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[68] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[69] Christian Ott, Dongheui Lee, and Yoshihiko Nakamura. Motion capture based human motion recognition and imitation by direct marker control. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 399–405, 2008.

[70] Dae-Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 91–98. IEEE, 2008.

[71] Shinsuk Park, Robert D Howe, and David F Torchiana. Virtual fixtures for robotic cardiac surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 1419–1420. Springer, 2001.

[72] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, ICRA. IEEE International Conference on*, pages 763–768, 2009.

[73] Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, and Stefan Schaal. Towards associative skill memories. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 309–315. IEEE, 2012.

[74] Peter Pastor, Ludovic Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 365–371. IEEE, 2011.

[75] Carlos Jesús Pérez-del Pulgar, Jan Smisek, Victor F Muñoz, and André Schiele. Using learning from demonstration to generate real-time guidance for haptic shared control. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 003205–003210. IEEE, 2016.

[76] Affan Pervez, Arslan Ali, Jee-Hwan Ryu, and Dongheui Lee. Novel learning from demonstration approach for repetitive teleoperation tasks. In *IEEE World Haptics Conference (WHC), 2017*, pages 60–65.

[77] Affan Pervez and Dongheui Lee. A componentwise simulated annealing em algorithm for mixtures. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 287–294. Springer, 2015.

[78] Affan Pervez and Dongheui Lee. Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, pages 1–18, 2017.

[79] Richard Alan Peters, Christina L Campbell, William J Bluethmann, and Eric Huber. Robonaut task learning through teleoperation. In *IEEE Int. Conf. on Robotics and Automation*, pages 2806–2811, 2003.

[80] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. *Technical University of Denmark*, pages 7–15, 2008.

[81] Pedro Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning*, pages 82–90, 2014.

[82] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

[83] Maura Power, Hedyeh Rafii-Tari, Christos Bergeles, Valentina Vitiello, and Guang-Zhong Yang. A cooperative control framework for haptic guidance of bimanual surgical tasks based on learning from demonstration. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 5330–5337, 2015.

[84] Gennaro Raiola, Xavier Lamy, and Freek Stulp. Co-manipulation with multiple probabilistic virtual guides. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 7–13. IEEE, 2015.

[85] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

[86] Sylvia Richardson and Peter J Green. On bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: series B (statistical methodology)*, 59(4):731–792, 1997.

[87] Jorma Rissanen. *Stochastic complexity in statistical inquiry*, volume 511. World scientific Singapore, 1989.

[88] Kathryn Roeder and Larry Wasserman. Practical bayesian density estimation using mixtures of normals. *Journal of the American Statistical Association*, 92(439):894–902, 1997.

[89] Louis B Rosenberg. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pages 76–82. IEEE, 1993.

[90] L. Rozo, S. Calinon, and D. G. Caldwell. Learning force and position constraints in human-robot cooperative transportation. In *Proc. IEEE Intl Symposium on Robot and Human Interactive Communication (Ro-Man)*, pages 619–624, Edinburgh, Scotland, UK, 2014.

[91] Leonel Rozo, Sylvain Calinon, Darwin Caldwell, Pablo Jiménez, and Carme Torras. Learning collaborative impedance-based robot behaviors. *parameters*, 1(1):1, 2013.

[92] Leonel Rozo, Pablo Jiménez, and Carme Torras. A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent service robotics*, 6(1):33–51, 2013.

[93] Leonel Rozo, Pablo Jimenez Schlegl, and Carme Torras. Sharpening haptic inputs for teaching a manipulation skill to a robot. In *IEEE Int. Conf. on Applied Bionics and Biomechanics*, pages 331–340, 2010.

[94] Leonel Dario Rozo, Pablo Jiménez, and Carme Torras. Learning force-based robot skills from haptic demonstration. In *CCIA*, pages 331–340, 2010.

[95] Elmar Rueckert, Jan Mundo, Alexandros Paraschos, Jan Peters, and Gerhard Neumann. Extracting low-dimensional control variables for movement primitives. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1511–1518. IEEE, 2015.

[96] Parinya Sanguansat. Multiple multidimensional sequence alignment using generalized dynamic time warping. *WSEAS Transactions on Mathematics*, 11(8):668–678, 2012.

[97] Matteo Saveriano, Sang ik An, and Dongheui Lee. Incremental kinesthetic teaching of end-effector and null-space motion primitives. In *IEEE International Conference on Robotics and Automation (ICRA), 2015*, pages 3570–3575.

[98] Matteo Saveriano, Yuchao Yin, Pietro Falco, and Dongheui Lee. Learning control policies using a simplified robot model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[99] Stefan 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.

[100] Alexander M Schmidts, Dongheui Lee, and Angelika Peer. Imitation learning of human grasping skills from motion and force data. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1002–1007. IEEE, 2011.

[101] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[102] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.

[103] Freek Stulp, Gennaro Raiola, Antoine Hoarau, Serena Ivaldi, and Olivier Sigaud. Learning compact parameterized skills with a single regression. In *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 417–422, 2013.

[104] Ales Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *Robotics, IEEE Transactions on*, 26(5):800–815, 2010.

[105] Naonori Ueda and Ryohei Nakano. Deterministic annealing em algorithm. *Neural Networks*, 11(2):271–282, 1998.

[106] Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey E Hinton. Smem algorithm for mixture models. *Neural computation*, 12(9):2109–2128, 2000.

[107] Naveed Ahmed Usmani, Tae-Hwan Kim, and Jee-Hwan Ryu. Dynamic authority distribution for cooperative teleoperation. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5222–5227. IEEE, 2015.

[108] Fabio Valente, Christian Wellekens, et al. Variational bayesian gmm for speech recognition.

[109] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *International conference on machine learning, proceedings of the sixteenth conference*, 2000.

[110] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[111] Kazuho Watanabe and Sumio Watanabe. Stochastic complexities of gaussian mixtures in variational bayesian approximation. *The Journal of Machine Learning Research*, 7:625–644, 2006.

[112] Pin-Chu Yang, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. Repeatable folding task by humanoid robot worker using deep learning. *IEEE Robotics and Automation Letters*, 2(2):397–403, 2017.

[113] Baibo Zhang, Changshui Zhang, and Xing Yi. Competitive em algorithm for finite mixture models. *Pattern recognition*, 37(1):131–144, 2004.

[114] Zhihua Zhang, Chibiao Chen, Jian Sun, and Kap Luk Chan. Em algorithms for gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983, 2003.