EPFL

Thèse n° 7777

# Product of experts for robot learning from demonstration

Présentée le 8 avril 2021

Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

**Emmanuel PIGNAT**

Acceptée sur proposition du jury

Prof. P. Vandergheynst, président du jury
Prof. H. Bourlard, Dr S. Calinon, directeurs de thèse
Prof. L. Righetti, rapporteur
Prof. O. Sigaud, rapporteur
Dr R. Boulic, rapporteur

École polytechnique fédérale de Lausanne

2021

# Acknowledgments

First, I would like to thank Dr Sylvain Calinon, my supervisor, for providing guidance and feedback throughout this project. I am also grateful for the trust he had in me and the freedom he gave me to pursue my own research directions.

I wish to thank all my colleagues at Idiap for the numerous and various discussions which both enriched the content of my thesis and made the time spent working more pleasant.

I am much obliged to my thesis jury members who spend their precious time to read and help me improve my thesis: my advisor Sylvain, Prof. Hervé Bourlard, Dr. Ronan Boulic, Prof. Ludovic Righetti, and Prof. Olivier Sigaud.

# Résumé

L'adaptation et la facilité de programmation sont des éléments nécessaires à une utilisation plus répandue de la robotique dans les entreprises et l'assistance aux personnes. L'apprentissage par démonstration est une des approches possibles pour répondre à ce problème. Le but de ce domaine est de développer des algorithmes et des interfaces qui facilitent la programmation de robots et permettent à des utilisateurs inexpérimentés de le faire.

Alors que la configuration d'un bras robotique est définie par les angles des articulations, les postures et mouvements sont souvent plus facile à comprendre dans différents espaces de travail. Ces espaces de travail sont exploités par la plupart des approches d'apprentissage par démonstrations. Par contre, les différents modèles de tâches sont souvent appris indépendamment dans les différents espaces de travail et seulement combinés plus tard, lors du contrôle du robot. Cette simplification implique différentes limitations telles que de déterminer la précision et la hiérarchie entre les différentes sous-tâches considérées. De même, cette simplification empêche la compréhension des tâches secondaires, dont la résolution est masquée par celle de tâches plus importantes.

Dans cette thèse, notre but est de surmonter ces limitations en proposant un cadre de travail rigoureux pour l'apprentissage par démonstrations, fondé sur le produit d'experts. Dans ce modèle, les données sont modélisées comme la combinaison de plusieurs sources, appelées experts. Chacune d'entre elles donne son opinion sur un aspect différent des données, qui correspond aux différents espaces de travail. Mathématiquement, les experts sont des distributions de probabilité qui sont multipliés entre eux et leur produit renormalisé. Des distributions de deux natures différentes sont étudiées dans cette thèse.

Dans une première partie, les produits d'experts sont utilisés pour modéliser des distributions de configuration de bras robotiques. Ces distributions sont des éléments clés de plusieurs approches d'apprentissage. Ils sont souvent utilisés pour définir des mouvement en considérant une dépendance supplémentaire à une variable de phase. Au travers de multiples expériences, nous montrons les avantages apportés par l'apprentissages dans plusieurs espaces de travail simultanément. Plusieurs comparaisons avec des modèles plus généraux de l'état de l'art sont exécutées. Toutefois, l'apprentissage simultané requiert des approximations coûteuses. Une approche d'entraînement alternative, reposant sur les techniques variationnelles, est développée. Finalement, une extension du produit d'experts est proposée pour introduire une hiérarchie entre les tâches.

Dans la deuxième partie de la thèse, les produits d'experts sont utilisés pour l'apprentissage de contrôleurs stochastiques. Nous proposons l'apprentissage de primitives de mouvements telles que des distributions de trajectoires. À la place d'estimer des constantes de normalisations complexes, l'approche antagoniste générative est étudiée. Les contrôleurs sont paramétrés comme des produits de distributions Gaussiennes de différentes natures (vitesse, accélération ou force), agissant dans différents espaces de travail. En fournissant un modèle approximatif de la dynamique du système, le contrôleur est entraîné par des algorithmes du gradient stochastique, pour que

la distribution de trajectoires exécutées sur le robot ressemble à la distribution de démonstrations.

**Mots-clés:** modèles génératifs, apprentissage par démonstration, robotique, produits d'experts, apprentissage par imitation

# Summary

Adaptability and ease of programming are key features necessary for a wider spread of robotics in factories and everyday assistance. Learning from demonstration (LfD) is an approach to address this problem. It aims to develop algorithms and interfaces such that a non-expert user can teach the robot new tasks by showing examples.

While the configuration of a manipulator is defined by its joint angles, postures and movements are often best explained under several task spaces. These task spaces are exploited by most of the existing LfD approaches. However, models are often learned independently in the different task spaces and only combined later, at the controller level. This simplification implies several limitations such as recovering the precision and hierarchy of the different tasks. They are also unable to uncover secondary task masked by the resolution of primary ones.

In this thesis, we aim to overcome these limitations by proposing a consistent framework for LfD based on product of experts models (PoEs). In PoEs, data is modelled as a fusion from multiple sources or "experts". Each of them is giving an "opinion" on a different view or transformation of the data, which corresponds to different task spaces. Mathematically, the experts are probability density functions, which are multiplied together and renormalized. Distributions of two different nature are targeted in this thesis.

In the first part of the thesis, PoEs are proposed to model distributions of robot configurations. These distributions are a key component of many LfD approaches. They are commonly used to define motions by introducing a dependence to time, as observation models in hidden-Markov models or transformed by a time-dependent basis matrix. Through multiple experiments, we show the advantages of learning models in several task spaces jointly in the PoE framework. We also compare PoE against more general techniques like variational autoencoders and generative adversarial nets. However, training a PoE requires costly approximations to which the performance can be very sensitive. An alternative approach to contrastive divergence is presented, by using variational inference and mixture model approximations. We also propose an extension to PoE with a nullspace structure (PoENS). This model can recover tasks that are masked by the resolution of higher-level objectives.

In the second part of the thesis, PoEs are used to learn stochastic policies. We propose to learn motion primitives as distributions of trajectories. Instead of approximating complicated normalizing constants as in maximum entropy inverse optimal control, we propose to use a generative adversarial approach. The policy is parametrized as a product of Gaussian distributions of velocities, accelerations or forces, acting in different task spaces. Given an approximate and stochastic dynamic model of the system, the policy is trained by stochastic gradient descent, such that the distributions of rollouts match the distribution of demonstrations.



**Keywords:** generative models, learning from humans, learning from demonstration, robotics, product of experts, imitation learning

# Contents

# List of Figures

# List of Tables

*The fact is, that civilization requires slaves. The greeks were quite right there. Unless there are slaves to do the ugly, horrible, uninteresting work, culture and contemplation become almost impossible. Human slavery is wrong, insecure, and demoralizing. On mechanical slavery, on the slavery of the machine, the future of world depends. [...] Up to the present, man has been, to a certain extent, the slave of machinery, and there is something tragic in the fact that as soon as man had invented a machine to do his work he began to starve. This, however, is, of course, the result of our property system and our system of competition. One man owns a machine which does the work of five hundred men. Five hundred men are, in consequence, thrown out of employment, and, having no work to do, become hungry and take to thieving. [...] Were that machine the property of all, every one would benefit by it. It would be an immense advantage to the community.*

Oscar Wilde

# 1 Introduction

About one century and a half later, Oscar Wilde's wise words [120] seem more relevant than ever. These noble ideas of public ownership having only partially survived into the twentieth century [37], a solution to the referred problem still needs to be found. Maybe, a better idea would be to give each one of those five hundred men a machine and granting knowledge to use it.

Back in the fifteenth century, Johannes Gutenberg had already played an important role in such a learning process, which earned him the honorific title of "Man of the Millennium"[42]. His democratic innovation made possible the intellectual, political or religious developments in the following centuries [38]. Nowadays, internet and MOOCs are taking over the relay of sharing and dramatically accelerating its spread [39]. Furthermore, at the material level, 3D printing recently offers a democratic solution for production [109].

Regarding robotics, the democratization is still in its early stages [33, 89]. Traditional robots are expensive, hazardous to work with and difficult to program. Today, their use is only beneficial in big factories, where they perform predefined tasks in a highly controlled environment. The cost of setting up this technology is excessively high for Small and Medium-sized Enterprises (SMEs), which creates unequal opportunities as compared to bigger corporations [36, 108]. Besides cheaper platforms, the emerging generation of robots targets multiple characteristics to be equally attractive to SMEs:

- Easy and fast to program: a non-expert user should be able to quickly set the robot (in few minutes or hours).

- Flexible to use: the platform and its programming interface should be flexible to tackle a wide range of tasks.

- Adaptation capacity: the robot should be able to work in a less structured environment in which the task execution might need to be adapted.

- Safe: it should be safe to interact with, and allow different people to work in the same space.

These abilities are also key components for a wider spread of personalized assistive robotics, or broadly assistive technologies. In the case of assisting a person to dress, the robot should be able, for instance, to adapt to different morphologies, pathologies or stages of recovery, which implies a wide range of requirements for movement generation and physical interactions. This behavior is person-dependent, and cannot be pre-engineered (not fixed in time). It must continuously adapt to the user by considering acclimating or rehabilitation periods and aging.

Multiple learning paradigms can assist in meeting these requirements. Reinforcement learning [114] is one of them, promising autonomous learning of skills with minimal human intervention. Provided with rewards, the robot learns to behave in a way to maximize them. As attractive as it may be, this strategy has several drawbacks. It relies either on a long and possibly dangerous exploration phase in the system or in simulation. In some of the latest work [46], robots can learn to open a door with about 3 hours of physical interaction. Concerning simulation, its setup requires dedicated knowledge and might be hard if not infeasible (for example in the case of human-robot interaction). A better-suited paradigm for the targeted applications (SMEs and assistive robotics) is learning from demonstration (LfD) or imitation learning. Its purpose is to learn how to perform a task from demonstrations of a human expert. These demonstrations can be collected by different means: for example through kinesthetic teaching, where the data is acquired by hand guiding the robot.

To achieve such a challenge, this thesis proposes to treat holistically the multimodal sensory information acquired by the robot (awareness of its environment) together with its physical actuation (capability to act in and modify its environment). Concretely, we will build upon the popular notion of movement primitives widely used as building blocks in robotics to create complex motor programs [57]. However, instead of limiting it to motion trajectories, we will consider the sensory computation and physical actuation altogether. To achieve this, we plan to enlarge the notion of movement primitives to a richer set of behaviors by considering primitives as local phenomena or responses including reaction, sensorimotor and impedance primitives. The development of machine learning tools is aimed to treat the information statistically by focusing on high-dimension and multimodal coordination aspects, and considering models encoding both time-dependent and time-independent behaviors. The developed algorithms will specifically focus on exploiting robotic prior knowledge such as to increase the efficiency of the learning procedure.

## 1.1 I-DRESS project

This thesis is part of a CHIST-ERA project, I-DRESS, which aims to develop a robotic platform providing physical assistance for dressing tasks to disabled users. Two scenarios were investigated: help putting on a shoe or a coat. For this project, we collaborate with Bristol robotics laboratory and UPC Barcelona. Their roles are complementary and focus on user studies, perception and higher-level planning.

This assistance is currently provided by healthcare workers, which is not always convenient. From the workers perspective, there is a lack of employees dedicated to this service, while the activity takes time and effort and is certainly not gratifying. From the patients perspective, such an assistance is not fairly well received because it significantly reduces their sense of independence (e.g., in order to go out, the patient depends on someone else to get dressed). Providing robots with dressing assistance capabilities would have benefits on both sides. However, it is still not possible to pre-program all the dressing behaviors and requirements in advance. In this context, the LfD paradigm provides a human-oriented solution to transfer such assistive skills from a non-expert user to the robot. This can be achieved by means of kinesthetic teaching or motion capture system, where several demonstrations of the task executed in various situations can be used to let the robot rapidly acquire the person-specific requirements and preferences.

**Preliminary work** The applications proposed in the I-DRESS project was the topic of two publications [19, 94]. These works are only superficially presented in this thesis, in the following paragraphs. The limitations of the techniques presented in these works motivated a deep reassessment and the need for new approaches. The limitations are discussed in Sec. 2.1.1.

In LfD, skills are generally decomposed into elementary building blocks or movement primitives (MPs) that can be recombined in parallel and in series to create more complex motor programs. They are particularly suitable to generate motion trajectories. However, in order to tackle the highly multimodal interaction involved in assistive tasks, the notion of movement primitives should be enlarged to a richer set of behaviors including reaction, sensorimotor and impedance primitives.

In particular, such a model should also be able to encode both time-independent and time-dependent behaviors. A typical example of time-independence in this human-centric context arises when holding a coat and waiting for someone to come; the duration of the associated movement primitive to help the person should be here triggered by the proximity and attention of the user, and, in this case, it is time-independent. Other parts of the skill are in contrast time-dependent when a movement needs to be completed after being initiated, which typically appears when more dynamic features are required, such as slipping on pants. Often, a relative time dependence is required to guarantee a cohesive evolution of the movement (i.e., with a local time instead of an absolute time).

The above issue is closely related to the problem of organizing the movement primitives in series and parallel, as well as for deciding which one to choose and when to switch between them. In [94], the approach is based on a generative model,

encoding sensorimotor time-series. Motions are generated by computing conditional distributions in this model, allowing the robot to react to its perception, such as the intentions of the user and his movements. In [19], the activation of low-level movement primitives is decided by a high-level discrete planner. It imposes to model the problem by hand at the discrete level by subdividing the task into elementary motions and hard-coding transition rules between them. In these two works, we demonstrate the capability of these approaches with a Baxter robot, by considering the task of helping a user to put on the sleeve of a jacket and a shoe.

## 1.2 Thesis organization

**Background** This chapter gives an overview on the thesis research background. The first section 2.1 focuses on relevant approaches in learning from demonstrations. It is presented under the perspective of probability distributions. In the following sections, more general knowledge about distributions and robotics is presented. In Sec. 2.2.1, a link between distributions and cost functions is presented. Then, in the following section 2.3, various methods to approximated unnormalized densities are reviewed. Those techniques can be used to approximate product of experts, which are presented in the next section 2.4. The chapter closes on with Sec. 2.5, which gives basic knowledge about robotics.

**Product of task-space experts** This chapter addresses the first part of the research question: how to preserve and exploit the information about the structure of the kinematic chain when learning distributions in several task spaces? The solution proposed in this thesis is to learn distributions of robot configurations $p(\boldsymbol{q})$ as a combination of distributions defined in several task spaces. To achieve that, product of experts (PoE) are proposed as a consistent framework to learn $p(\boldsymbol{q})$.

In Section 3.1.3, PoEs are formally presented and the practical implications of properly normalizing the density for robotics are discussed. In the second part of the section, a method to train PoEs using variational inference is proposed. In Section 3.2, an extension of PoEs with nullspace filter (PoENS) is proposed. This extension makes it possible to learn multiple prioritized tasks and their hierarchy. Secondary tasks, which are masked by the resolution of more important tasks, can also be recovered with this approach. Several distributions and transformations are presented in Section 3.3. They help the practitioner to tackle a wide range of robotic problems. In Section 3.4, two different control strategies compatible with PoEs are presented. Finally, in Section 3.5, several experiments are presented to compare the proposed models with other density estimation techniques.

**Product of policies** This chapter addresses the second part of the research question: how to preserve and exploit the information about the structure of the kinematic chain and the dynamics when learning movement primitives in several task spaces? The solution proposed in this chapter is to define a Gaussian controller in each task space. The robot is controlled with the fusion of these controllers, as a product of

Gaussians. These controllers are optimized such that the distribution of trajectories executed on the robot is similar to the distribution of demonstrations, compared in the different task spaces. To achieve that, we propose to use the generative adversarial framework.

In Sec. 4.1, a method to train MPs in the generative adversarial framework is presented. In Sec. 4.1.1, the structure of a trajectory is presented as a state space model, which includes a policy, an observation and a dynamic model. A policy structured as a product of Gaussian policies defined in different task spaces is presented. In Sec. 4.1.2, we propose a particular way to compare the distribution of trajectories executed on the robot and the distribution of demonstrations. The aim is to increase the stability and reduce the need for demonstrations of the training process. In Sec. 4.2, we propose a method to treat uncertainties in the dynamic models. Some parametrization of the Gaussian policies are presented in Sec. 4.3. Finally, in Sec. 4.4, several experiments are proposed.

**Summary and recommendations** In this chapter, the main results of this thesis are summarized. We also propose several open research directions.

# 2 Background

This chapter gives an overview of the thesis research background. The first section 2.1 focuses on relevant approaches in learning from demonstrations, presented under the perspective of probability distributions. In the following sections, more general knowledge about distributions and robotics is presented. In Sec. 2.2.1, a link between distributions and cost functions is presented. Then, in Sec. 2.3, various methods to approximate unnormalized densities are reviewed. These techniques can be used to approximate products of experts, which are presented in the next section. The chapter closes on with Sec. 2.5, which gives basic knowledge about robotics.

## 2.1 Learning from demonstration

In this section, we will review some of the notable works in learning from demonstrations (LfD) from a statistical perspective.

In LfD, the robot is expected to imitate the expert's behavior. It is expected to act similarly under the same conditions and ideally to generalize the behavior to new situations. To model the problem more formally, the information about the system (robot and environment) at time $t$ is summarized as the state variable $\boldsymbol{\xi}_t$. It can include positions, velocities of some parts of the robot as well as positions of external objects or users. Often, the state is not directly accessible but only through a partial and noisy observation $\boldsymbol{y}_t$. The action of the robot $\boldsymbol{u}_t$ is chosen according to a policy function taking into account current and previous observations. A sequence of actions and observations (or states) is often referred to as a trajectory $\boldsymbol{\tau} = \{\boldsymbol{y}_1, \boldsymbol{u}_1, \dots, \boldsymbol{y}_T, \boldsymbol{u}_T\}$.

A natural cost function for LfD [32] should make the distribution of trajectories $p(\boldsymbol{\tau})$ from the robot and the expert match. Before reviewing the different approaches to achieve this goal, the use of probability distributions needs to be motivated.

Learning a distribution of trajectories or policies, as opposed to only the optimal one, can be justified by multiple reasons. In reinforcement learning, it is important that the agent explores sufficiently to avoid falling into poor local optima [74]. The exploration can be led by a guiding distribution, which can be initialized by LfD [74]. This initialization can overcome the typically very long exploration phase [82]. Exploration is also required when the robot has to learn to predict the effect of its actions, namely a dynamic model. A challenging problem is to perform this task safely, by avoiding damages. This can be done by imitating search patterns from human [30] or setting up an optimal control problem to cover a distribution of states [4]. A similar stochastic search can be found in surveillance and search applications [8]. Similarly, some manipulation tasks like cleaning, mixing or polishing require stochasticity.

Distributions are also fundamentals for intention recognition. For example, in human-robot collaboration, the robot should make sure that its intentions are conveyed while performing the task [27]. A formal way to infer the intention given the observation of trajectories is to apply Bayes rule. It requires to model multiple conditional distributions of trajectories given different goals and their normalizing constant. Variability in the execution also makes the motion look more human-like. An evident example is the synthesis of handwriting which should exhibit some variability [25].

Keeping a distribution of solutions is also necessary for adaptation. Adaptation might be required by the inclusion of new objectives to fulfill, such as impromptu obstacles. Additional controllers, completing these new objectives, can be combined with the current solution [17, 92, 101]. Adaptation might also be required by unforeseen perturbations, which can be provided by the use of trajectory planning or optimal control. Those techniques benefit from a library of good and diversified initial guesses [72]. This library can be stored as a conditional distribution of trajectories or policies.

### 2.1.1 From static distributions to trajectories

In LfD, many methods model trajectories by using static probability distributions of observations $\boldsymbol{y}_t$ with an additional dependence to a phase variable. We critically review these methods by outlining two problems which result in distortions at the final level of the trajectories.

The first problem arises from the way the dynamic features or dependence to time is introduced. In [17], a time-conditioned distribution of observations is learned using Gaussian mixture regression (GMR). In order to treat unaligned, partial and cyclic demonstrations, the use of hidden semi-Markov models (HSMM) was proposed in [16, 18]. HSMM can encapsulate precisely position and timing information about motions. In [92] motion is induced by a time-dependent basis matrix.

A common point of these techniques is that the true temporal dependencies (induced by the dynamics of the system) are not taken into account at the learning phase, but only introduced at the synthesis stage. For example, HSMM assumes that two consecutive observations are independent given the discrete latent state. In [16], dynamic consistency is only re-introduced later using linear quadratic tracking (LQT).

From a statistical point of view, this problem can be explained in several ways. It can be interpreted as dropping important dependencies at learning and introducing them back at synthesis. Alternatively, it can be interpreted as not considering a properly normalized density on the space of trajectories $\boldsymbol{\tau}$.

This problem was already raised in the context of parametric speech synthesis [126]. For hidden Markov models, the solution is given by explicitly imposing a dynamic relation [125]. In robotic, the equivalent of this dynamic constraint is the dynamic model, which is often not known and stochastic.

The second problem of these approaches is the handling of the multiple task spaces in which the statistical models are learned. Because of redundancies and non-linearities, movements are often difficult to understand directly in joint space. LfD approaches often learn models in different tasks spaces, which are task-relevant transformations of the joint space. Position and orientations of different parts of the robot are typically used. Less common features like manipulability measures [104] can also be considered as task spaces. Several approaches encode motions in several coordinate systems attached to objects of interest [16, 81, 86]. The idea is to provide an adaptation to the motions of these objects by adopting an object-centric representation. The problem of these LfD approaches is that the different task spaces are considered independently at the learning stage and their dependence only restored when synthesizing trajectories, in the same way that it is done for the dynamic consistency. Several works that encode motion in multiple task spaces and joint spaces are reviewed here.

In [17], objectives from multiple task spaces and configuration space are considered. Multiple time-dependent Gaussian distributions of end-effector positions and joint angles are learned separately, with an assumption of independence. These competing objectives are only combined at the control stage, by exploiting a product of Gaussians in the velocity space. In [16], multiple Gaussian models of motion are learned independently in different task spaces. Using properties of linear transformations and product of Gaussians, the different models are combined in closed-form before synthesizing sequences. In [124], this approach is extended to the encoding of orientation in multiple coordinate systems. In [92], the authors propose a combination of multiple Gaussian distributions of trajectories (ProMP). The different models are also defined both in configuration space and in different task spaces but are learned separately. They are then only combined at the controller level, as different acceleration commands.

Similarly, in [111], the authors proposed to fuse multiple independent Gaussian models in different task spaces and configuration space as a product of Gaussians. The fusion is approximated locally using a linearization of the forward kinematics. The authors also propose to learn a hierarchy of tasks using linear transformations with nullspace filters. Their approach is however limited to velocity commands and tasks where the target is known. The method that we present in Sec. 3.2 can cope with samples of static configurations $\boldsymbol{q}$, which requires more advanced tools to take into account the non-linearities. The different tasks do not need to be known before and our method can uncover auxiliary tasks that are masked by primary ones.

In [86], segments of trajectories are analyzed in different coordinate systems, simi-

larly to [16]. A unique coordinate system is then selected for each segment, by looking at the similarity between end-points. The idea of selecting relevant task spaces is explored in several other works. In [81], the demonstrations are projected in a set of task spaces. Different criteria, such as the variance of the data and saliency, are used.

In [77], the variance of several distributions learned separately is mapped to weights modulating the task prioritizations. The variances are learned from several demonstrations as in [17].

A common point of these works is that the different models are learned independently and their prioritization or importance is related to their variance. Indeed, secondary tasks do exhibit a higher variance but the experiments confirm the necessity to distinguish between the variance and the prioritization. In the experiments in Sec. 3.5, we show that competing tasks can be understood only if the models are trained jointly, especially to understand the characteristics of secondary tasks.

**Distributions outside LfD** Outside LfD, learning distributions of robot configurations is also a topic of interest for sampling-based path planning [5]. Sampling-based path planning methods require to sample valid configurations (e.g. collision-free) and to connect them to create paths. Randomly sampling the configuration space and keeping valid samples is not efficient because of a possible low acceptance rate. Some works focus on learning the distribution to sample from. In [56], conditional distributions are learned using a conditional variational autoencoder [113]. The distribution is conditioned on some external factors, such as an obstacle occupancy grid, to provide adaptation. In [71], the authors use a similar approach with a Gaussian mixture model learned from previous collision-free configurations.

### 2.1.2 Conditional models

A very generic approach for LfD and more generally for imitation learning is model-free behavioral cloning. In the statistical case, the training is reduced to learn the conditional distribution $p(\boldsymbol{u}_t|\boldsymbol{y}_t)$ in a supervised manner given pairs of control command and observations. This conditional distribution is called the policy and is often denoted $\boldsymbol{\pi_\theta}(\boldsymbol{u}_t|\boldsymbol{y}_t)$. It can be learned with several techniques, such as locally weighted regression (LWR) [7], locally weighted projection regression (LWPR) [116], neural networks [28] or GMR [63]. Despite its apparent simplicity and generality, this method has one major drawback, which prevents its wider use in robotics: policies trained by behavioral cloning are known to be very brittle. Due to modelling errors, perturbations or different initial conditions, executing such a policy can quickly lead the robot far from the distribution of states visited during the supervised learning phase. This problem is often referred to as distributional shift [103]. When applied in a real system, the actions can therefore be dangerous and lead to catastrophic outcomes.

Several works have tackled the distributional shift problem in the general case. In [103], the authors used a combination of an expert and a learner policy. In [69], perturbations are added in order to force the expert to demonstrate recovery strategies, resulting in a more robust policy.

A subset of these approaches focuses on learning manipulation tasks from a small set of demonstrations [50, 63, 91]. In order to guarantee safe actions, these techniques typically impose specific constraints on the policy, by introducing time-dependence structures or by developing hybrid, less general approaches.

More closely related to the manipulation tasks considered in this thesis, Khansari-Zadeh et al. have used GMR to learn a policy of the form $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ [63]. They used conditional distributions $p(\dot{\boldsymbol{x}}|\boldsymbol{x})$ in a joint model of $p(\boldsymbol{x}, \dot{\boldsymbol{x}})$ represented as a Gaussian mixture model (GMM). A structure was imposed on the parameters to guarantee asymptotic convergence but limits the variety of tasks that can be learned.

Dynamical movement primitives (DMP) is a popular approach that combines a stable controller (spring-damper system) with non-linear forcing terms decaying over time through the use of a phase variable, ensuring convergence at the end of the motion [110]. A similar approach is used in [50] where the non-linear part is encoded using conditional distributions in GMM. Due to their underlying time dependence, these approaches are often limited to either point-to-point or cyclic motions of a known period, with limited temporal and spatial robustness to perturbations.

If a reward function for the task is accessible, an interesting alternative is to combine the proposed policy imitation strategy with reinforcement learning [82, 99], where the imitation loss can reduce the exploration phase while the reinforcement learning overcomes the limits of imitation.

In some classifications of techniques [88], methods encoding distribution of trajectories like [17, 92] are seen as behavioral cloning. In their case, the mapping is from a context variable to whole trajectories. The problem, which we have already pointed out in the previous section, is that the resulting trajectories might not be feasible. The system is referred to as being underactuated. Also, reintroducing dynamical consistency before synthesis often introduces distortions.

Besides stability issues, two other problems limit the usage of supervised behavioral cloning. First, the control command may not be observed. For example, for kinesthetic teaching, where the human user demonstrates the tasks by directly manipulating the robot, the torques command of the robot cannot be measured. They could be if teleoperation were used. Another example arises when a robot needs to imitate by observing a human or another robot. In this case, internal actions like forces are not accessible. A second problem, that arises also in the same context is known as the correspondence problem [11]. The kinematics and dynamics of the system on which the demonstrations are performed may not match those from the learner.

### 2.1.3 Model-based behavioral cloning

Many of the problems of model-free behavioral cloning techniques are addressed by their model-based alternatives. However, they are more complex to set up, because of the need to learn a dynamics model and their computational complexity. In [45], the dynamic model is learned with GMR. This model is used to match observation sequences from a learner and a teacher with different dynamics. It allows learning stable whole-body motions of a humanoid robot [44]. In [32], the dynamic model

is learned using Gaussian processes (GP). Following [21], the distribution of states is approximated as factorized Gaussians using moment matching. The policy is optimized to minimize the KL divergence between the distribution of demonstrations and the predicted distribution of trajectories. This method offers a more rigorous approach to imitate distributions of trajectories as compared to hidden Markov models or GMR models presented in the previous section. The cost of matching trajectories also produces more robust policies than the model-free supervised learning cost. This would be shown in an experiment in Sec. 4.4.2.

In [3], target trajectories from helicopter aerobatics maneuverers are inferred from a set of demonstrations. They are then reproduced by iteratively running an LQR-based controller and refining the dynamics model. A similar approach is used in [115] to imitate surgical tasks. However, in both of these works, only an optimal policy is retrieved and the variability of the demonstrated trajectories is lost. Also, the system cannot generalize to new situations and only reproduce fixed trajectories. An extension of this approach is proposed in [16] to reproduce trajectories in multiple coordinate systems attached to objects of interests.

### 2.1.4 Inverse optimal control/inverse reinforcement learning

In inverse optimal control (IOC) or inverse reinforcement learning (IRL), the robot is trying to infer the cost function minimized by the behavior of the demonstrator. The cost function is known to be a more succinct, robust and transferable representation of the task than the policy [2]. This has two major implications. First, it is easier to transfer skills across systems, for example, if a robot should learn a task by observing a human performing it. Secondly, it provides better generalization and adaptation capabilities to varying situations. This approach has however several drawbacks. The main one, pointed out in [85], is that the policy demonstrated can be optimal for many cost functions, thus providing an ambiguity. Different approaches tackle this problem. Maximum entropy inverse reinforcement learning is proposed in [128]. It frames the problem as a maximum likelihood estimation of an intractable (unnormalized) density over trajectories $\boldsymbol{\tau}$

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \propto \exp(-c_{\boldsymbol{\theta}}(\boldsymbol{\tau})), \tag{2.1}$$

where $c_{\boldsymbol{\theta}}$ is the cost function. The normalizing constant is said to be intractable if the integral has no known or easy form. This approach has been applied to robotics where the cost is parametrized with a neural network [35]. These techniques are computationally expensive. They rely on complex approximation methods to compute an estimate of the gradient of the unnormalized density function. This approximation, which has to be done at each step of gradient descent of the parameters of the cost, corresponds to solving a maximum entropy (direct) reinforcement learning problem. In the same way that the generative models presented in Sec. 2.1.1, maximum entropy IOC learns a distribution over trajectories. However, the normalization is done on the space of dynamically consistent trajectories (given the system dynamics), which ensures that the true space is considered.

In robotics, IOC was also applied in [61]. However, their approach is limited to learning a weight vector of different cost features (joint limits, manipulability measure, elbow position). In [59], the cost is also learned on a feature vector of the proposed task spaces. Sparsity is ensured using $L_1$ regularization which allows retrieving and selecting important task spaces. These two latter approaches formulate the problem as inferring a cost that acts on different task spaces. The relation between the different task spaces and the dependencies between the different tasks are thus better taken into account than in the probabilistic models presented in Sec. 2.1.1.

## 2.2 Probability distributions

In this section, we first give an overview of different approaches to learn probability distributions. Then, we discuss several techniques coming from Bayesian statistics to approximate unnormalized densities.

### 2.2.1 Learning probability distributions

Estimating probability density functions have been for long a major field in statistics and machine learning. Estimating complex densities using a sum of simpler densities has been proposed with kernels in [93] or mixture models in [22]. Unfortunately, these techniques can be very inefficient in high-dimensional spaces, such as the configuration of a robot. With the rising popularity of neural networks, deep generative models have been used to learn distributions of higher dimensions, such as images. Restricted Boltzmann machine [53] and deep Boltzmann machine [105] are popular models. Like products of experts (PoE), proposed in [52], these models are trained by maximizing an intractable likelihood function. Several approximations are necessary to compute the gradient of the log-likelihood.

These complex approximations motivated the development of various models that do not represent the likelihood but allow sampling from the distribution like generative networks [9], generative adversarial nets (GAN) [41] and variational autoencoders (VAE) [66]. Unlike previous techniques, they do not require approximations when computing the gradient of the cost to optimize. However, once the model is trained, computing the likelihood is either impossible or requires approximations. These generative machines have been very popular recently and are used to learn complex, high-dimensional distributions from huge datasets.

### 2.2.2 Energy-based models

In energy-based models [70], the density is modeled with an energy value $E_\theta(\boldsymbol{x})$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z}\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x})). \tag{2.2}$$

Training this model by maximizing the likelihood of the data is challenging [52] because of the intractable normalizing constant. This process is detailed in Sec. 3.1.3.

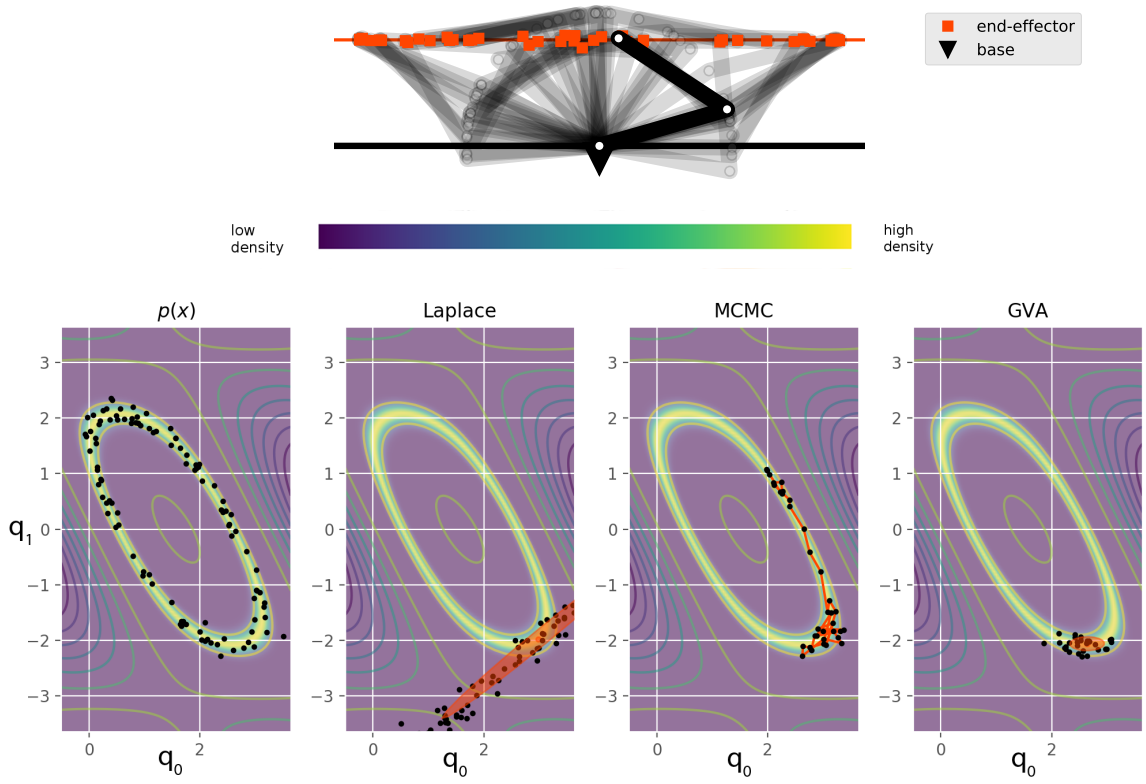



**Figure 2-1:** *Top:* A 2-DoF planar robot constraint used as an illustrative energy model in this section. The energy is defined on the joint angles and corresponds to the quadratic distance between the end-effector and the orange horizontal line. Multiple samples from this model are displayed. *Bottom:* On left, the samples are reported in joint space. The density of the energy model is evaluated on a grid and displayed as a colormap. The approximations discussed in this section and their corresponding samples are displayed on the right.

Maximum likelihood estimation is normally done with gradient descent. Estimating the gradient of the normalizing constant requires to sample from $p_\theta(\boldsymbol{x})$ which is already a hard problem. Moreover, the sampling has to be done for each step of gradient descent while the parameters $\theta$ are continuously changing.

As presented above, maximum entropy inverse optimal control [128] is an energy-based model, where the energy is given by the cost function.

## 2.3 Approximating distributions

Approximation methods are required to work with unnormalized densities. An adequate approximation method should enable several elements: we should be able to draw samples from it, estimate the normalizing constant and its gradient with respect to the model parameters, and also identify the modes of the distribution. Such methods are found in Bayesian statistics, where the same problem of approximat-

ing an unnormalized density is faced. Indeed, in Bayesian statistics, the posterior distribution of model parameters

$$p(\boldsymbol{\theta}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{2.3}$$

is the unnormalized product of a prior distribution $p(\boldsymbol{\theta})$ and a likelihood $p(\boldsymbol{x}|\boldsymbol{\theta})$. The posterior has a closed-form expression only in the case where conjugate priors are used. In this case, the posterior is of the same family as the prior. However, choosing conjugate priors massively restrains modeling choices.

In the following section, we would refer to the unnormalized density as $\tilde{p}(\boldsymbol{x})$. The different approximation methods are illustrated in Fig. 2-1 on a robotic example.

### 2.3.1 Laplace approximation

One of the simplest method to approximate unnormalized density is the Laplace approximation. It relies on Taylor's theorem to perform a second-order expansion on the mode of the unnormalized log-density. The procedure to compute this approximation is simple. First, the mode of the unnormalized density has to be found, for example with gradient descent techniques. Then the curvature of the unnormalized density has to be estimated at the mode. The gradient vanishing at the mode, the unnormalized log-density is approximated as

$$\log \tilde{p}(\boldsymbol{x}) \approx \log \tilde{p}(\boldsymbol{x}_0) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^\top \boldsymbol{H}(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0). \tag{2.4}$$

The distribution is approximated as a Gaussian distribution with the same curvature

$$\mathcal{N}(\boldsymbol{x}|\, \boldsymbol{x}_0, \boldsymbol{H}(\boldsymbol{x}_0)^{-1}). \tag{2.5}$$

This approximation is only valid locally and is poor if the unnormalized density has a complex shape or is multimodal, as shown in Fig. 2-1.

The Laplace approximation is used in robotics to approximate distributions of optimal control trajectories [73, 74]. This distribution is used to guide a policy search exploration or to approximate the gradient of the normalizing constant for IOC in [35].

### 2.3.2 Makov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a class of methods to approximate $p(\boldsymbol{x})$ with samples. If MCMC methods can represent arbitrary complex distributions, they suffer from some limitations.

For example, they are known not to scale well to high-dimensional spaces, which is particularly constraining for our application. A lot of samples are required to cover high-dimensional distributions.

The key component of many MCMC methods is the definition of a proposal move, which makes the random walker (chain) more likely to visit areas of high probabili-

ties. The design of this proposal move is algorithmically restrictive because it needs to ensure that the distribution of samples at equilibrium is proportional to the unnormalized density. Furthermore, it is difficult to obtain good acceptance rates in high dimension, especially with very correlated $\tilde{p}(\boldsymbol{x})$, as shown in Fig. 2-1.

By representing the distribution only through samples, it is also difficult to assess if this latter is well covered. A huge part of the space and distant modes might remain undiscovered. It is also difficult to know the granularity of the approximated distribution.

Except for some particular approaches, such as [20], MCMC methods require an exact evaluation of $\tilde{p}(\boldsymbol{x})$. On the contrary, stochastic variational inference (SVI), presented in the next section, only requires a stochastic estimate of the gradient of $\log \tilde{p}(\boldsymbol{x})$. There are many advantages to this unconstraining requirement. Experts transformations that are too costly to compute exactly can be approximated. If the PoE is conditional, batches of conditional values can be used. Also, the gradient can be redefined such as a hierarchy between the task is set up, as presented in Sec. 3.2.

Finally, MCMC methods struggle with multimodal distributions. Chains are unlikely to cross large regions of small density. If multiple chains are run in parallel, the respective mass of each mode is difficult to estimate. A proper approach of this problem is to design particular proposal steps to move between distant modes, as proposed in [112], which is algorithmically restrictive.

### 2.3.3 Variational inference

Variational inference (VI) [118] is another popular class of methods that recasts the approximation problem as an optimization. VI approximates the *target density* $\tilde{p}(\boldsymbol{x})$ with a tractable density $\tilde{q}(\boldsymbol{x}; \boldsymbol{\lambda})$ called the *variational density*. $\boldsymbol{\lambda}$ are the *variational parameters* and are subject to optimization. A density is called tractable if drawing samples from it is easy and that the density is properly normalized. VI tries to minimize the intractable KL-divergence between the renormalized density $p(\boldsymbol{x})$ and the variational density $\tilde{q}(\boldsymbol{x}; \boldsymbol{\lambda})$

$$D_{\mathrm{KL}}(\tilde{q}||p) = \int_{\boldsymbol{x}} \tilde{q}(\boldsymbol{x}; \boldsymbol{\lambda}) \log \frac{\tilde{q}(\boldsymbol{x}; \boldsymbol{\lambda})}{p(\boldsymbol{x})} d\boldsymbol{x}. \tag{2.6}$$

Given that $p(\boldsymbol{x}) = \tilde{p}(\boldsymbol{x})/\mathcal{C}$ where $\mathcal{C}$ is the normalizing constant, we can rewrite the previous divergence as

$$D_{\mathrm{KL}}(\tilde{q}||p) = \int_{\boldsymbol{x}} \tilde{q}(\boldsymbol{x}; \boldsymbol{\lambda}) \log \frac{\tilde{q}(\boldsymbol{x}; \boldsymbol{\lambda})}{\tilde{p}(\boldsymbol{x})} d\boldsymbol{x} + \log \mathcal{C}, \tag{2.7}$$

which can be evaluated up to a constant and minimized. The first term is the negative *evidence lower bound* (ELBO). This term can be estimated by sampling as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}) &= \int_{\boldsymbol{x}} \tilde{q}(\boldsymbol{x};\boldsymbol{\lambda}) \log \frac{\tilde{q}(\boldsymbol{x};\boldsymbol{\lambda})}{\tilde{p}(\boldsymbol{x})} d\boldsymbol{x}, && (2.8)\\ &= \mathbb{E}_{\tilde{q}}[\log \tilde{q}(\boldsymbol{x};\boldsymbol{\lambda}) - \log \tilde{p}(\boldsymbol{x})], && (2.9)\\ &\approx \frac{1}{N}\sum_{n=1}^{N} \big(\log \tilde{q}(\boldsymbol{x}^{(n)};\boldsymbol{\lambda}) - \log \tilde{p}(\boldsymbol{x}^{(n)})\big), && (2.10)\\ &\text{with} \quad \boldsymbol{x}^{(n)} \sim \tilde{q}(\,\cdot\,|\boldsymbol{\lambda}),\end{aligned}$$

where $N$ is the number of samples. The reparametrization trick proposed in [106] and [100] allows a noisy estimate of the gradient $\mathcal{L}(\boldsymbol{\lambda})$ to be computed. It is compatible with stochastic gradient optimization like Adam [65]. For example, if $\tilde{q}$ is Gaussian, this is done by sampling $\boldsymbol{\eta}^{(n)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and applying the continuous transformation $\boldsymbol{x}^{(n)} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\eta}^{(n)}$, where $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$ is the covariance matrix. $\boldsymbol{L}$ and $\boldsymbol{\mu}$ are the *variational parameters* $\boldsymbol{\lambda}$. More complex mappings as normalizing flows can be used as in [102].

### Zero forcing properties of minimizing $D_{\mathrm{KL}}(q||p)$

It is important to note that due to the objective $D_{\mathrm{KL}}(\tilde{q}||p)$, $\tilde{q}$ is said to be zero forcing. If $\tilde{q}$ is not expressive enough to approximate $\tilde{p}$, it would miss some mass of $\tilde{p}$ rather than giving a high probability to locations where there is no mass.

### Mixture model variational distribution

For computational efficiency, the *variational density* $\tilde{q}(\boldsymbol{x};\boldsymbol{\lambda})$ is often chosen as a factorized distribution, using the mean-field approximation [118]. Correlated distributions can be approximated by a Gaussian distribution with full covariance [87]. These approaches fail to capture the multimodality and arbitrary complexity of $\tilde{p}(\boldsymbol{x})$. The idea to use a mixture for greater expressiveness as *approximate distribution* was initially proposed in [14], with a recent renewal of popularity [6, 47, 80].

A mixture model is built by summing the probability of $K$ mixture components

$$\tilde{q}(\boldsymbol{x}|\boldsymbol{\lambda}) = \sum_{k=1}^{K} \pi_k \, \tilde{q}_k(\boldsymbol{x}|\boldsymbol{\lambda}_k), \quad \sum_{k=1}^{K} \pi_k = 1, \tag{2.11}$$

where $\pi_k$ is the total mass of component $k$. The components $\tilde{q}_k$ can be of any family accepting a continuous and invertible mapping between $\boldsymbol{\lambda}$ and the samples. The discrete sampling of the mixture components according to $\pi_k$ has no such mapping.

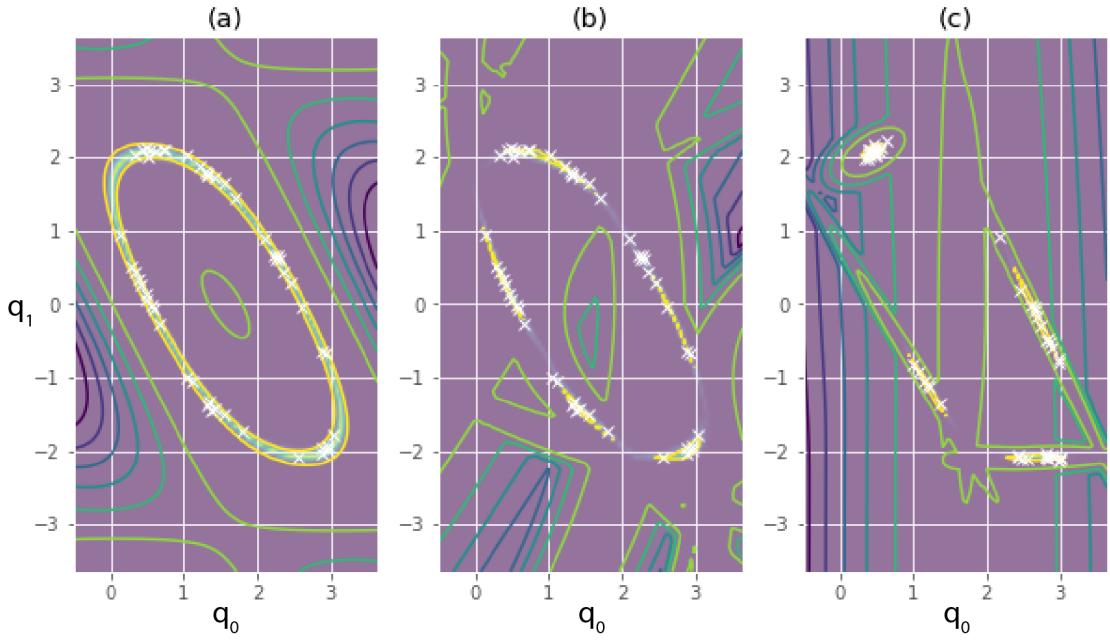



**Figure 2-2:** Variational approximation of an unnormalized density with mixture models. The density shown in *(a)* results from an elongated Gaussian defined on the end-effector of a 2-DoF planar robot. The density is approximated by a tractable distribution, as a mixture of banana-shaped distributions *(b)* or Gaussians *(b)*. On purpose, only 5 mixture components were used in the illustrations to better highlight the differences. A more precise approximation can be achieved with a higher number.

Instead, the variational objective can be rewritten as

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{\tilde{q}}[\log \tilde{q}(\boldsymbol{x};\boldsymbol{\lambda}) - \log \tilde{p}(\boldsymbol{x})], \tag{2.12}$$

$$= \sum_{k=1}^{K} \pi_k \mathbb{E}_{\tilde{q}_k}[\log \tilde{q}(\boldsymbol{x};\boldsymbol{\lambda}) - \log \tilde{p}(\boldsymbol{x})], \tag{2.13}$$

meaning that we need to compute and get the derivatives of expectations only under each component distribution $\tilde{q}_k(\boldsymbol{x}|\boldsymbol{\lambda}_k)$.

### Mixture components distributions

Gaussian components with full covariance matrix are a natural choice for robotics, due to the quadratic form of its log-likelihood. It can be exploited in standard robotics approaches like linear quadratic tracking (LQR) or inverse kinematics (IK). For some situations, where $\tilde{p}(\boldsymbol{x})$ is very correlated, we propose to use *"banana-shaped"* distribution [48], which is done by applying the following differentiable mapping to Gaussian

samples $\boldsymbol{\eta}^{(n)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$,

$$f_{\boldsymbol{\kappa}}(\boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\eta}^{(n)}) = f_{\boldsymbol{\kappa}}(\boldsymbol{z}^{(n)}) = \begin{bmatrix} z_0^{(n)} + \sum_{i=1}^{N} \kappa_i \, {z_i^{(n)}}^2 \\ z_1^{(n)} \\ \vdots \\ z_N^{(n)} \end{bmatrix}. \tag{2.14}$$

This mapping can be applied along different parametrized directions. As illustrated in Fig. 2-2 (b), we get a Gaussian with full covariance, where $\boldsymbol{\kappa}$ is a supplementary *variational parameter* encoding curvature.

## 2.4 Product of experts

Products of experts have been proposed in [51] as an alternative to mixture models to compensate for their poor efficiency in high-dimensional space. The combination of the distributions (called experts) is done by a product instead of a summation, which provides much sharper distributions. If computing the normalizing constant of a sum of distributions is straightforward, the major problem with PoEs, being energy-based models, is to compute this quantity and its derivative, which requires approximate methods. Sampling from a PoE is also much more difficult than from a mixture model.

Many works have used the term "product of experts" to express the fusion of several models. Many of them are also not considering the joint training of the models as originally proposed in [52]. For example, two long short-term memory (LSTM) are combined in [60] to generate jazz melodies. In robotics, the fusion of multiple sources of data (e.g. sensors) was expressed as a PoE in [97], for the localization of mobile robots and obstacle avoidance. In [16], the fusion of trajectory models learned independently in multiple coordinate systems is also referred to as a product of Gaussians (PoG).

Jointly training the experts, as proposed in [52], has been used in several applications. In robotics, a product of contact models is proposed in [67] to predict the motion of manipulated objects. In speech processing, PoEs have been used for vowel classification in [24]. In [126], speech sequences are modeled using a product of multiple acoustic models. These acoustic models consist of transformations, both linear (discrete cosine transform, summation) and non-linear (quadratic), and various distributions, such as Gaussian, Gamma or log-Gaussian. The authors claim that distortions of the generated speech are often due to the different acoustic models being learned separately and only combined later, at the synthesis stage. The authors proposed to use the PoE framework, enabling the training of multiple acoustic models cooperatively. Our claim is that, for robotics applications, learning the different models separately and combining them later, also induce heavy distortions. Many works learn different models (in configuration space, in task space, manipulability measures, ...) separately and only combine them at the control stage. In Chapter 3,

we show the advantages of learning these models collaboratively in a way that the dependencies between the different features are kept and the kinematic structure of the robot taken into account.

### 2.4.1 Product of Gaussians

In the general case, the renormalized product $p(\boldsymbol{x})$ has no closed-form expression. A notable exception is Gaussian experts with linear transformations. In this case, the product $p(\boldsymbol{x})$ is Gaussian. Many techniques such as Kalman filter or linear quadratic tracker (LQT) [62] can be reformulated as a product of Gaussians (PoG) under linear transformations [16]. Another application of PoGs is to fuse trajectories encoded in several coordinate systems to provide adaptation to moving objects [16].

We recall here the main relations to compute the product of linearly transformed Gaussians. These results will be useful in Chapter 4 to fuse multiple controllers in an efficient and differentiable manner. The Gaussian distribution can also be parametrized with $\boldsymbol{\eta}$ and $\boldsymbol{\Lambda}$ as

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \equiv \mathcal{N}_{\boldsymbol{\eta}}(\boldsymbol{\eta}, \boldsymbol{\Lambda}) \quad \text{with} \quad \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}, \boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}. \tag{2.15}$$

This parametrization facilitates the notations and prevents the use of pseudo-inverse, both in the equations and software implementations. A first well-known result, obtained by the properties of expectation, is that linear transformation of Gaussian random variable are Gaussian. If the variable $\boldsymbol{x}$ is Gaussian $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\boldsymbol{z} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{c}$, then $\boldsymbol{z}$ is Gaussian

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{C}\boldsymbol{\mu} + \boldsymbol{c}, \boldsymbol{C}\boldsymbol{\Sigma}\boldsymbol{C}^\top). \tag{2.16}$$

Depending on the rank of $\boldsymbol{C}$, this distribution might be degenerated (the covariance matrix has eigenvalues equals to zero).

We now consider the case where we observe $\boldsymbol{x}$ trough a linear transformation as $\boldsymbol{z} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{c}$. We know that $\boldsymbol{z}$ is distributed as a Gaussian $\boldsymbol{z} \sim \mathcal{N}_{\boldsymbol{\eta}}(\boldsymbol{\eta}, \boldsymbol{\Lambda})$ and we would like to infer the distribution of $\boldsymbol{x}$. By explicitly inverting the linear transformation we get

$$\boldsymbol{C}\boldsymbol{x} + \boldsymbol{c} \sim \mathcal{N}_{\boldsymbol{\eta}}(\boldsymbol{\eta}, \boldsymbol{\Lambda}), \tag{2.17}$$

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{C}^\dagger(\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} - \boldsymbol{c}), \boldsymbol{C}^\dagger\boldsymbol{\Sigma}\boldsymbol{C}^{\dagger\top}), \tag{2.18}$$

where $\cdot^\dagger$ denotes the pseudo-inverse. In the case where $\boldsymbol{C}$ is not invertible, the other parametrization can be used to cancel out the pseudo-inverse

$$\boldsymbol{x} \sim \mathcal{N}_{\boldsymbol{\eta}}(\boldsymbol{C}^\top\boldsymbol{\Lambda}\boldsymbol{C}\boldsymbol{C}^\dagger(\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} - \boldsymbol{c}), \boldsymbol{C}^\top\boldsymbol{\Lambda}\boldsymbol{C}), \tag{2.19}$$

$$\boldsymbol{x} \sim \mathcal{N}_{\boldsymbol{\eta}}(\boldsymbol{C}^\top(\boldsymbol{\eta} - \boldsymbol{\Lambda}\boldsymbol{c}), \boldsymbol{C}^\top\boldsymbol{\Lambda}\boldsymbol{C}). \tag{2.20}$$

In this case, the Gaussian can also be degenerate with eigenvalues of the precision matrix equals to zero. For example, if $\boldsymbol{x}$ are joint velocities of a redundant manipulator, $\boldsymbol{z}$ linear and angular velocities of the end-effector and $\boldsymbol{C}$ the Jacobian, the

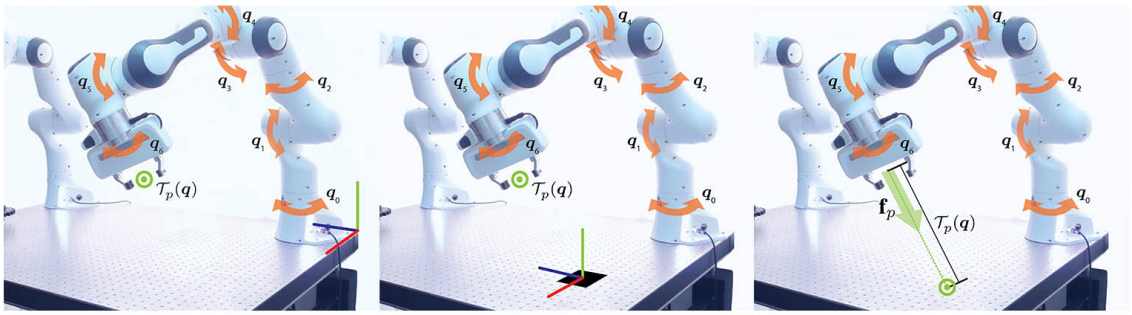



**Figure 2-3:** Example of task spaces. *Left:* Position and orientation of the end-effector in the global coordinate system. *Center:* Position and orientation of the end-effector in a coordinate system attached to an object of interest. *Right:* Distance to an object.

distribution of joint velocities would have a free axis. This distribution will only be valid if fused as a PoG with a regularizing distribution of joint velocities.

The fusion has a simple expression with the alternative parametrization [121]

$$\mathcal{N}_{\boldsymbol{\eta}}\Big(\boldsymbol{x}\,\Big|\,\sum_{p=1}^{P}\boldsymbol{\eta}_p,\sum_{p=1}^{P}\boldsymbol{\Lambda}_p\Big)\propto\prod_{p=1}^{P}\mathcal{N}_{\boldsymbol{\eta}}(\boldsymbol{x}\,|\,\boldsymbol{\eta}_p,\boldsymbol{\Lambda}_p). \tag{2.21}$$

However, the result is easier to interpret with the standard parametrization as the mean of the product becomes a weighted sum of the means

$$\Big(\sum_{p=1}^{P}\boldsymbol{\Sigma}_p^{-1}\Big)^{-1}\sum_{p=1}^{P}\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}_p=\Big(\sum_{p=1}^{P}\boldsymbol{\Lambda}_p\Big)^{-1}\sum_{p=1}^{P}\boldsymbol{\Lambda}_p\boldsymbol{\mu}_p, \tag{2.22}$$

where the weights are given by the precision matrices.

## 2.5 Robot kinematics and dynamics

The $d_{\boldsymbol{q}}$-dimensional state of a fixed-base robotic manipulator is defined by its joint angles $\boldsymbol{q}$. The poses and movements are often best explained in several task spaces. In this thesis, task spaces are not limited to the position and orientation of the end-effector; they are defined as a set of $P$ non-linear functions $\mathcal{T}_{\boldsymbol{\xi},p}:\mathbb{R}^{d_{\boldsymbol{\xi}}}\rightarrow\mathbb{R}^{k_{\boldsymbol{\xi}}^p}$. Accordingly, a set of linear functions maps control command $\boldsymbol{u}$ (joint velocities or acceleration) to the different task spaces $\mathcal{T}_{\boldsymbol{u},p}:\mathbb{R}^{d_{\boldsymbol{u}}}\rightarrow\mathbb{R}^{k_{\boldsymbol{u}}^p}$. In case of torques, the inverse of this transformation is defined $\mathcal{T}_{\boldsymbol{u},p}^{-1}:\mathbb{R}^{k_{\boldsymbol{u}}^p}\rightarrow\mathbb{R}^{d_{\boldsymbol{u}}}$. These transformations can be parametrized; for example, the pose of the end-effector is often computed in multiple coordinate systems [16, 81, 86]. Various relevant task spaces are presented in Sec. 3.3.1.

We denote $\boldsymbol{x}_p$ the value in task space $p$ with

$$\boldsymbol{x}_p = \mathcal{T}_p(\boldsymbol{q}), \tag{2.23}$$

and $\boldsymbol{J}_p = \partial \mathcal{T}_p / \partial \boldsymbol{q}$ its Jacobian. The velocity $\dot{\boldsymbol{x}}_p$ and acceleration $\ddot{\boldsymbol{x}}_p$ in task space $p$ are given by the differential relationship as

$$\dot{\boldsymbol{x}}_p = \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{2.24}$$

$$\ddot{\boldsymbol{x}}_p = \boldsymbol{J}_p(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_p(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{2.25}$$

The relation between the joint torque $\boldsymbol{\tau}$ and the generalized force $\mathbf{f}_p$ is given as

$$\boldsymbol{J}_p^\top(\boldsymbol{q})\, \mathbf{f}_p = \boldsymbol{\tau}. \tag{2.26}$$

The desired control commands are often defined in the task spaces and the previous relations inverted to get the corresponding joint angle commands. If the manipulator is redundant, the solutions exist and are given as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_p^\dagger(\boldsymbol{q})\dot{\boldsymbol{x}}_p + (\boldsymbol{I} - \boldsymbol{J}_p^\dagger(\boldsymbol{q})\boldsymbol{J}_p(\boldsymbol{q}))\, \dot{\boldsymbol{x}}_{\bar{p}}, \tag{2.27}$$

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_p^\dagger(\boldsymbol{q})(\ddot{\boldsymbol{x}}_p - \dot{\boldsymbol{J}}_p(\boldsymbol{q})\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}_p^\dagger(\boldsymbol{q})\boldsymbol{J}_p(\boldsymbol{q}))\, \ddot{\boldsymbol{x}}_{\bar{p}}, \tag{2.28}$$

where $\dot{\boldsymbol{x}}_{\bar{p}}$ and $\ddot{\boldsymbol{x}}_{\bar{p}}$ are additional commands coming from the resolution of other objectives. The matrix $(\boldsymbol{I} - \boldsymbol{J}_p^\dagger(\boldsymbol{q})\boldsymbol{J}_p(\boldsymbol{q}))$ projects the additional commands onto the nullspace of $\boldsymbol{J}_p$, making sure that these commands do not interfere with the primary one.

The equation of motions of a manipulator is given as

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{2.29}$$

where $\mathbf{M}(\boldsymbol{q})$ is the inertia matrix, $\mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the Coriolis vector and $\mathbf{g}(\boldsymbol{q})$ the gravity vector. Using inverse dynamics equations, the torque vector achieving the desired joint angle accelerations can be computed. To compensate for perturbations and imperfect models, feedback controllers are commonly used [84].

# 3

# Product of task-space experts

This chapter addresses the first part of the research question: how to preserve and exploit the information about the structure of the kinematic chain when learning distributions in several task spaces? The solution proposed in this thesis is to learn distributions of robot configurations $p(\boldsymbol{q})$ as a combination of distributions defined in several task spaces. To achieve that, product of experts (PoE) are proposed as a consistent framework to learn $p(\boldsymbol{q})$. In order to provide adaptation to external parameters $\boldsymbol{s}$, such as position and orientation of objects, we also consider conditional distributions $p(\boldsymbol{q}|\boldsymbol{s})$.

As acquiring data by manipulating the robot is often costly, we focus on problems where only small datasets are provided and in which generalization capabilities with respect to the external parameters $\boldsymbol{s}$ are important. In contrast, many of the current works in machine learning rely on big datasets. It enables complex distributions to be learned with little or no prior knowledge about the structure of the data. Our approach aims to exploit at best the existing robotic knowledge, while keeping the possibility to learn more complex distributions. Following Occam's razor principle, our approach aims to find simple explanations for complex distributions. We show that simpler explanations not only lead to more interpretable models but also increase the generalization capabilities and reduce the need for data.

In robotics, these explanations often correspond to task spaces providing distributions of simpler shapes. For example, a Gaussian distribution of the end-effector might result in a very complex distribution of joint angle configurations, as shown in Fig. 3-2. The configuration $\boldsymbol{q}$ is often not of primary interest; poses in different task spaces, distances to objects, pointing directions or bimanual correlations are often more important. Hence, many approaches in LfD only learn distributions of these transformed quantities. These distributions are often learned independently and combined only at the control level. In this chapter, we show that this approach has several drawbacks. First, it is unaware of the kinematic structure of the robot

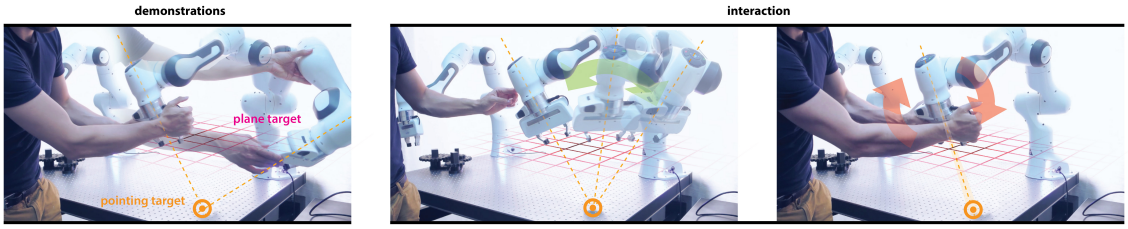



**Figure 3-1:** Product of experts can be used to set up virtual guides on robotic manipulators. A user demonstrates kinesthetically several configurations fulfilling the task objectives (*left*). In this example, the end-effector of the robot should stay close to a horizontal plane, while pointing to a desired target. The objectives are then inferred by the robot as target distributions under several transformations (task spaces). By exploiting the torque control capabilities of recent robots, the different objectives are tracked. Using an optimal control strategy, feedback gains can be computed according to the precision of the different objectives. As a resulting behavior, the robot is free to move along directions that are not constrained by the objectives (*center*), while preventing deviations from the objectives (*right*).

and of the limited range of values that can be reached within each task space. It results in a confusion between the characteristics of the task and the capabilities of the robot. Secondly, when distributions under several task spaces are learned independently, the relation between them is ignored. It then becomes impossible to properly understand each task and the required precision. Moreover, when some tasks are prioritized, secondary objectives can only be recovered if the dependencies between the task spaces are considered. They are indeed masked by the resolution of the tasks of higher importance.

The main contribution of this chapter is to apply the products of experts (PoE) approach in [51] to robotics. This approach can combine the interpretability, compactness and precision of task-space distributions with the kinematic awareness of the configuration space. Particularly, the main detailed contributions are:

1. **Training PoEs (Section 3.1)** an approach to train PoEs using variational inference, better suited to the targeted robotic applications than the original approach from [52].
2. **PoE with prioritizations (Section 3.2)** a novel technique that leverages the PoE formulation in combination with null space operators to learn task priorities from demonstrations with minimal prior knowledge compared to the state-of-the-art (e.g. no need to know task references a priori, recover secondary masked tasks).
3. **New perspectives for PoEs in robotics (Section 3.3, 3.4)** an extensive formulation of experts describing various relevant manipulation skills in robotics (e.g. position/orientation/joint space/manipulability, movement primitives, ergodic control, prioritization) whose fusion can be harmoniously learned using our approach.

4. **Experiments (Section 3.5)** a detailed analysis of the proposed approach, highlighting the capabilities of learning from few data as well as the ability to learn an arbitrary number of prioritized tasks.

**Organization of the chapter** In Sec. 3.1.3, PoEs are formally presented and the practical implications of the normalization constant for robotics are discussed. In the second part of the section, a method to train PoEs using variational inference is proposed. In Sec. 3.2, we propose an extension of PoEs with nullspace filter (PoENS). Classical nullspace approaches for inverse kinematic are first presented. Then, this approach is used to redefine the derivative of the log-likelihood of the PoE such as to induce a hierarchy. Several distributions and task spaces are presented in Sec. 3.3. They help the practitioner to tackle a wide range of robotic problems. In Sec. 3.4, two different control strategies compatible with PoEs are then presented (Fig. 3-1 illustrates one of them). Finally, in Sec. 3.5, several experiments are presented to compare the proposed models with other density estimation techniques.

## 3.1 Product of task-space experts

Products of experts, proposed in [51], are models in which several densities $p_m$ (which are called experts) are multiplied together and the product renormalized. Each expert can be defined on a different view or transformation of the data $\mathcal{T}_m(\boldsymbol{q})$, and the resulting density expressed as

$$p(\boldsymbol{q}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M) = \frac{\prod_m p_m(\mathcal{T}_m(\boldsymbol{q}) \,|\boldsymbol{\theta}_m)}{\int_z \prod_m p_m(\mathcal{T}_m(\boldsymbol{z}) \,|\boldsymbol{\theta}_m)}. \tag{3.1}$$

For compactness, we will later refer to the unnormalized product as

$$\tilde{p}(\boldsymbol{q}) = \prod_m p_m(\mathcal{T}_m(\boldsymbol{q}) \,|\boldsymbol{\theta}_m), \tag{3.2}$$

where we drop the parameters of the experts $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M$ in the notation. In robotics, the transformations $\mathcal{T}_m(\boldsymbol{q})$ correspond to different task spaces. They can either be given, such as the forward kinematics of a known manipulator, or parametrized (subject to estimation). Several task spaces that can be used in robotic problems are presented in Sec. 3.3.1.

As an example, $\boldsymbol{q}$ can be the configuration of a humanoid (joint angles and a floating base 6-DoF transformation). The different task spaces can consist of the forward kinematics of several links, like the feet and the hands. The densities of the corresponding experts can be multivariate Gaussian distributions, defining where these links should be located.

This way of combining distributions is very different from more traditional mixture models. In mixture models, each distribution is in configuration space, which can be of high dimensions ($> 30$) for humanoids. Many components and a laborious tuning are necessary to cover this space. Often, the distribution is also very sharp, creating

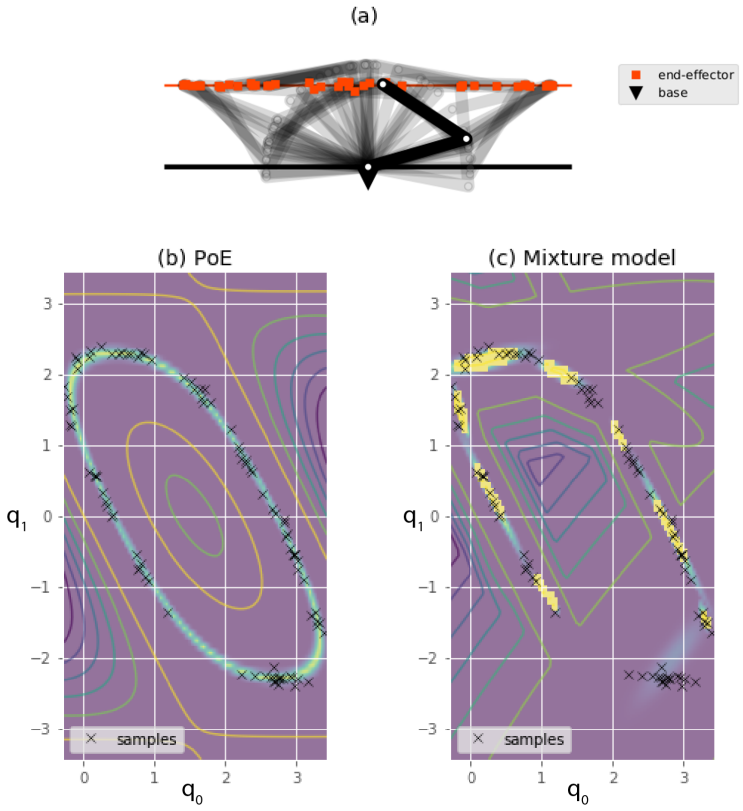



**Figure 3-2:** Product of experts can learn complex distributions of joint angles as a fusion of simple distributions within several task spaces. *(a)* 30 samples of the configuration of a 2-DoF planar robot are displayed. The end-effector follows an elongated Gaussian distribution. The distribution in configuration space has a more complex shape, which it is difficult to represent. *(b)* With our approach, we learn configuration-space distributions as a product of simple densities in different task spaces. It is thus possible to represent this sharp distribution easily with few parameters. *(c)* A standard approach like a Gaussian mixture model, directly applied in configuration space, needs much more samples and parameters to represent this distribution. Following Occam's razor principle, finding simpler explanations (fewer parameters of the model) to understand complex dataset leads to better generalization.

a low-dimensional manifold in a higher-dimensional space, as illustrated in Fig. 3-2. These sharp distributions are hard to be represented with mixtures, even in low-dimensional spaces. In PoEs, each expert can constrain a subset of the dimensions or a particular transformation of the configuration space. The product is the intersection of all these constraints, which can be very sharp. It has far fewer parameters to train than a mixture. A PoE density is also smoother than a mixture, which is by definition multimodal. This feature is particularly beneficial for applications in control, as it will be presented in Sec. 3.4. Moreover, if the model needs to encode time-varying configurations, it can easily be extended to a mixture of PoEs. Great flexibility is then allowed to set up such model; some experts can be shared among mixture components to encode global constraints (preferred posture, static equilibrium), while some others can have different parameters for each mixture component.

### 3.1.1 Importance and implications of the normalization constant

The renormalization is an important aspect that distinguishes this approach from the ones in which the models are learned independently. In these approaches, the recorded data is first transformed into the quantities of interest $\mathcal{T}_m(\boldsymbol{q})$, or directly recorded in this form. Then, either the different quantities are stacked to learn one model, or different models on the different quantities are trained separately. In the latter case, the models are combined at the control level. This approach corresponds to a PoE in which the normalization constant has been dropped. The renormalization might seem to be a mathematical preoccupation, but it has several practical implications.

- Renormalizing the product allows some experts to give constant probabilities to all configurations. If the normalization was done experts-wise, this would imply close to zero probability everywhere, and result in a very low likelihood. Experts which do not influence the product can be dropped without penalizing the overall likelihood. Hence, explaining the data with sparse models (a small number of experts) is possible. It often leads to better generalization. In Sec. 3.5.1, an experiment is presented in which the model should distinguish between a configuration-space or a task-space target distribution.

- When using a mixture of PoEs, as in [16], not considering the renormalization further reduces generalizations capabilities. It prevents mixture components to identify patterns appearing only in a subset of the task spaces, which would have been sufficient to explain the full configuration. Instead, the mixture components are allocated such that the representation is compact under each transformation, preventing good generalization capability.

- The proper support (set of possible values that $\mathcal{T}_m(\boldsymbol{q})$ can take) of each expert distribution is taken into account. Renormalizing on $\boldsymbol{q}$ makes sure that the support considered by each expert is defined by the set of possible configurations and its transformation in the different task spaces. For example, let us consider

an expert as a non-degenerate multivariate Gaussian defining the position of the end-effector of a fixed manipulator. The support of a Gaussian defining a task-space objective is normally $\mathbb{R}^3$. By using a PoE, which takes into account $\boldsymbol{q}$ and the forward kinematics transformation, the support turns into the actual workspace of the robot.

When training the model, it means that the high probability regions of each expert are not necessarily where the data is. With a proper support, the model is trained such that the transformed data is in a region of higher probability than the remaining values that it can take. Practically, it means that the model can clearly distinguish between the targeted task, represented by the PoE, and the kinematic capability of the robot. The importance of the support is best noticed when transferring models between robots with different kinematic chains, or when tasks should be uncovered and grasped even if their realization is prevented by kinematic constraints.

- The realization of tasks is sometimes prevented by complementary or competitive objectives. In Sec. 3.2, an extension to PoE is presented to admit strict hierarchies between the tasks. As with kinematic constraints, considering the proper support and renormalization allows recovering the masked tasks. This possibility will be illustrated by several experiments in Sec. 3.5.2, also shown in Fig. 3-11.

- The use of unnormalized expert densities is enabled. Especially for orientation statistics, some interesting distributions have intractable normalizing constant, which sometimes dissuade their use. In a PoE, these distributions can be used with no overhead, as the normalization only occurs at the level of the product. Also, the normalization constant is typically easier to compute on joint angles than on orientation manifolds. An experiment is presented in Sec. 3.5.4 in which a joint distribution of positions and rotation matrices is learned.

When the experts are learned independently, nothing ensures that the PoE matches the data distribution. It only becomes similar in some particular cases. For example, when the demonstrated data has an important variance under all-but-one task spaces. Moreover, no kinematic constraint should prevent the resolution of the task (e.g., a task-space target distribution with all its mass inside the robot workspace).

### 3.1.2 Estimating PoEs

In the general case, the renormalized product $p(\boldsymbol{q})$ has no closed-form expression. A notable exception is Gaussian experts with linear transformations. In our case, this is of limited interest, due to the nonlinearities of the task spaces we are considering.

Approximation methods are required to work with PoEs. An adequate approximation method should enable several elements: we should be able to draw samples from it, to estimate the normalizing constant, to estimate its gradient with respect to the model parameters, and to identify the modes of the distribution. Such methods

are found in Bayesian statistics and were reviewed in Sec. 2.3. In Bayesian statistics, the same problem of approximating an unnormalized density occurs. Indeed, the posterior distribution of model parameters $\boldsymbol{\theta}$ can be viewed as a product of two experts: a likelihood and a prior.

### 3.1.3 Training PoEs

From a given dataset of robot configurations $\boldsymbol{Q}$, maximum likelihood (or maximum a posteriori) of the intractable distribution $p(\boldsymbol{q}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)$ should be computed.

It can be done using gradient descent, as proposed in [51]. The derivative of the log-likelihood of the PoE can be separated into the derivative of the unnormalized expert and the normalizing constant

$$\frac{\partial \log p(\boldsymbol{q}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)}{\partial \boldsymbol{\theta_m}} = \frac{\partial \log p_m(\boldsymbol{q}|\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta_m}} - \frac{\partial \log \mathcal{C}(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)}{\partial \boldsymbol{\theta}_m}. \tag{3.3}$$

The derivative of the normalizing constant is intractable and requires approximation methods. It can be written as

$$\frac{\partial \log \mathcal{C}(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)}{\partial \boldsymbol{\theta}_m} = \int_c p(\boldsymbol{c}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M) \frac{\partial \log p_m(\boldsymbol{c}|\boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta_m}} \, d\boldsymbol{c}, \tag{3.4}$$

which is the expected derivative of the unnormalized expert log-likelihood under the current PoE distribution. The averaged derivatives over the dataset $\boldsymbol{Q}$ and with respect to the parameters of all experts $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_1^\top, \dots, \boldsymbol{\theta}_M^\top\right]^\top$ can be written as

$$\left\langle \frac{\partial \log p(\boldsymbol{q})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{Q}} = \left\langle \frac{\partial \log \tilde{p}(\boldsymbol{q})}{\partial \boldsymbol{\theta}} \right\rangle_{\boldsymbol{Q}} - \left\langle \frac{\partial \log \tilde{p}(\boldsymbol{q})}{\partial \boldsymbol{\theta}} \right\rangle_{p(\boldsymbol{q})}, \tag{3.5}$$

where $< \cdot >_{p(\cdot)}$ denotes the expectation over the distribution $p$. Intuitively, it means that we compare the expected gradient of the unnormalized density over the dataset $\boldsymbol{Q}$ with the expected gradient over the current density $p(\boldsymbol{q})$. At convergence, the expected difference should be zero; it means that the distribution of the PoE $p(\boldsymbol{q})$ matches the data distribution $\boldsymbol{Q}$. Maximizing the log-likelihood of the data is also equivalent to minimizing the KL divergence between the data distribution and the equilibrium distribution.

In all but a few cases, the expectation under the current PoE density has no closed-form. Worse, estimating this integral with samples is not trivial since drawing samples from the current PoE is difficult. In [52], it is proposed to use a few sampling steps initialized at the data distribution $\boldsymbol{Q}$. With only a few sampling steps, these Markov chains cannot reached their equilibrium distribution (the PoE) but still provide a biased estimation of the gradient. Unfortunately, this approach fails when $p(\boldsymbol{q}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)$ has multiple modes. The few sampling steps never jump between modes, resulting in the incapacity of estimating their relative mass. Wormholes have been proposed as a solution in [119], but are algorithmically restrictive.

As an alternative, we propose to use VI with the proposed mixture distribution

to approximate $p(\boldsymbol{q})$. Throughout the training process, this approximation is kept updated to be able to draw samples from the PoE. The process thus alternates between minimizing $D_{\mathrm{KL}}(\tilde{q}||p)$ with current $p$ and using current $\tilde{q}$ to compute the gradient (3.3). The approximate distribution $\tilde{q}$ can either be used as an importance sampling distribution or directly (if expressive enough to represent $p$). The expected gradient over the current density $p(\boldsymbol{q})$ becomes

$$\left\langle \frac{\partial \log \tilde{p}(\boldsymbol{q})}{\partial \boldsymbol{\theta}} \right\rangle_{p(\boldsymbol{q})} \approx \sum_{n=1}^{N} w_n \frac{\partial \log \tilde{p}(\boldsymbol{q}^{(n)})}{\partial \boldsymbol{\theta}} \Big/ \sum_{n=1}^{N} w_n, \tag{3.6}$$

$$\text{with} \quad \boldsymbol{q}^{(n)} \sim \tilde{q}(\,\cdot\,|\boldsymbol{\lambda}) \quad \text{and} \quad w_n = \frac{\tilde{p}(\boldsymbol{q}^{(n)})}{\tilde{q}(\boldsymbol{q}^{(n)}|\boldsymbol{\lambda})}, \tag{3.7}$$

where $w_n$ are the importance weights.

It is also possible to estimate this expectation by using a mixture of samples from the variational distribution $q$ and from Markov chains initialized on the data. It combines the advantages of the two methods. Samples from $q$ are necessary to estimate the relative mass of distant modes, while the few steps of the Markov chain reduce the variance of the derivative. Empirically, we noticed that they tend to stabilize the learning procedure.

In the derivation, only the experts parameters $\boldsymbol{\theta} = \left[\boldsymbol{\theta}_1^\top, \dots, \boldsymbol{\theta}_M^\top\right]^\top$ were trained. It is also possible to train the parameters of their associated transformation without any modification to the procedure.

**Initializing PoEs** As the training procedure is iterative, a good initialization of $\boldsymbol{\theta}_m$ speeds up the learning process. For the experts with simple forms of maximum likelihood estimation (MLE) and known transformation $\mathcal{T}^m$, we propose independent initializations with maximum likelihood. As detailed in the related work in Sec. 2.1.1, many works in robotics train the model independently. Thus, we propose to initialize the PoE that way and improve it by considering the proper renormalization.

## 3.2 Product of experts with nullspace (PoENS)

We often encounter tasks where some objectives are more important than others. The problem of controlling a robot when the different objectives and their hierarchy are known has already been addressed in several works, see e.g. [83]. It is usually done by filtering commands minimizing secondary objectives with nullspace projection operators. It ensures that these commands stay within the subspace of commands minimizing higher level objectives.

In our work, considering a hierarchy has an additional interest. The realization of some subtasks might be prevented and masked by the resolution of higher-level subtasks. Thus, it is not trivial to identify secondary objectives in the dataset. Ignoring their lower priority when training the model will lead to problems (at best, minimizing their importance, and at worst, completely missing them). These tasks

can be understood only by considering their lower priority and their entanglement with higher priority subtasks. The priority awareness was already considered in other works, but focused on learning control policies [76, 78]. The task to consider the priority at the level of distributions is computationally more complex and has not been proposed before to our knowledge. In this section, we propose to extend the PoE framework to the use of nullspace filters.

Illustratively, the idea is to define a hierarchy between the experts such that secondary experts can express their opinions only in subspaces that primary experts do not care about. For example, let us consider that $\boldsymbol{q} \in \mathbb{R}^7$ are the joint angles of a 7-DoF manipulator and $p_1(\mathcal{T}_1(\boldsymbol{q}))$ is a Gaussian distribution of the end-effector position. The system has still 4 DoFs in which secondary experts can express their opinion.

Our approach consists of redefining the derivative of the log-likelihood of the PoE with a nullspace filter. This filter cancels the derivative of secondary experts in the space of primary experts, making sure that secondary experts have no power in the area of expertise of primary experts. Each expert acts on a transformation of the configuration $\boldsymbol{q}$

$$\boldsymbol{y}_m = \mathcal{T}_m(\boldsymbol{q}), \tag{3.8}$$

and have the differential relationship

$$\dot{\boldsymbol{y}}_m = \boldsymbol{J}_m(\boldsymbol{q})\, \dot{\boldsymbol{q}}, \tag{3.9}$$

where $\boldsymbol{J}_m(\boldsymbol{q}) = \frac{\partial \mathcal{T}_m(\boldsymbol{q})}{\partial \boldsymbol{q}}$ is the Jacobian matrix of the task space $\mathcal{T}_m$. When computing inverse kinematics with a priority, as in [83], the general solution of (3.9) is

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_m^{\dagger}(\boldsymbol{q})\ \dot{\boldsymbol{y}}_m + \big(\boldsymbol{I} - \boldsymbol{J}_m^{\dagger}(\boldsymbol{q})\boldsymbol{J}_m(\boldsymbol{q})\big)\boldsymbol{z}, \tag{3.10}$$

where $\boldsymbol{J}_m^{\dagger}$ is the pseudoinverse of $\boldsymbol{J}_m$ and $\boldsymbol{z}$ is an arbitrary vector. The nullspace filter $\boldsymbol{N}_m(\boldsymbol{q}) = \boldsymbol{I} - \boldsymbol{J}_m^{\dagger}(\boldsymbol{q})\boldsymbol{J}_m(\boldsymbol{q})$ makes sure that the arbitrary vector has no effect in the task space $\mathcal{T}_m$ as

$$\boldsymbol{J}_m(\boldsymbol{q})\big(\boldsymbol{I} - \boldsymbol{J}_m^{\dagger}(\boldsymbol{q})\boldsymbol{J}_m(\boldsymbol{q})\big)\, \boldsymbol{z} = \big(\boldsymbol{J}_m(\boldsymbol{q}) - \boldsymbol{J}_m(\boldsymbol{q})\big)\, \boldsymbol{z} = \boldsymbol{0}.$$

By applying the chain rule, the derivative of the log-likelihood with respect to the configuration $\boldsymbol{q}$ becomes

$$\frac{\partial \log \tilde{p}(\boldsymbol{q}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)}{\partial \boldsymbol{q}} = \sum_m \frac{\partial \log p(\mathcal{T}_m(\boldsymbol{q})|\boldsymbol{\theta}_m)}{\partial \boldsymbol{q}}, \tag{3.11}$$

$$= \sum_m \frac{\partial \log p(\boldsymbol{y}|\boldsymbol{\theta}_m)}{\partial \boldsymbol{y}} \boldsymbol{J}_m(\boldsymbol{q}). \tag{3.12}$$

The nullspace filter $N_m(\boldsymbol{q})$ is used to filter the derivative coming from other experts such that they do not affect the space where the expert $m$ is acting. For example, if

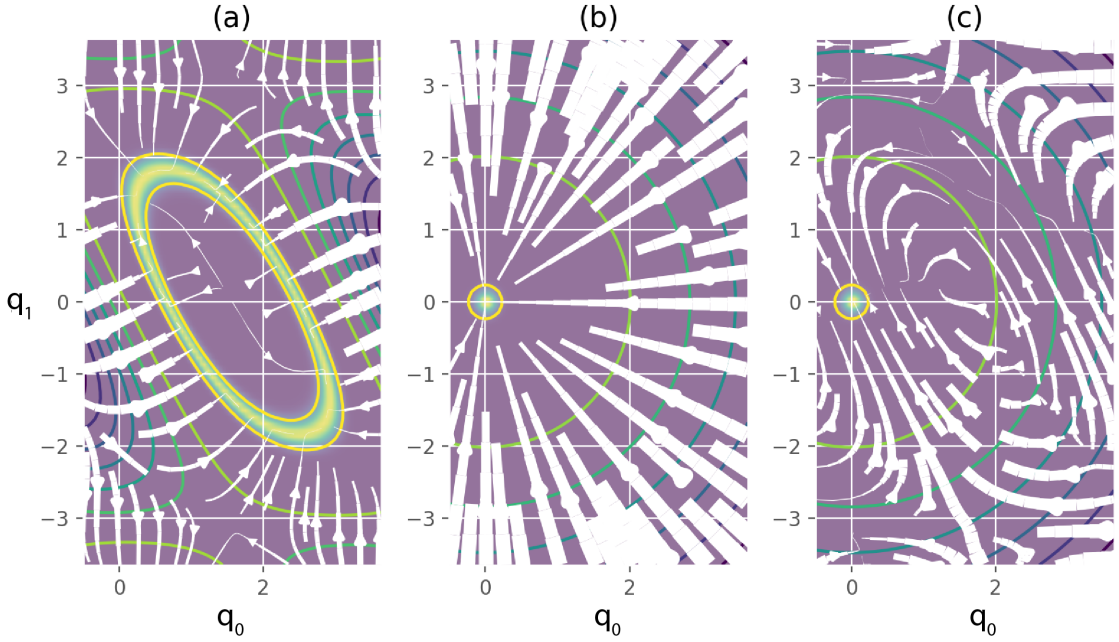



**Figure 3-3:** Densities and derivatives of a two-expert problem. A 2-DoF planar robot is considered as in Fig. 3-2. The first expert is an elongated Gaussian distribution on its end-effector. The second expert acts in joint angles and defines a preferred configuration. *(a)* The density of the expert $\log p_1(\mathcal{T}_1(\boldsymbol{q})|\boldsymbol{\theta}_1)$ is shown as a colormap. The derivative $\partial \log p(\boldsymbol{y}|\boldsymbol{\theta}_1)/\partial \boldsymbol{y}\, \boldsymbol{J}_1(\boldsymbol{q})$ is displayed as streamlines. *(b)* The same is done with the second objective in configuration space. *(c)* The derivative of the second experts is this time filtered as $\partial \log p(\boldsymbol{y}|\boldsymbol{\theta}_2)/\partial \boldsymbol{y}\, \boldsymbol{J}_2(\boldsymbol{q})\boldsymbol{N}_1(\boldsymbol{q})$.

we have two experts, with expert $m = 1$ the primary the derivative becomes

$$\frac{\partial \log p(\mathcal{T}_m(\boldsymbol{q})|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)}{\partial \boldsymbol{q}} = \frac{\partial \log p(\boldsymbol{y}|\boldsymbol{\theta}_1)}{\partial \boldsymbol{y}} \boldsymbol{J}_1(\boldsymbol{q}) + \frac{\partial \log p(\boldsymbol{y}|\boldsymbol{\theta}_2)}{\partial \boldsymbol{y}} \boldsymbol{J}_2(\boldsymbol{q}) \boldsymbol{N}_1(\boldsymbol{q}). \tag{3.13}$$

Fig. 3-3 provides an example where the terms of this derivative are displayed with and without the nullspace filter for the secondary objectives. We note that when using automatic differentiation libraries such as TensorFlow [1], gradients can be easily redefined with this filter.

While in standard PoEs, it was possible to evaluate the unnormalized log-likelihood, the PoENS is defined only by the gradient of this quantity. It is thus not possible to evaluate the unnormalized log-likelihood $\tilde{p}$. It constrains the class of methods to approximate the density. Stochastic variational inference can be employed, as it only requires stochastic evaluation of the gradient. This characteristic is shared with only a very few Monte Carlo methods such as [20]. In this mini-batch variant of Hamiltonian Monte Carlo method [29], no corrective Metropolis-Hastings steps are used as they are too costly.

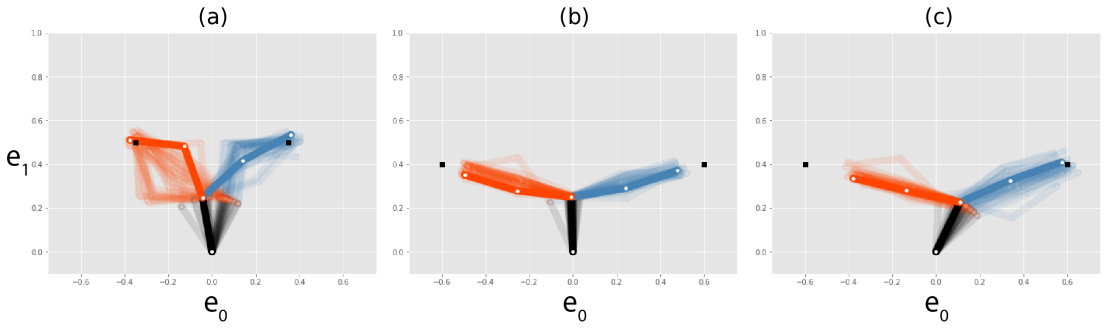



**Figure 3-4:** 5-DoF bimanual planar robot with two forward kinematics objectives. Variational inference is used to generate the displayed samples. *(a)* The two tasks are compatible and the distribution of solution is approximated. *(b)* No nullspace, the two tasks are of the same importance. *(c)* The gradient of the objective of the orange arm is projected onto the nullspace of the Jacobian of the forward kinematics of the blue arm. This is achieved with stochastic variational inference that only requires to evaluate the gradient of the unnormalized density.

Fig. 3-4 shows a 5-DoF bimanual planar robot with two forward kinematics objectives. When the tasks are compatible, the filtering does not affect the solutions. The gradient of the objective of the orange arm can be projected onto the nullspace of the Jacobian of the forward kinematics of the blue arm, resulting in a prioritization.

## 3.3 Useful distributions and task spaces for robotics

In this section, several task spaces $\mathcal{T}^m$ and experts models $p_m$ related to common robotic problems are presented with a practitioner perspective. This can be used as a toolkit to unify various problems into the PoE framework.

### 3.3.1 Task spaces

Several common task spaces are presented. These transformations can be known and fixed, as the forward kinematics of the end-effector. They can also be partially known, for example the position of an object held by the known end-effector, which constitutes a new end-effector. Fully unknown transformations can also be learned with neural networks.

**Forward kinematics (FK)** One of the most common transformations used in robotics is forward kinematics, computing poses (position and orientation) of links given the robot configuration $\boldsymbol{q}$. Forward kinematics can be computed in several task spaces associated with objects of interest, as in [16, 81, 86]. In these works, analyzing movements from several coordinate systems allows for generalizations with respect to movements of their associated objects. These works only consider cases where the transformation is fully known. The approaches in these works, based on separately

transforming the data and learning the models, are not compatible with partially known transformations, as opposed to ours. In particular, two types of unknowns can be considered with PoE: unknown coordinate systems or free parameters in the kinematic chain. These two cases will be considered in the experiments of Sec. 3.5.3.

In the first case, the robot could, for example, have to track an object that can move. We could have a dataset split into several subparts in which the pose of this object is constant. The unknown displacement of the objects can be subject to optimization, as well as the parameters of the distribution representing the target within its associated coordinate system.

The second case can occur if a new end-effector is added. For example, it can be a tool grasped by the robot, whose position is given by

$$\mathcal{T}_m(\boldsymbol{q}) = \boldsymbol{F}_R(\boldsymbol{q})\boldsymbol{d} + \boldsymbol{F}_x(\boldsymbol{q}), \tag{3.14}$$

where $\boldsymbol{F}_x(\boldsymbol{q})$ and $\boldsymbol{F}_R(\boldsymbol{q})$ are respectively the position and rotation matrix of the known end-effector, and $\boldsymbol{d}$ the displacement of the tool. The parameters $\boldsymbol{d}$ can be optimized as well when training the PoE with maximum likelihood. The results is that a new end-effector is found with which the distribution of configurations $\boldsymbol{q}$ is best explained and compact.

**Manipulability** Measures of manipulability are other interesting transformations in robotics. The most simple form is a scalar defined as

$$\mathcal{T}_m(\boldsymbol{q}) = \sqrt{\det(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^\top)}, \tag{3.15}$$

which can be used to explain the avoidance of singular configurations in the dataset, as in [61].

For a more precise description of the configuration, velocity and force ellipsoids can be used [123]. The velocity ellipsoid is defined by the matrix

$$\mathcal{T}_m(\boldsymbol{q}) = \left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^\top\right)^{-1}, \tag{3.16}$$

and the force by its inverse. If we consider a zero-centered and unit-covariance Gaussian distribution of joint velocity, the velocity ellipsoid corresponds to the precision matrix (inverse of the covariance) of the task-space velocities. It thus relates to the feasible distribution of task-space velocities (respectively forces). For example, this full matrix can be used to define the distribution of configurations in which the transfer of velocities or forces along a direction should be maximized.

Such matrix is positive semi-definite, which should be taken into account for the choice of the associated expert distribution. An option is to work with its Cholesky decomposition. A Wishart distribution is not expressive enough to define variances along different directions. In [58], it is proposed to use Gaussian distributions in the tangent space of manifolds. This choice is motivated by the geometry of symmetric positive definite matrices (SPD). However, this approach ignores the geometry of the robot (the manipulability is a function of the joint angles and the kinematic struc-

ture). This comes with two important limitations. First, the proper support of the manipulability ellipsoid is ignored. By considering a distribution on the SPD manifold, it is assumed that this space can be covered by the robot, while the robot might only cover a subspace of it (which can also vary among robots). A second limitation is that manipulability is often employed as secondary objective. For example, when playing golf, hitting the ball at the right place is of higher importance than replicating a desired manipulability ellipsoid. The subspace of SPD matrices is further limited when it lies in the nullspace of more important tasks. A simple experiment with a manipulability measure will be presented in Sec. 3.5.2 to show that this should be considered to learn the targeted manipulability.

Similarly, such consideration is important to transfer manipulability objectives between robots with different kinematic chains. In [58], manipulability ellipsoid distributions are expressed on the set of symmetric positive definite matrices. We will show in the experiments of Sec. 3.5.2 that manipulability-related tasks would be better transferred by considering that the distributions are expressed on subsets of these matrices. These experiments will motivate that a better support of the distributions can be considered by taking into account the differences in the kinematic chain configurations.

**Relative distances** A *relative distance space* is proposed in [122]. It computes the distances from multiple virtual points on the robot to other objects of interest (targets, obstacles). It can, for example, be used in environments with obstacles, providing an alternative or complementing standard forward kinematics.

**Center of Mass (CoM)** From the forward kinematics of the center of mass of each link and their mass, it is possible to compute the center of mass (CoM) of the robot. When considering mobile or legged robots, the CoM should typically be located on top of support polygons to satisfy static equilibrium on flat surfaces.

**Jacobian pseudoinverse iterations** The following transformation is more interesting for the problem of sampling configurations from a given PoE than for the one of maximum likelihood from a dataset. Precise kinematics constraints imply a very correlated $\tilde{p}(\boldsymbol{q})$, as shown in Fig. 3-2. In the extreme case of hard kinematics constraints, the solutions are on a low dimensional manifold embedded in configuration space. With most of the existing methods, it is very difficult to sample from this correlated PoE and to approximate it. Dedicated methods address the problem of representing a manifold as [117] or sampling from it, as [127]. In [10], a projection strategy is proposed. Configurations are sampled randomly and projected back using an iterative process. We propose a similar approach where the projection operator $\mathcal{P}_N$ would be used as transformation $\mathcal{T}^m$. Inverse kinematics problems are typically solved iteratively with

$$\mathcal{P}(\boldsymbol{q}) = \boldsymbol{q} + J(\boldsymbol{q})^{\dagger}\Big(\hat{\boldsymbol{p}} - \boldsymbol{F}(\boldsymbol{q})\Big), \tag{3.17}$$

where $\hat{\boldsymbol{p}}$ is the target and $J(\boldsymbol{q})^\dagger$ is the Moore-Penrose pseudo-inverse of the Jacobian. This relation is derivable and can be applied recursively with

$$\mathcal{P}_0(\boldsymbol{q}) = \mathcal{P}(\boldsymbol{q}), \tag{3.18}$$

$$\mathcal{P}_{n+1}(\boldsymbol{q}) = \mathcal{P}\Big(\mathcal{P}_{n-1}(\boldsymbol{q})\Big). \tag{3.19}$$

Then, the distribution

$$p_m(\boldsymbol{q}) \propto \mathcal{N}\Big(\mathcal{P}_N(\boldsymbol{q})\Big|\, \hat{\boldsymbol{p}}, \sigma \boldsymbol{I}\Big), \tag{3.20}$$

is the distribution of configurations which converges in $N$ steps to $\mathcal{N}(\hat{\boldsymbol{p}}, \sigma\boldsymbol{I})$. Thanks to the very good convergence of the iterative process (3.17), $\sigma$ can be set very small. However, this approach has a similar (but less critical) problem as in [10]. The resulting distribution is slightly biased toward regions where the forward kinematics is close to linear (constant Jacobian), which are those where more mass converges to the manifold.

With high DoFs robots, it might be computationally expensive to run iteration steps inside the stochastic gradient optimization and propagate the gradient. Another approach would be to define heuristically (or learn) $\boldsymbol{\Sigma}_h$ such that $\mathcal{N}(\boldsymbol{q}|\, \hat{\boldsymbol{p}}, \sigma\boldsymbol{I} + \boldsymbol{\Sigma}_h)$ is close to $\mathcal{N}(\mathcal{P}_N(\boldsymbol{q})|\, \hat{\boldsymbol{p}}, \sigma\boldsymbol{I})$.

**Neural network** When more data are available, more general and complex task spaces can be considered, such as neural networks. For example, it can be used to define a complex distribution in which the end-effector should be. Coupled with a control strategy (see Sec.3.4), it can provide virtual guides as in [98] to constraint the robot on a trajectory or within a shape, as illustrated in Fig. 3-5.

Particularly, we recommend using invertible neural networks for practical reasons of initializing the training procedure and for its interesting properties of global maximum when controlling the robot. Training a PoE with contrastive divergence is not as efficient as training models with tractable likelihood. Therefore, we propose to initialize the PoE by training all the experts that have tractable likelihood independently. Thus, we propose to treat an expert composed of a neural network transformation as a distribution with a change of variable. This provides a tractable likelihood as in [23], that can be used for initializing. Given that $\mathcal{T}_m$ is bijective, the tractable likelihood is given by the change of variable

$$p(\boldsymbol{q}) = p_m\big(\mathcal{T}_m(\boldsymbol{q})\big) \left| \det \left( \frac{\partial \mathcal{T}_m(\boldsymbol{q})}{\partial \boldsymbol{q}^\top} \right) \right|, \tag{3.21}$$

where $p_m$ is a simple distribution. $p_m$ can be chosen as a zero-centered and unit covariance Gaussian. Actually, a full covariance Gaussian can be described in this framework with $\mathcal{T}_m$ being a linear transformation. If we were to train the neural network without considering the renormalization, $\mathcal{T}_m$ will try to push all the configuration of the dataset to the mode of $p_m$. The term $\left| \det \left( \partial \mathcal{T}_m(\boldsymbol{q}) / \partial \boldsymbol{q}^\top \right) \right|$ can be seen as a cost preventing the contraction around the mode.

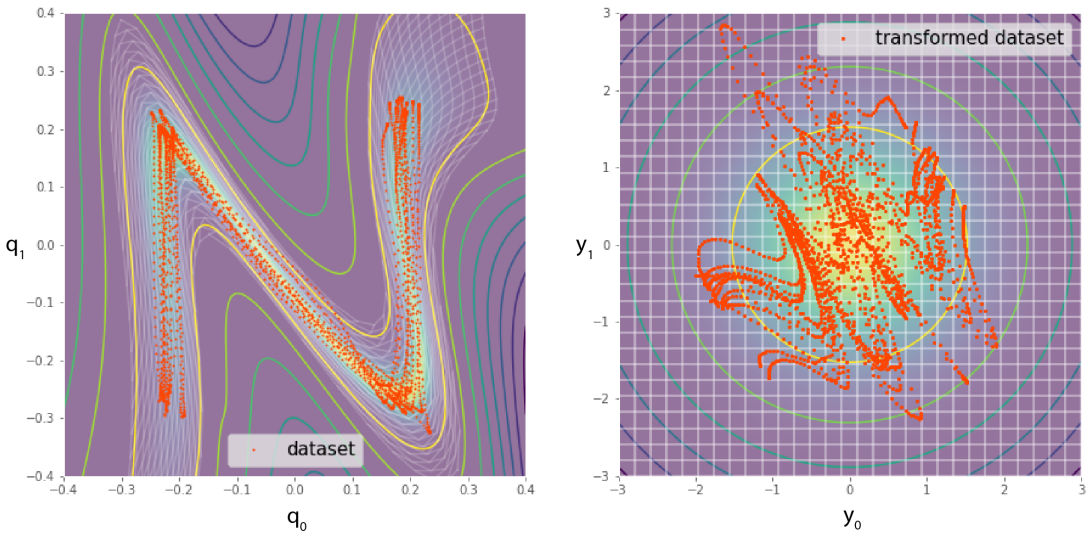



**Figure 3-5:** For greater expressiveness, an invertible neural network can be used as a task space $\mathcal{T}_m$. Instead of optimizing the parameters of the expert density $p_m$, which can be set as a zero-centered and unit covariance Gaussian (right), the parameters of the network can be trained. The neural network becomes a transformation under which the dataset (N shape, *left*) becomes a simple distribution (Gaussian, *right*). The densities are shown as a colormap with isolines. A grid that undergoes the inverse transformation is also shown. This approach allows complex attractors or guiding distributions to be learned, where it is easy to control the robot in the transformed space.

To avoid overfitting, we propose a strategy to penalize abrupt change in the transformation $\mathcal{T}_m^{-1}$. We propose to minimize the expectation of a measure of local change of the Jacobian of the inverse of the transformation $\partial\mathcal{T}_m^{-1}$ under a distribution $p_r$

$$\mathbb{E}_{p_r(\boldsymbol{y})}\left[\int_{\boldsymbol{r}} \left|\frac{\partial\mathcal{T}_m^{-1}(\boldsymbol{y})}{\partial\boldsymbol{y}^\top} - \frac{\partial\mathcal{T}_m^{-1}(\boldsymbol{y}+\alpha\boldsymbol{w})}{\partial(\boldsymbol{y}+\alpha\boldsymbol{w})^\top}\right|\boldsymbol{w}\right], \tag{3.22}$$

where $\boldsymbol{w}$ is a unit vector on $\mathcal{S}^{n_m}$, $n_m$ being the dimensionality of $\boldsymbol{y}$, and $\alpha$ a small positive scalar. The distribution $p_r$ can be chosen as a $p_m$ but with up to 3 to 10 times bigger standard deviations, which would make sure that the transformation is smooth even further from the dataset. This cost can be optimized with stochastic gradient descent by evaluating the derivative on samples of $\boldsymbol{y}^{(n)} \sim p_r(\cdot)$ and of unit vectors $\boldsymbol{w}^{(n)}$.

Another interesting property of using an invertible network is that the density under the transformation keeps a unique global maximum if the expert density $p_m$ has a single one. It is justified because invex functions (functions that have a single global minimum and no local minimum) are still invex under a diffeomorphism [96]. It is especially advantageous when we want to control the robot to track the density of the PoE, as explained in Sec. 3.4.

### 3.3.2 Distributions

Several common distributions are now presented to handle the task spaces presented in the above.

**Multivariate normal distribution (MVN)** An obvious choice for forward kinematics objectives is the Gaussian or multivariate normal distribution (MVN). Its log-likelihood is quadratic, making it compatible with standard inverse kinematics and optimal control techniques,

$$\mathcal{N}(\boldsymbol{q}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \exp\Big( -\frac{1}{2}(\boldsymbol{q}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{q}-\boldsymbol{\mu})\Big), \tag{3.23}$$

where $\boldsymbol{\mu}$ is the location parameter and $\boldsymbol{\Sigma}$ is the covariance matrix. This distribution is standard in robotics to represent full trajectories [91], subparts of trajectories with a hidden Markov model [16] or a joint distribution of a phase and robot variables with a mixture [17].

**Matrix Bingham—von Mises—Fisher distributions (BMF)** To handle orientations, for example, represented as a rotation matrix, Matrix Bingham—von Mises—Fisher distribution (BMF) [64] can be used. Its normalizing constant is intractable and requires approximation [68]. In our case, this is not a problem since we integrate over robot configurations. Its density

$$p_{\text{BMF}}(\boldsymbol{Q}|\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) \propto \exp\Big( \operatorname{tr}(\boldsymbol{C}^\top \boldsymbol{Q} + \boldsymbol{B}\,\boldsymbol{Q}^\top \boldsymbol{A}\,\boldsymbol{Q})\Big), \tag{3.24}$$

has a linear and a quadratic term as a Gaussian. $\boldsymbol{A}$ and $\boldsymbol{B}$ are often chosen as symmetric and diagonal matrices, respectively. By imposing additional constraints on $\boldsymbol{A}$ and $\boldsymbol{B}$, this density can be written as a Gaussian distribution on vectorized rotation matrices. By setting to zero the derivative of the log-likelihood

$$\frac{\partial l(\boldsymbol{Q})}{\partial \boldsymbol{Q}} = \boldsymbol{C}^T + 2\boldsymbol{A}\boldsymbol{Q}\boldsymbol{B} = 0, \tag{3.25}$$

and exploiting several properties of the trace and the Kronecker product we can rewrite the density as proportional to

$$\mathcal{N}(\operatorname{vec}(\boldsymbol{Q})|\operatorname{vec}(\boldsymbol{A}^{-1}\boldsymbol{C}^T\boldsymbol{B}^{-1}), (\boldsymbol{A}\otimes\boldsymbol{B})^{-1}), \tag{3.26}$$

$\boldsymbol{A}$ and $\boldsymbol{B}$ should be both invertible and $(\boldsymbol{A}\otimes\boldsymbol{B})$ positive definite.

In some tasks, it may be interesting to encode correlations between positions and orientations. Rewritten as a Gaussian, it is possible to create a joint distribution of position and rotation matrices

$$\mathcal{N}\left( \begin{bmatrix} \boldsymbol{q} \\ \operatorname{vec}(\boldsymbol{Q}) \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu} \\ \operatorname{vec}(\boldsymbol{A}^{-1}\boldsymbol{C}^T\boldsymbol{B}^{-1}) \end{bmatrix}^\top, \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{xx}} & \boldsymbol{\Sigma}_{\boldsymbol{xX}} \\ \boldsymbol{\Sigma}_{\boldsymbol{Xx}} & (\boldsymbol{A}\otimes\boldsymbol{B})^{-1} \end{bmatrix} \right), \tag{3.27}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{xX}} \in \mathbb{R}^{3\times 9}$ is the covariance between positions and orientations. This joint distribution can be ensured as valid by imposing a constraint on $\boldsymbol{\Sigma}_{\boldsymbol{xX}}$ with Schur complement condition for positive definiteness

$$\boldsymbol{\Sigma} \succ 0 \iff \boldsymbol{\Sigma}_{\boldsymbol{xx}} \succ 0, (\boldsymbol{A} \otimes \boldsymbol{B})^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{Xx}} \boldsymbol{\Sigma}_{\boldsymbol{xx}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{xX}} \succ 0. \tag{3.28}$$

This complex parametrization and constraints are not mandatory in the PoE framework, as the experts do not need to be valid and properly normalized distributions themselves. However, it can help to reduce the number of parameters and to stabilize the learning procedure. As an alternative approach, we present an experiment in Sec. 3.5.4 in which a joint distribution of positions and orientations is learned with a low-rank structure on the covariance.

**Matrix normal distribution** Matrix valued transformations can be encoded with a matrix normal distribution

$$p_{\text{MN}}(\boldsymbol{Q}|\boldsymbol{M}, \boldsymbol{U}, \boldsymbol{V}) \propto \exp\Big( -\frac{1}{2} \operatorname{tr}\big[\boldsymbol{V}^{-1}(\boldsymbol{Q}-\boldsymbol{M})^\top \boldsymbol{U}^{-1}(\boldsymbol{Q}-\boldsymbol{M})\big]\Big), \tag{3.29}$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{U}$ are $\boldsymbol{V}$ $n \times n$ and $p \times p$ positive definite matrices, respectively. It can also be written as a distribution of vectorized matrix as

$$\operatorname{vec}(\boldsymbol{M}) \propto \mathcal{N}\big(\operatorname{vec}(\boldsymbol{M}), (\boldsymbol{V} \otimes \boldsymbol{U})^{-1}\big), \tag{3.30}$$

where the covariance matrix has fewer parameters than in a Gaussian.

**Probabilistic movement primitives** So far, the considered distributions targeted static configurations. Probabilistic movement primitives (ProMP) is a way to build Gaussian distributions of trajectories [91].

An observation $\boldsymbol{q}_t$ (position or joint angles) follows a Gaussian distribution

$$p_{\text{ProMP}}(\boldsymbol{q}_t|\boldsymbol{\Phi}_t, \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{x}}) = \mathcal{N}(\boldsymbol{q}_t|\boldsymbol{\Psi}_t^\top \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{x}}), \tag{3.31}$$

where $\boldsymbol{w}$ is a weight vector and $\boldsymbol{\Phi}_t$ a time-dependent basis matrix. The weight vector also follows a Gaussian distribution. The marginal distribution of the observation given the parameters of the model becomes

$$p_{\text{ProMP}}(\boldsymbol{q}_t|\boldsymbol{\Phi}_t, \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{x}}) = \int \mathcal{N}(\boldsymbol{q}_t|\boldsymbol{\Psi}_t^\top \boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{x}}) \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) d\boldsymbol{w}, \tag{3.32}$$

$$= \mathcal{N}(\boldsymbol{q}_t|\boldsymbol{\Psi}_t^\top \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Psi}_t^\top \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t + \boldsymbol{\Sigma}_{\boldsymbol{x}}). \tag{3.33}$$

ProMPs are fully compatible with our framework. We can consider, for example, a product of ProMPs in multiple task spaces and configuration space. For approximating the product of ProMPs, it is also possible to use variational inference with a mixture of Gaussians, as proposed in Sec. 2.3.3. Instead of using full location and full covariance, the mixture components can be a ProMP as well. In Fig. 3-6, a mixture of ProMPs in configuration space is used to approximate a product of a ProMPs in

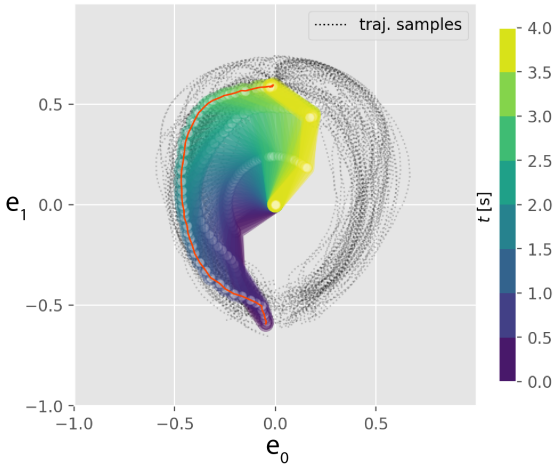



**Figure 3-6:** To encode distributions of trajectories, probabilistic movement primitives (ProMP) can be used as experts in a PoE. In this example, a product of two ProMP experts is considered: one defines a joint angle trajectory, while the other defines the trajectory of the end-effector. A 3-DoF planar robot is used in this example. The product of ProMPs is approximated using variational inference as a mixture of ProMPs in configuration space. Samples of trajectories of the end-effector are displayed, as well as a full sequence of configurations. Even if each expert is Gaussian, the product of ProMPs is multimodal because of the planar robot kinematics.

task space and in configuration space. A 3-DoF planar robot is used, which induced the multimodality of the product.

**Approximate distributions in the tangent space of manifolds** Another approach compatible with the PoE framework is to consider Gaussian distributions in the tangent space of manifolds, as a way to encode orientations [124] or manipulability ellipsoids [58]. It has the advantages of being generic to many different types of data. Also, the distribution is approximately normalized in the tangent space. It provides an easier way to initialize this expert individually (using MLE) than if it was unnormalized. The only restriction is that the logarithmic map $\text{Log}_{\boldsymbol{\mu}}(\boldsymbol{q})$, mapping elements from the manifold to the tangent space, should be differentiable. The density is given by

$$p_{\text{MVN}}(\boldsymbol{q}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \exp\Big( -\frac{1}{2}\text{Log}_{\boldsymbol{\mu}}(\boldsymbol{q})^{\top}\, \boldsymbol{\Sigma}^{-1}\, \text{Log}_{\boldsymbol{\mu}}(\boldsymbol{q})\Big). \tag{3.34}$$

**Cumulative distribution functions (CDF)** Inequality constraints, such as static equilibrium, obstacles or joint limits can be learned using cumulative distribution function

$$p(x \leq b), \quad \text{with} \quad x \sim \mathcal{N}\Big(\mathcal{T}(\boldsymbol{q}), \sigma^2\Big), \tag{3.35}$$

where $\mathcal{T}(\boldsymbol{q})$ is a scalar. For example, for half-plane constraints, $\mathcal{T}(\boldsymbol{q})$ could be $\boldsymbol{w}^\top \boldsymbol{q}$, or for joint limits on first joint $\mathcal{T}(\boldsymbol{q}) = \boldsymbol{q}_0$.

The use of the CDF makes the objectives continuous and allows safety margin determined by $\sigma$ to be considered.

Obstacles constraints might be impossible to compute exactly and require collision checking techniques [31]. With stochastic optimization, our approach is compatible with a stochastic approximation of the collision related cost, which might speed up computation substantially.

**Uni-Gauss distributions** In Sec. 3.2, we showed how a PoE is compatible with standard nullspace approaches to represent hierarchy between multiple tasks. Another way to address this problem is to use uni-Gauss experts, as proposed in [51]. These experts combine the distribution defining a non-primary objective $p_m$ with a uniform distribution

$$p_{\mathrm{UG},m}(\boldsymbol{q}) = \pi_m p_m(\boldsymbol{q}) + \frac{1 - \pi_m}{c}, \tag{3.36}$$

which means that each objective has a probability $p_m$ to be fulfilled and $1 - \pi_m$ not to be fulfilled.

Classical prioritized approaches exploit redundancies of the robot to achieve multiple tasks simultaneously [83]. A nullspace projection matrix is used such that commands required to solve a secondary task do not influence the primary task. Using uni-Gauss experts is a less strict approach that does not necessarily require redundancies. Even if there is no redundancy to solve a secondary task without influencing the first one, there may be some mass at the intersection between the different objectives.

There are two possible ways of estimating $\tilde{p}(\boldsymbol{q})$ in the case of Uni-Gauss experts. If the number of tasks is small, we can introduce, for each task $m$, a binary random variable indicating if the task is fulfilled or not. For each combination of these variables, we can then compute $\tilde{p}(\boldsymbol{q})$. The ELBO can be used to estimate the relative mass of each of these combinations, as done in model selection. For example, if the tasks are not compatible, their product would have a very small mass, as compared to the primary task. In the case of numerous objectives, this approach becomes computationally expensive because of the growing number of combinations. We can instead marginalize these variables and we fall back on (3.36). For practical reasons of flat gradients, the uniform distribution can be implemented as a distribution of the same family as $p_m$, but with a higher variance.

## 3.4 Control

In this section, we present two control strategies that can be used together with PoEs. In this section, the controller are not learned, as it will be done in Chapter 4. They are derived in order to constraint the robot or make it cover the distribution defined by the PoE. In the first control strategy presented, the PoE defines the preferred configurations of the robot. The robot should go to these configurations

and stay there, facing perturbations. The negative log density $-\log p(\boldsymbol{q}|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M)$ of the PoE is used as a cost function in an optimal control problem. This approach is illustrated in Fig. 3-1. In the second scenario, the PoE defines a distribution of configurations that the robot should actively visit.

### 3.4.1 Optimal control

In optimal control, control commands $\boldsymbol{u}$ are computed with the aim of minimizing a cost based on the control commands and on the states of the system $\boldsymbol{\xi}$. The discrete time linear quadratic tracker (LQT) is a popular tractable subproblem of optimal control, where the dynamics of the system are linear

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{A}_t \boldsymbol{\xi}_t + \boldsymbol{B}_t \boldsymbol{u}_t, \tag{3.37}$$

with $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$ being the time-dependent parameters of the system. In LQT, the cost is quadratic, given as

$$\boldsymbol{J} = \frac{1}{2} \sum_{t=1}^{N-1} \Big( (\boldsymbol{\xi}_t - \boldsymbol{z}_t)^\top \boldsymbol{L}_t (\boldsymbol{\xi}_k - \boldsymbol{z}_t) + \boldsymbol{u}_t^\top \boldsymbol{R} \boldsymbol{u}_t \Big), \tag{3.38}$$

where $\boldsymbol{z}_t$ is a desired state, $\boldsymbol{L}_t$ is a matrix weighting[1] the deviation from the desired state and $\boldsymbol{R}$ is a matrix weighting the penalization of the control commands. The minimization of the control commands have multiple underlying goals, such as reducing energy consumption, producing smooth movements through small accelerations, or ensuring safety through the use of small forces. The resulting optimal controller is linear, taking the form

$$\boldsymbol{u}_t = -\boldsymbol{K}_t \boldsymbol{\xi}_t + \boldsymbol{K}_t^v \boldsymbol{v}_{t+1}, \tag{3.39}$$

where $\boldsymbol{K}_t \boldsymbol{\xi}_t$ is a feedback term and $\boldsymbol{K}_t^v \boldsymbol{v}_{t+1}$ is a feedforward term. More details about the derivations can be found in [15].

We can use this formulation to compute a controller that would stay in regions of high density of the PoE. Let us consider that we have a linearized model of the manipulator

$$\begin{bmatrix} \boldsymbol{q}_{t+1} \\ \dot{\boldsymbol{q}}_{t+1} \end{bmatrix} = \boldsymbol{A}_t \begin{bmatrix} \boldsymbol{q}_t \\ \dot{\boldsymbol{q}}_t \end{bmatrix} + \boldsymbol{B}_t \boldsymbol{u}_t, \tag{3.40}$$

where $\boldsymbol{q}$ is the configuration of the robot and $\boldsymbol{u}_t$ can be joint accelerations or torques, depending on the controller used on the robot. The state can be augmented with the

[1] In control, $\boldsymbol{Q}$ is usually used as a notation for this matrix, but it is already used to denote a dataset of configurations in this thesis.

different experts transformations

$$\boldsymbol{\xi}_t = \begin{bmatrix} \mathcal{T}_1(\boldsymbol{q}_t) \\ \vdots \\ \mathcal{T}_M(\boldsymbol{q}_t) \\ \hline \boldsymbol{q}_t \\ \dot{\boldsymbol{q}}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_{1,t} \\ \vdots \\ \boldsymbol{y}_{M,t} \\ \hline \boldsymbol{q}_t \\ \dot{\boldsymbol{q}}_t \end{bmatrix}. \tag{3.41}$$

The dynamics of the augmented system are linearized using the Jacobian $\boldsymbol{J}_m(\boldsymbol{q}) = \frac{\partial \mathcal{T}_m(\boldsymbol{q})}{\partial \boldsymbol{q}}$ of each task space $m$

$$\boldsymbol{y}_{m,t+1} \approx \boldsymbol{y}_{m,t} + \boldsymbol{J}_m(\boldsymbol{q}_t)\,(\boldsymbol{q}_{t+1} - \boldsymbol{q}_t). \tag{3.42}$$

The complete system can then be written in matrix form as

$$\overbrace{\begin{bmatrix} \boldsymbol{y}_{1,t+1} \\ \vdots \\ \boldsymbol{y}_{M,t+1} \\ \hline \boldsymbol{q}_{t+1} \\ \dot{\boldsymbol{q}}_{t+1} \end{bmatrix}}^{\boldsymbol{\xi}_{t+1}} = \overbrace{\left[\begin{array}{ccc|cc} \boldsymbol{I}_{d_1} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \Delta t \hat{\boldsymbol{J}}_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{I}_{d_M} & \boldsymbol{0} & \Delta t \hat{\boldsymbol{J}}_M \\ \hline \boldsymbol{0} & \cdots & \boldsymbol{0} & \multicolumn{2}{c}{\boldsymbol{A}_t} \end{array}\right]}^{\tilde{\boldsymbol{A}}_t} \overbrace{\begin{bmatrix} \boldsymbol{y}_{1,t} \\ \vdots \\ \boldsymbol{y}_{M,t} \\ \hline \boldsymbol{q}_t \\ \dot{\boldsymbol{q}}_t \end{bmatrix}}^{\boldsymbol{\xi}_t} + \overbrace{\begin{bmatrix} \begin{bmatrix} \boldsymbol{0} & \hat{\boldsymbol{J}}_1 \end{bmatrix} \boldsymbol{B}_t \\ \vdots \\ \begin{bmatrix} \boldsymbol{0} & \hat{\boldsymbol{J}}_M \end{bmatrix} \boldsymbol{B}_t \\ \hline \boldsymbol{B}_t \end{bmatrix}}^{\tilde{\boldsymbol{B}}_t} \boldsymbol{u}_t. \tag{3.43}$$

The system $\tilde{\boldsymbol{A}}_t$ and $\tilde{\boldsymbol{B}}_t$ should not depend on the state; for this reason, the Jacobians should be evaluated as $\hat{\boldsymbol{J}}_m$. There could be evaluated either at the local maximum of the PoE density where the robot would converge or at the current state of the robot. The cost is defined as

$$J = \sum_{t=1}^{N-1} \Big( - \sum_{m=1}^{M} \log p_m(\boldsymbol{y}_{m,t}) + \frac{1}{2} \dot{\boldsymbol{q}}_t^\top \tilde{\boldsymbol{L}} \dot{\boldsymbol{q}}_t + \frac{1}{2}\, \boldsymbol{u}_t^\top \boldsymbol{R} \boldsymbol{u}_t \Big), \tag{3.44}$$

where $\tilde{\boldsymbol{L}}$ penalizes the velocities. With the weight control matrix $\boldsymbol{R}$, they can be chosen as a diagonal matrix

$$\tilde{\boldsymbol{L}} = \begin{bmatrix} \varsigma_1^{-2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \varsigma_P^{-2} \end{bmatrix},\ \boldsymbol{R} = \begin{bmatrix} \tau_1^{-2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \tau_P^{-2} \end{bmatrix}, \tag{3.45}$$

where $\varsigma_p$ and $\tau_p$ are respectively the range of velocities and the forces (or accelerations) allowed on joint $p$. They can be interpreted as defining a desired Gaussian distribution of velocities and forces of standard deviation $\varsigma_p$ and $\tau_p$ on joint $p$. If all the experts

$p_m$ are Gaussian, this cost can be rewritten exactly as (3.38) with

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{\Sigma}_1^{-1} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\Sigma}_M^{-1} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \tilde{\boldsymbol{L}} \end{bmatrix}, \ \boldsymbol{z_t} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_M \\ \mathbf{0} \end{bmatrix}, \tag{3.46}$$

since the log-likelihood is a quadratic function. In the other case, a quadratic approximation can be used as in iterative LQR, see [75]. The resulting controller is a feedback controller as in (3.39), where the feedback is executed on the different expert transformations

$$\boldsymbol{u}_t = \sum_{m=1}^{M} -\boldsymbol{K}_{m,t}\mathcal{T}_m(\boldsymbol{\xi}_t) + \boldsymbol{K}^v_{m,t}\boldsymbol{v}_{m,t+1}. \tag{3.47}$$

The gains $\boldsymbol{K}_{m,t}$ obtained for each expert $m$ depend on the required precision (the higher the precision, the higher the gain).

If the robot is controlled by torque commands, the whole procedure of training a PoE and controlling the robot can be used to set up automatic virtual guides, as shown in Fig. 3-1. Multiple constraints, such as keeping an orientation, staying on a plane, or pointing towards an object can be learned from demonstration, and reproduced by the robot using the proposed feedback controller. The robot will then be free to be manipulated along directions that are not encoded by the experts (or that have a high variance).

Fig. 3-7 shows a simulation of 7-DoF manipulator controlled with this strategy, with a perturbation occurring at $t = 2.35s$.

This controller can easily be implemented in a robotic manipulator with two separate loops. A high-level loop solves the LQT problem while a low-level control loop executes the last computed controller until a new one is available. These two loops are described in Alg. 1 and 2.

### 3.4.2 Ergodic control

In the second control strategy, a trajectory should be planned to visit the PoE density. This problem is referred to as ergodic coverage [79] and has multiple applications, such as surveillance or target localization. Closer to the considered manipulation tasks, it can be used to discover objects, learn dynamics or polish/paint surfaces. In [8], the ergodic coverage objective is formulated as a KL divergence between the density to visit $p(\boldsymbol{q})$ and the time-average statistics of the trajectory $\Gamma(\boldsymbol{q})$

$$D_{\mathrm{KL}}(\Gamma||p) = \int_{\boldsymbol{q}} \Gamma(\boldsymbol{q}) \log \frac{\Gamma(\boldsymbol{q})}{p(\boldsymbol{q})} d\boldsymbol{q}. \tag{3.48}$$

The time-average statistics defines the density covered by the trajectory of the robot. Using the formulation with a KL divergence, particular sensors or areas of influence

**Algorithm 1:** High-level loop (coded in Python)

set $\boldsymbol{R}$ and $\tilde{\boldsymbol{Q}}$ as in (3.45)
compute $\boldsymbol{Q}$ and $\boldsymbol{z}$, LQT cost from the expert distributions as in (3.46)
**while** *controlling* **do**
    (*loop duration :* $2-10$[ms])
    compute $\boldsymbol{A}$ and $\boldsymbol{B}$, linearized model of the manipulator around current state
    get $\boldsymbol{q}$ current configuration of the robot
    **foreach** *expert* $m$ **do**
        compute $\hat{\boldsymbol{J}}_m(\boldsymbol{q})$
    **end**
    compute $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$, augmented linear model as in (3.43)
    solve finite or infinite horizon LQT with constant parameters over the horizon
    publish $\boldsymbol{K}_1$, $\boldsymbol{K}_1^v$, $\boldsymbol{v}_2$, LQT controller parameters
**end**

**Algorithm 2:** Low-level control loop (coded in C++)

**while** *controlling* **do**
    (*loop duration :* $<1$[ms])
    read $\boldsymbol{K}_1$, $\boldsymbol{K}_1^v$, $\boldsymbol{v}_2$, current LQT controller parameters
    get $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ current configuration and velocity of the robot
    **foreach** *expert* $m$ **do**
        compute $\mathcal{T}_m(\boldsymbol{q})$ (call Python functions)
    **end**
    compute $\boldsymbol{\xi}$ as in (3.41)
    compute and apply torques or accelerations as $\boldsymbol{u} = -\boldsymbol{K}_t\boldsymbol{\xi}_1 + \boldsymbol{K}_1^v\boldsymbol{v}_2$
**end**

can be taken into account. For example, a Gaussian can be used to model the area around the robot perceived by its sensors. The time-average statistics become a mixture of Gaussians

$$\Gamma(\boldsymbol{q}) \approx \frac{1}{N}\sum_{t=0}^{N-1}\mathcal{N}(\boldsymbol{q}|\boldsymbol{q}_t, \boldsymbol{\Sigma}), \tag{3.49}$$

where $\boldsymbol{q}_t$ is the configuration of the robot at time $t$ (for a trajectory of $N-1$ timesteps). The covariance $\boldsymbol{\Sigma}$ can model the coverage of the sensor. A small $\boldsymbol{\Sigma}$ means a short-distance coverage and implies finer trajectories. The objective (3.48) is the same as in variational inference and can be optimized with the ELBO objective in (2.9). Thus, it is not required to have a properly normalized density to visit. Given that the covered zone is modeled by a tractable density (from which we can sample, and

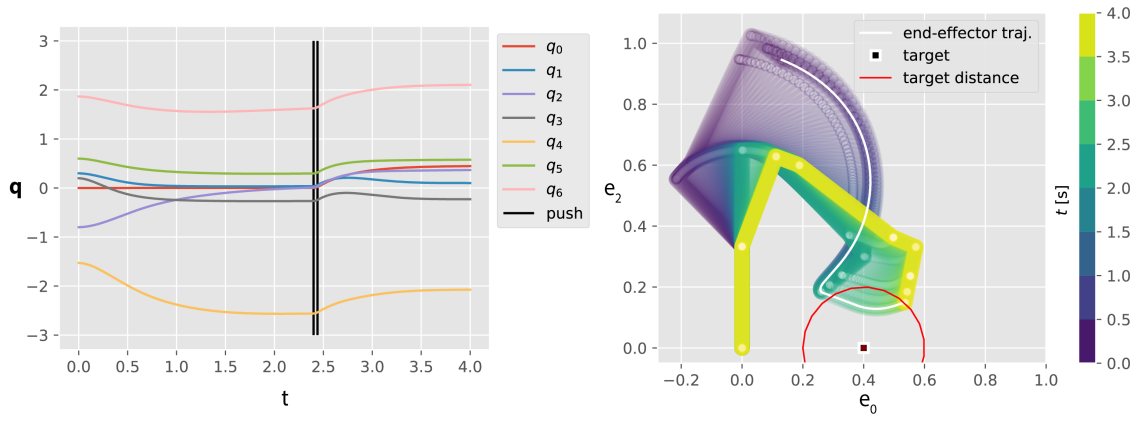



**Figure 3-7:** Example of 7-DoF manipulator controlled with an LQT. To show the time evolution, only the kinematic chain is displayed. The PoE is the intersection between a vertical plane ($e_1 = 0$) and a sphere. The two experts act on the end-effector. They are defined by a Gaussian distribution on $e_1$ around 0 and by a Gaussian distribution on the log-distance to a target (black square). On the left, the evolution of the values of the joint angles is displayed over time. At $t = 2.35[s]$, a perturbation in configuration space occurs. The robot converges to a new local mode of the PoE.

properly normalized), the ergodic objective can be optimized easily with stochastic gradient descent. Compared to the ELBO objective in (2.9), an additional constraint has to be added for limiting velocities and accelerations so that $\{\boldsymbol{q}_t\}_{t=1}^{N-1}$ is a consistent trajectory.

Fig. 3-8 shows a trajectory optimized to cover the PoE. The robot is assumed to be controlled by velocity commands. Velocities and accelerations are minimized, and the time-averaged statistics is a mixture of Gaussians in task space.

## 3.5 Experiments

### 3.5.1 Multimodal distributions

In a first experiment, we show the advantages of using variational inference to approximate the derivative of the normalizing constant. We consider a 2-DoF planar robot that should learn to distinguish between an operational-space or a configuration-space objective. Three datasets are used and shown in Fig. 3-9. They correspond to the following subtasks: (a) only configuration space target (b) only task space target, involving a multimodal PoE (c) a task-space target with a joint angle preference, involving two modes of unequal weights. The PoE model is defined by the following

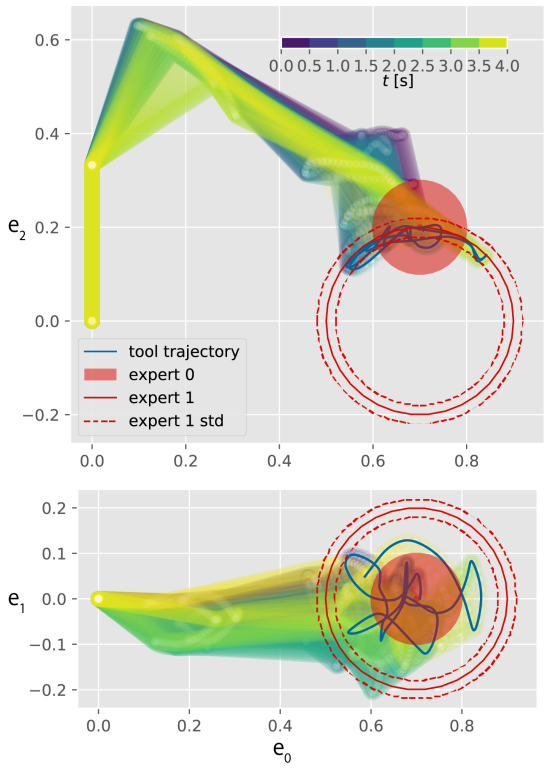



**Figure 3-8:** Time evolution of a robot covering a PoE distribution indicated with a colormap. The trace of the end-effector is shown with a blue line. The trajectory is optimized using ergodic control. The density is defined by a position constraint of the end-effector (expert 0-red disk) and a distance constraint (expert 1-red circle), defining a virtual sphere.

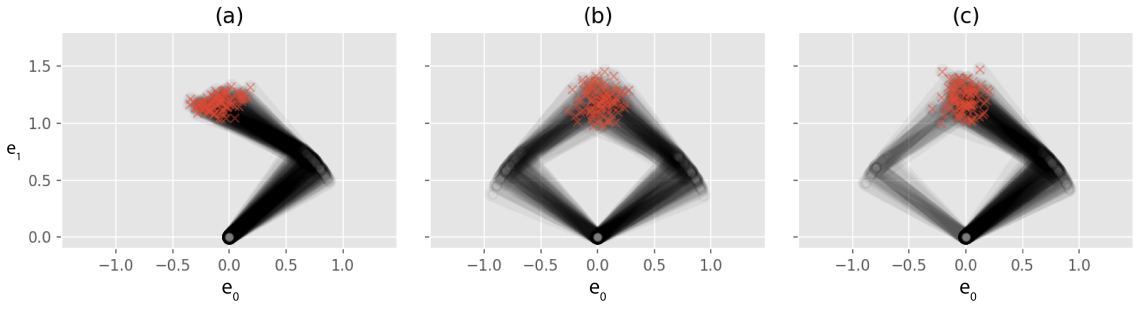



**Figure 3-9:** Samples of configurations for three tasks that illustrate the advantages of using variational inference with a mixture model for training a PoE. A 2-DoF planar robot is considered. *(a)* Joint angle target. *(b)* Operational space target *(c)* Operational space target and joint angle target.

task spaces and distributions:

$$\begin{aligned}
\text{state}: \quad & \boldsymbol{q} \in \mathbb{R}^2, \\
\text{task spaces}: \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{q}) \in \mathbb{R}^2, \\
& \boldsymbol{y}_2 = \mathcal{T}_2(\boldsymbol{q}) = \boldsymbol{q} \in \mathbb{R}^2, \\
\text{experts}: \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \sigma_1 \boldsymbol{I}), \\
& \boldsymbol{y}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \sigma_2 \boldsymbol{I}), \\
\text{parameters to learn}: \quad & \boldsymbol{\mu}_1, \sigma_1, \boldsymbol{\mu}_2, \sigma_2,
\end{aligned}$$

where $\boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{q})$ is the position of the end-effector of the planar robot. The first and second experts analyze task-space objectives and configuration-space objectives, respectively. Given three datasets of $N = 30$ configurations, the corresponding models are learned either with contrastive divergence (as proposed in [52]) or using variational inference (as proposed in Sec. 3.1.3). The process is repeated multiple times with random initializations of the parameters.

Quantitative evaluations are performed by computing the alpha-divergence with $\alpha = 1/2$ between the ground-truth density from which the dataset was sampled and the density learned with the different techniques. This divergence is also related to the Bhattacharyya coefficient as

$$D_{\alpha=1/2}(p||\tilde{q}) = -2 \log \int \sqrt{p(\boldsymbol{q})\tilde{q}(\boldsymbol{q})} d\boldsymbol{q}. \tag{3.50}$$

This integral was evaluated by discretizing the space. Results are reported in Table 3.1. In all cases, the variational approximation performs better. The performance of contrastive divergence is especially poor in case (a), showing high variations. The explanation comes from the incapability of standard sampling methods to jump across distant modes. Only the existence of a second mode, as in case (b) and (c) can help to distinguish between a configuration space and a task space target. A Markov chain initialized on the data never discovers the second mode that would be implied by a task space target. This potential waste of probability mass goes unnoticed by contrastive divergence. In contrast, variational inference with a mixture of Gaussians can localize this region of probability. The high variance of the contrastive divergence is due to the nonexistent gradient between the relative standard deviations $\sigma_1$ and $\sigma_2$ of the two experts. The final results are thus determined mostly by the random initialization.

### 3.5.2 Hierarchical tasks

In this set of experiments, the robot has to learn two competitive objectives, where one has a higher priority than the other. The secondary objective is masked by the resolution of the first objective and has to be uncovered.

**Planar robot** We first evaluate our approach in simpler cases, with planar robots. In the first task, we consider a 5-DoF planar robot with two dependent arms, as

| Task | (a) | (b) | (c) |
|---|---|---|---|
| CD | $0.837 \pm 0.597$ | $0.039 \pm 0.030$ | $0.063 \pm 0.026$ |
| **VI** | $\mathbf{0.067 \pm 0.060}$ | $\mathbf{0.016 \pm 0.022}$ | $\mathbf{0.014 \pm 0.004}$ |

**Table 3.1:** Quantitative evaluations of the learned distribution using contrastive divergence (CD) and our approach based on variational inference with a Gaussian mixture model (VI). The table shows alpha-divergence $D_{\alpha=1/2}$ measures between the different approximations and the ground-truth distribution, computed for the three tasks shown in Fig. 3-9.

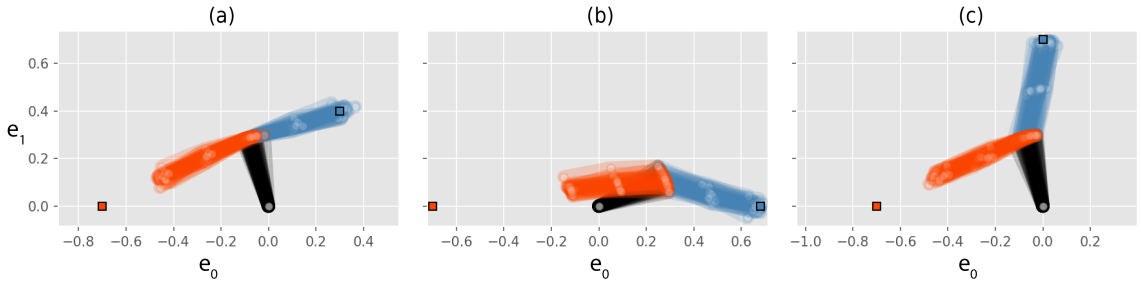



**Figure 3-10:** Dataset for the bimanual task with a hierarchy. Three situations are given, where each end-effector should track the target of the same color (displayed as a square).

illustrated in Fig. 3-10. Each end-effector should track its own target, which should be recovered from demonstrations. The end-effector with the highest priority task has three different targets $\{\boldsymbol{\mu}_1^{(i)}\}_{i=0,\dots,2}$ and the other end-effector has a fixed target $\boldsymbol{\mu}_2$. For each of the three cases $i = 0, \dots, 2$, $N = 30$ independent samples $\hat{\boldsymbol{q}}_n$ are given. The samples are generated by approximating a ground-truth product of experts with a mixture of $K = 50$ Gaussian components. The end-effectors try to follow a normal distribution with a standard deviation $\sigma_1 = \sigma_2 = 0.02$ around their respective targets. In each situation, the secondary target is not reachable.

The PoE model is defined as:

$$
\begin{aligned}
\text{state :} \quad & \boldsymbol{q} \in \mathbb{R}^5, \\
\text{task spaces :} \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_{R,\boldsymbol{x}}(\boldsymbol{q}) \in \mathbb{R}^2, \\
& \boldsymbol{y}_2 = \mathcal{T}_2(\boldsymbol{q}) = \boldsymbol{F}_{L,\boldsymbol{x}}(\boldsymbol{q}) \in \mathbb{R}^2, \\
\text{experts :} \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1^{(i)}, \sigma_1 \boldsymbol{I}) \,|\, i = 0, \dots, 2, \\
& \boldsymbol{y}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \sigma_2 \boldsymbol{I}), \\
\text{parameters to learn :} \quad & \{\boldsymbol{\mu}_1^{(i)}\}_{i=0,\dots,2}, \boldsymbol{\mu}_2, \sigma_1, \sigma_2.
\end{aligned}
$$

$\boldsymbol{F}_{R,\boldsymbol{x}}$ and $\boldsymbol{F}_{L,\boldsymbol{x}}$ denote respectively the forward kinematics (position only) of the right

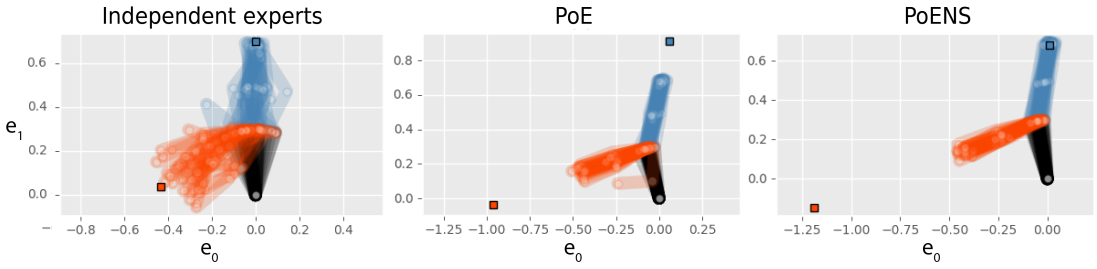



**Figure 3-11:** Samples from the different models learned for the bimanual task with a hierarchy. *Left:* Using independent experts, the secondary task is not well understood and its variance is over-estimated. *Center:* Using a PoE, the target of the primary task is inferred to be further away. *Right:* Using a PoE with a nullspace structure, the targets as well as the priorities are recovered.

(blue) and left (orange) arm. From the set of samples $\hat{\boldsymbol{q}}_n$, the position of the targets $\{\boldsymbol{\mu}_1^{(i)}\}_{i=0,\dots,2}$, $\boldsymbol{\mu}_2$ and their standard deviation $\sigma_1$ and $\sigma_2$ should be retrieved.

We compare three approaches to learn the model. The first consists of maximum likelihood estimation of each expert separately, after applying the transformations to the dataset. With this approach, the main task is well understood. It is not the case of the secondary task, which is masked. The left end-effector displays a large variance, as shown in Fig. 3-11(a). This variance cannot be explained without taking into account the dependence between the two tasks and their prioritization. The second approach is to use a PoE with no prioritization between the objectives. The dependence between the tasks is taken into account, which results in a better understanding of the secondary objective. The prioritization is still not taken into account, which has two negative effects. As shown in Fig. 3-11(b), the targets of the right arm $\{\boldsymbol{\mu}_1^{(i)}\}_{i=0,\dots,2}$ are inferred further than they are, and the standard deviation of the secondary task $\sigma_2$ is slightly exaggerated. These two misinterpretations compensate the missing hierarchical structure by artificially increasing the importance of the primary task. In the third approach, the information of hierarchy is restored by using a product of experts with the nullspace structure (PoENS) as presented in Sec. 3.2. This time, the standard deviations of the two tasks as well as their respective targets are well retrieved.

Quantitative results are produced in a slightly different manner than for the previous experiment. As only the gradient of the unnormalized log-likelihood of the PoENS is defined, they are first approximated using variational inference with a mixture of $K = 50$ Gaussians. Results are reported in Table 3.2 under task (a). As noticed in Fig. 3-11, the divergence is the smallest with the PoENS.

In the second task with planar robots, we are interested in the manipulability measure presented in Sec. 3.3.1. The considered robot has three joints and its principal task is to track a point with its end-effector. One degree of freedom remains, which should be used to maximize the manipulability measure. This objective is set as a log-normal distribution on the determinant of the manipulability matrix (i.e., Gaussian distribution on the log). It defines which manipulability value $\mu_2$ to track, together

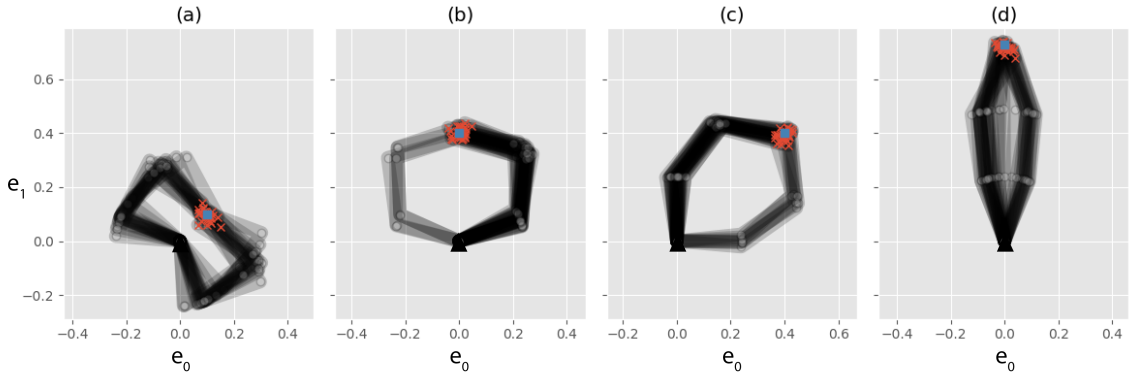



**Figure 3-12:** Dataset for the hierarchical task with a manipulability objective. The robot should track the target (displayed as a blue square) and maximize a manipulability measure. Four different targets are given.

with the allowed variation $\sigma_2$. In the same way as in the previous experiment, the end-effector should track four different targets $\{\boldsymbol{\mu}_1^{(i)}\}_{i=0,\dots,3}$. For each of the four cases $i = 0, \dots, 3$, $N = 30$ independent samples $\hat{\boldsymbol{q}}_n$ are given, which are shown in Fig. 3-12. The PoE model is defined as:

$$
\begin{aligned}
\text{state :} \quad & \boldsymbol{q} \in \mathbb{R}^3, \\
\text{task spaces :} \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{q}) \in \mathbb{R}^2, \\
& y_2 = \mathcal{T}_2(\boldsymbol{q}) = \log\det\left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^\top\right) \in \mathbb{R}^+, \\
\text{experts :} \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1^{(i)}, \sigma_1 \boldsymbol{I}) \,|\, i = 0, \dots, 3 \\
& y_2 \sim \mathcal{N}(\mu_2, \sigma_2), \\
\text{parameters to learn :} \quad & \{\boldsymbol{\mu}_1^{(i)}\}_{i=0,\dots,3}, \mu_2, \sigma_1, \sigma_2,
\end{aligned}
$$

where $\boldsymbol{F}(x)$ is the position of the final link.

As before, we compare three ways of learning the model. As a baseline, we follow the approach of [58] by considering independent training of the experts. We employ a less elaborated description of the manipulability task than in [58], by considering the volume of the manipulability ellipsoid instead of the full ellipsoid. For our study, this simplified scalar descriptor is sufficient to showcase the advantages of PoENS over an independent training of the experts.

Fig. 3-13 presents samples generated from the different models (for a target as in Fig. 3-13-*(b)*). When the experts are learned independently (*left*), the manipulability is not well understood. The samples are tracking a manipulability that is lower than expected, and the variance is overestimated. When training the PoE without the nullspace structure (*center*), the distribution of generated samples better matches the dataset. However, the task-space target is not well estimated (it is inferred closer to the base of the robot to counterbalance the manipulability objective). In contrast, the samples retrieved from the PoENS (*right*) show that the task-space target is well understood.

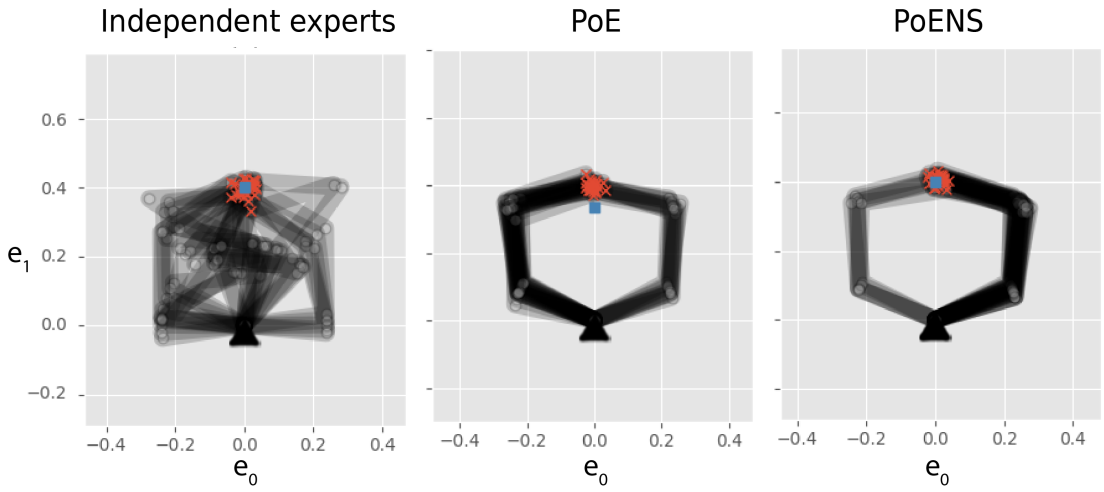



**Figure 3-13:** Samples from the different models learned for the task with a manipulability objective. *Left:* Using independent experts, the secondary task is not well understood (the manipulability measure has a big variance in the dataset). *Center:* Using a PoE, the estimation of the principal objective is biased. *Right:* By providing the hierarchical structure, the two objectives are well understood.

Quantitative evaluations for a bimanual robot are also reported in Table 3.2.

These results show that considering independent training of the experts has two limitations: (1) it does not exploit the dependence between the task-space position $\boldsymbol{F}(x)$ and the manipulability; and (2) it does not exploit the hierarchical structure that often relegates the manipulability objective to a secondary objective. When this structure is not exploited, the manipulability objective cannot be understood, as it is not directly observed. By maximum likelihood estimation of the log-normal distribution on the dataset, the variance is high and the mean value does not reflect the fact that we were targeting the maximum of manipulability. For example, the demonstration data in Fig. 3-12-*(d)* are characterized by very small manipulability ellipsoids, which would be interpreted erroneously if modeled independently from the tracking task.

**7-DoF manipulator** We conduct a similar experiment with a 7-DoF Panda robot. The dataset mimics a welding task, in which the position of the end-effector is more important than its orientation. The robot is supposed to track the position of the component to weld while preferably keeping its orientation vertical. We give three sets of $N = 30$ independent samples $\hat{\boldsymbol{q}}_n$ for three different target positions $\{\boldsymbol{\mu}_x^{(i)}\}_{i=0,\dots,3}$ as shown in Fig. 3-14. The orientation should be kept fixed (the same in the three

| Task | (a) | (b) |
|---|---|---|
| Independent experts | $1.814 \pm 0.055$ | $0.812 \pm 0.117$ |
| PoE | $0.258 \pm 0.101$ | $0.630 \pm 0.086$ |
| **PoENS** | $\mathbf{0.094 \pm 0.024}$ | $\mathbf{0.202 \pm 0.067}$ |

**Table 3.2:** Quantitative evaluations of the quality of the learned distribution for the two planar tasks. The table shows alpha-divergence $D_{\alpha=1/2}$ measures between the different approximations and the ground-truth distribution are computed. *(a)* Bimanual task *(b)* Manipulability objective.

cases). The PoE model is defined as:

$$\begin{aligned}
\text{state}: \quad & \boldsymbol{q} \in \mathbb{R}^7, \\
\text{task spaces}: \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_x(\boldsymbol{q}) \in \mathbb{R}^3, \\
& \boldsymbol{y}_2 = \mathcal{T}_2(\boldsymbol{q}) = \text{vec}(\boldsymbol{F}_R(\boldsymbol{q})) \in \mathbb{R}^9, \\
& \boldsymbol{y}_3 = \mathcal{T}_3(\boldsymbol{q}) = \boldsymbol{q} \in \mathbb{R}^7, \\
\text{experts}: \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_x^{(i)}, \sigma_x \boldsymbol{I}) \,|\, i = 0, \dots, 3, \\
& \boldsymbol{y}_2 \sim \mathcal{N}(\boldsymbol{\mu}_R, \sigma_R \boldsymbol{I}), \\
& \boldsymbol{y}_3 \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{q}}, \sigma_{\boldsymbol{q}} \boldsymbol{I}), \\
\text{parameters to learn}: \quad & \{\boldsymbol{\mu}_x^{(i)}\}_{i=0,\dots,3}, \sigma_x, \boldsymbol{\mu}_R, \sigma_R, \boldsymbol{\mu}_{\boldsymbol{q}}, \sigma_{\boldsymbol{q}}
\end{aligned}$$

where $\boldsymbol{F}_x(\boldsymbol{q})$ is the position of the end-effector and $\text{vec}(\boldsymbol{F}_R(\boldsymbol{q}))$ its vectorized rotation matrix. We choose a Gaussian distribution for the vectorized rotation matrix with an isotropic covariance matrix $\sigma_R \boldsymbol{I}$. As the renormalization arises at the joint level in a PoE, this choice is acceptable even if the rotation expert does not have the proper support.

We compared the same three ways of training the model. As expected, with independent experts, a mean orientation is computed, resulting in biased estimation of this objective (see Fig. 3-14 second column). The PoE without hierarchy performs a bit better, as it can understand the dependence between the position and the orientation induced by the kinematics. The estimation of the orientation is still biased (see Fig. 3-14 third column). As in previous experiments, the PoENS performs the best. The secondary objective of orientation is clearly understood.

For the quantitative experiments, we compare the dataset with the learned distributions. This comparison is done in each of the 3 situations and by using the maximum mean discrepancy[2] $\text{MMD}_u^2$, see [43]. The discrepancy is computed with 500 samples and a RBF kernel with $\gamma = 0.1$. The results are reported in Table 3.3

[2]with an unbiased estimate of the kernel mean, it is possible to get negative values. Negative values indicates a close to zero discrepancy; their absolute value can be discarded.

| Case | (0) | (1) | (2) |
|---|---|---|---|
| Independent experts | $1.3e^{-3}$ | $1.3e^{-2}$ | $8.2e^{-3}$ |
| PoE | $7.0e^{-4}$ | $1.7e^{-3}$ | $1.4e^{-3}$ |
| **PoENS** | $\mathbf{-9.8e^{-6}}$ | $\mathbf{-8.5e^{-5}}$ | $\mathbf{4.5e^{-4}}$ |

**Table 3.3:** Quantitative evaluations of the quality of the learned distribution for the 7-DoF Panda robot (welding task). The table shows maximum mean discrepancy $\text{MMD}_u^2$ measures between the dataset and different models for the 3 cases displayed in Fig. 3-14.

and in Fig. 3-14, where PoENS shows much better results.

### 3.5.3 Learning transformations and conditional PoEs

In these experiments, we compare PoEs with less structured ways to learn distributions. We show that PoEs, by providing more structure, require fewer samples to be trained. They also allow finding simple explanations for complex distribution, leading to quicker generalization. Unlike the previous experiments, the experts transformations are this time only partially known.

**New end-effector** We consider an experiment with the 7-DoF Panda robot in which a new end-effector is defined. The new end-effector has a fixed position $\boldsymbol{d}$ in the coordinate system of the known end-effector. We want to learn the distribution of configurations where the new end-effector follows a Gaussian distribution around $\boldsymbol{\mu}_x$, a given task-space position target. To compare the data-efficiency of the models, the datasets are composed of $N \in \{3, 10, 30, 100, 3000\}$ independent samples, as illustrated in Fig. 3-15-*left*. The PoE model is defined as:

$$\begin{aligned}
\text{state}: \quad & \boldsymbol{q} \in \mathbb{R}^7, \\
\text{task spaces}: \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_R(\boldsymbol{q})\boldsymbol{d} + \boldsymbol{F}_x(\boldsymbol{q}) = \boldsymbol{F}_d(\boldsymbol{q}) \in \mathbb{R}^3, \\
& \boldsymbol{y}_2 = \mathcal{T}_2(\boldsymbol{q}) = \boldsymbol{q} \in \mathbb{R}^7, \\
\text{experts}: \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_x, \sigma_x \boldsymbol{I}), \\
& \boldsymbol{y}_2 \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{q}}, \sigma_{\boldsymbol{q}} \boldsymbol{I}), \\
\text{parameters to learn}: \quad & \boldsymbol{d}, \boldsymbol{\mu}_x, \sigma_x, \boldsymbol{\mu}_{\boldsymbol{q}}, \sigma_{\boldsymbol{q}},
\end{aligned}$$

where $\boldsymbol{F}_R(\boldsymbol{q})$ is the rotation matrix and $\boldsymbol{F}_x(\boldsymbol{q})$ the position of the known end-effector. The second expert learns a preferred joint configuration. We compare the PoE with five other techniques to learn distributions. The first is a variational autoencoder [66]. As architecture for both the encoder and the decoder, we used 2 layers of 20 fully connected hidden units and tanh activation. The latent space was of 4 dimensions,

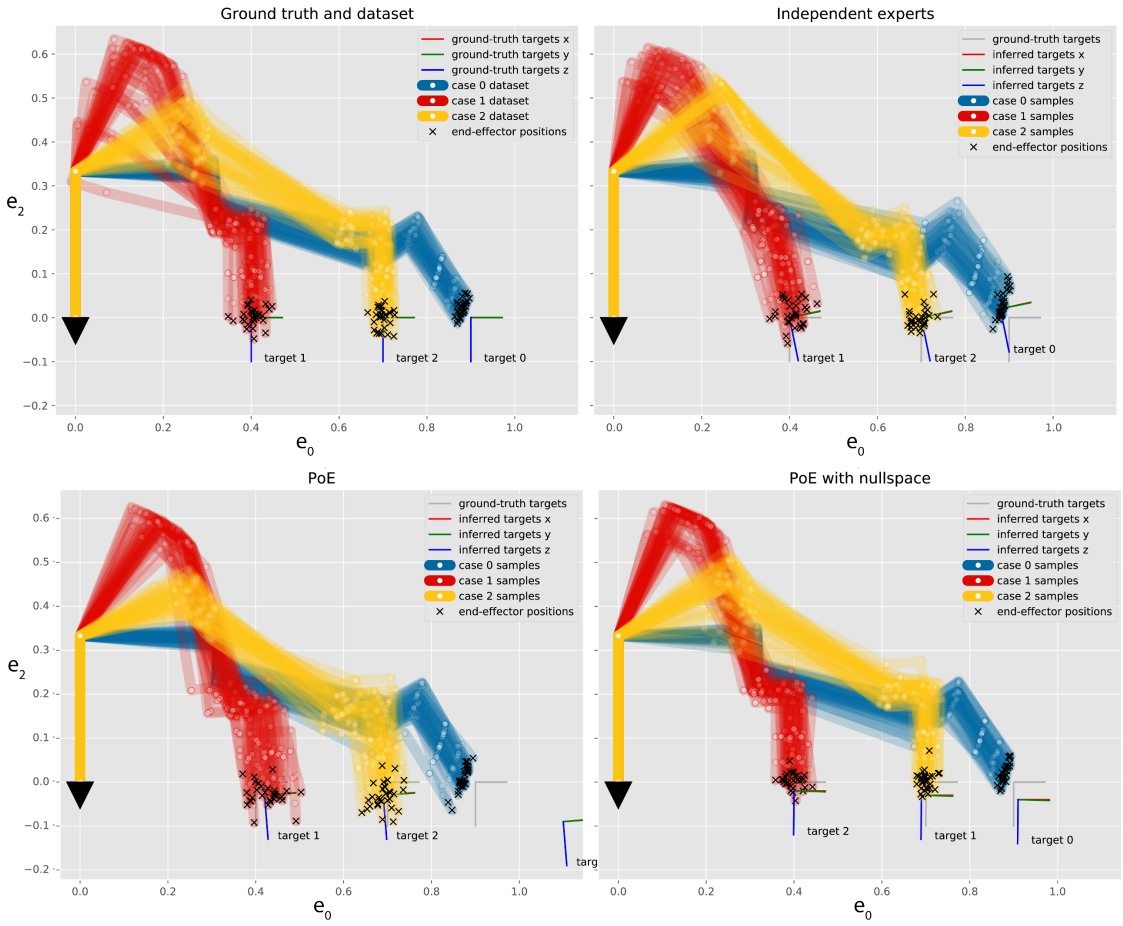



**Figure 3-14:** Dataset and samples from the different models for the hierarchical task with the 7-DoF Panda robot (welding task). Only the kinematic chain of the robot is displayed, such that multiple samples can be shown. The dataset provides three different situations (displayed in blue, yellow and red) in which the robot should track a different target with its end-effector. In each situation, the orientation should be held vertical as a secondary task.

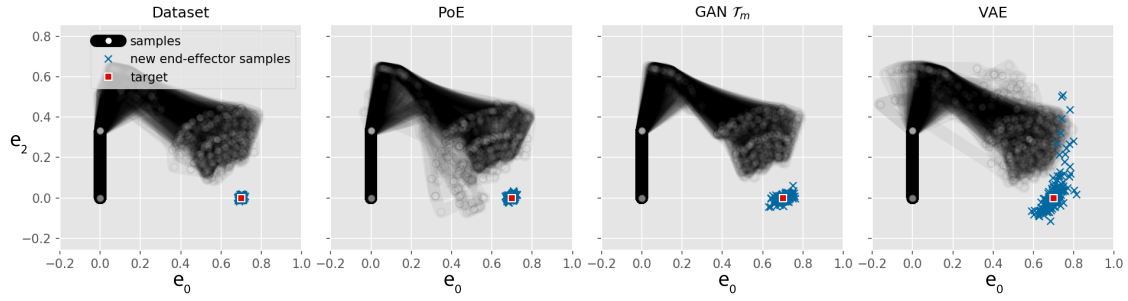



**Figure 3-15:** Dataset and samples from the different models for a task involving an unknown end-effector. In this illustration, the models were trained with 3000 samples.

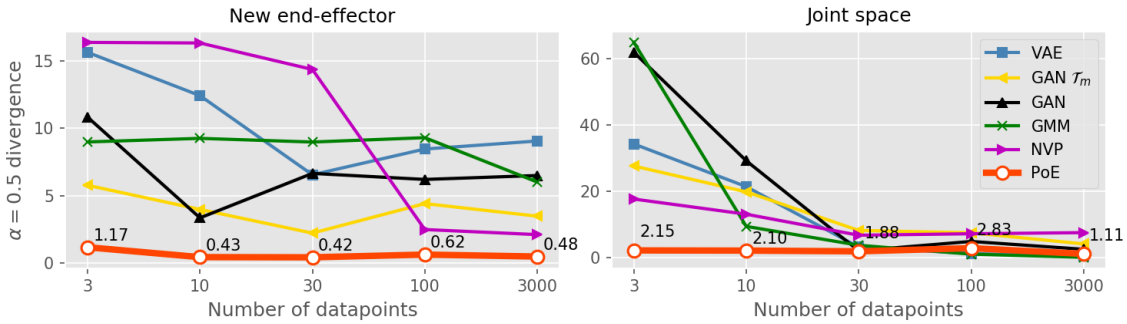



**Figure 3-16:** Quantitative results for a task involving an unknown end-effector. The graphs show alpha-divergence measures (with $\alpha = 1/2$) between the dataset and the different models for the different sizes of the dataset.

which corresponds to the remaining DoFs of the robot, after constraining 3 DoFs with the position of the new end-effector. The second technique considered is a Dirichlet process Gaussian mixture model [13]. In this model, the number of Gaussian components (with full covariances) is learned according to the number of datapoints and the complexity of the distribution. Another model tested is a transformed distribution using an invertible neural network (NVP) as in [23]. As architecture, four layers of 108 hidden units with relu activation were used. The two last models are GANs [41]. In these two cases, both the generator and discriminators are multilayer perceptrons with 3 layers of 50 hidden units each and sigmoid activation. The latent space of the generator is of 10 dimensions. In the second GAN, that we denote "GAN $\mathcal{T}_m$", the discriminator is helped by accessing the samples and demonstrations through the different task-space transformations.

For each of the six techniques, the model was learned with a different number of samples. Samples from the PoE, the VAE and the GAN $\mathcal{T}_m$are shown in Fig. 3-15, where the whole dataset was used to train the models. For each model, 500 samples were generated, for which the position of the new end-effector was computed. We compared the distribution of end-effector positions between each model and the dataset using maximum mean discrepancy $\text{MMD}_u^2$. The results are reported in Fig. 3-16. The PoE performs better than the others for each number of datapoints. Its performance is also less influenced by this number. Its advantage can be explained by the difficulty to represent the distribution in configuration space, which has a complex shape. Since the target is also very precise, this distribution is close to a 4 dimensional manifold embedded in a 7-dimensional space, which makes it particularly difficult to be encoded as a mixture of Gaussians. A Gaussian approximating a part of this manifold will have an important portion of its density outside of the manifold, resulting in an imprecise tracking. Only a high number of Gaussian components can approximate well this distribution. Moreover, training such a model requires huge datasets to cover well the entire manifold.

It is interesting to notice that GAN $\mathcal{T}_m$ performs quite well. In Fig. 3-15, samples from the new end-effector are tracking the target better with GAN $\mathcal{T}_m$ than with

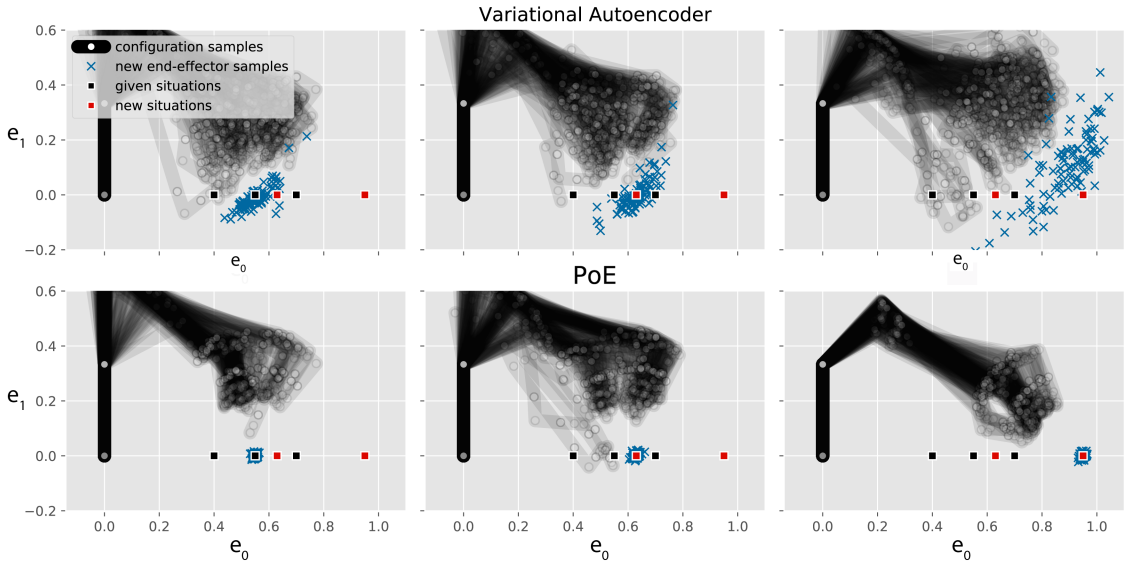



**Figure 3-17:** Dataset and samples from the different models for a task involving an unknown end-effector. Three cases are reproduced. *Left column:* The distribution is sampled on a known target. *Middle column:* The target is in-between two given targets. *Right column:* The target is far from the given targets. The first and second rows show results for a conditional variational autoencoder and a PoE, respectively.

VAE. The standard GAN was not displayed here but has very similar performance as the VAE. Unlike VAEs, GANs are very appropriate to exploit the existence of task spaces as hand-engineered features of the discriminator. To our knowledge, they are the best alternative to PoEs if more data is available.

**Conditional distributions** This experiment is similar to the previous one, but requires a generalization to different targets for the new end-effector. The dataset is split into 3 cases with $N = 1000$ datapoints for each. Unlike in the previous experiment, the targets $\{\boldsymbol{\mu}_x^{(i)}\}_{i=0}^2$ are given. The PoE model is defined as before, with:

$$\begin{aligned}
\text{state}: \quad & \boldsymbol{q} \in \mathbb{R}^7, \\
\text{task spaces}: \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_R(\boldsymbol{q})\boldsymbol{d} + \boldsymbol{F}_x(\boldsymbol{q}) = \boldsymbol{F}_d(\boldsymbol{q}) \in \mathbb{R}^3, \\
& \boldsymbol{y}_2 = \mathcal{T}_2(\boldsymbol{q}) = \boldsymbol{q} \in \mathbb{R}^7, \\
\text{experts}: \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}_x^{(i)}, \sigma_x \boldsymbol{I}) \,|\, i = 0, \dots, 3, \\
& \boldsymbol{y}_2 \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{q}}, \sigma_{\boldsymbol{q}} \boldsymbol{I}), \\
\text{parameters to learn}: \quad & \boldsymbol{d}, \sigma_x, \boldsymbol{\mu}_{\boldsymbol{q}}, \sigma_{\boldsymbol{q}}.
\end{aligned}$$

For the evaluations, we used a conditional variational autoencoder (VAE), where the targets $\{\boldsymbol{\mu}_x^{(i)}\}_{i=0}^2$ are concatenated to the corresponding datapoints at the entrance of the encoder and also concatenated to the latent variables at the entrance

| Case | (a) | (b) | (c) |
|---|---|---|---|
| VAE | $8.8e^{-4}$ | $2.2e^{-4}$ | $1.0e^{-2}$ |
| GMR | $2.7e^{-6}$ | $5.1e^{-6}$ | $9.0e^{-2}$ |
| **PoE** | $\mathbf{3.5e^{-9}}$ | $\mathbf{-2.2e^{-7}}$ | $\mathbf{3.4e^{-7}}$ |

**Table 3.4:** Quantitative results for the task with the new end-effector. The table shows maximum mean discrepancy $\text{MMD}_u^2$ measures between the dataset and the different models for the different cases. (a) The distribution is sampled on a known target. (b) The target is in-between two given targets. (c) Generalization with respect to a new target far from the given targets.

of the decoder. For the Gaussian mixture model, we encode the joint distribution of configurations $\boldsymbol{q}$ and targets and used conditioning over new targets to retrieve the corresponding distribution of states, similarly to Gaussian mixture regression (GMR). We evaluate the quality of the distribution with $\text{MMD}_u^2$ in three cases. In case (a) (Fig. 3-17-*left*), the target is one already given in the dataset. In case (b) (Fig. 3-17-*center*), the target is between two given targets. In (a) and (b), VAE performs quite well, with the same limitations as in the previous experiments. In the last case (Fig. 3-17-*right*), the target is far outside and the performance of VAE further reduced. The PoE performs very well in the three cases, as shown also in the quantitative evaluation reported in Table 3.4. The only drawback is that it requires some time to approximate the distribution given the new target, as only the unnormalized density of the PoE is directly accessible. Note that this is not a problem if the log-pdf is used as a reward function in optimal control, as proposed in Sec. 3.4. Regarding the computation time for training the three techniques, they are about one minute for both the VAE and the PoE while below 0.1 second for GMR. The three techniques being trained with an iterative procedure, their training time can be reduced at the expense of the precision.

### 3.5.4 End-effector position and rotation correlations

In this last experiment, we show how our framework can help to cope with orientation statistics. Typical distributions, such as matrix Bingham—von Mises—Fisher distribution (BMF) are hard to use, because of their intractable normalizing constant. In PoEs, the normalization happens in the configuration space, which makes the use of intractable density possible. Another tempting approach is to use a multivariate normal distribution of vectorized rotation matrices. When training with maximum likelihood estimation, this leads to bad approximations. Indeed, the wrong normalizing constant is considered because the space of rotation matrices is discarded. We show that using such vectorized distributions is fine with the PoE as the normalizing constant is computed in the product space. Also, in some cases, the Bingham—von

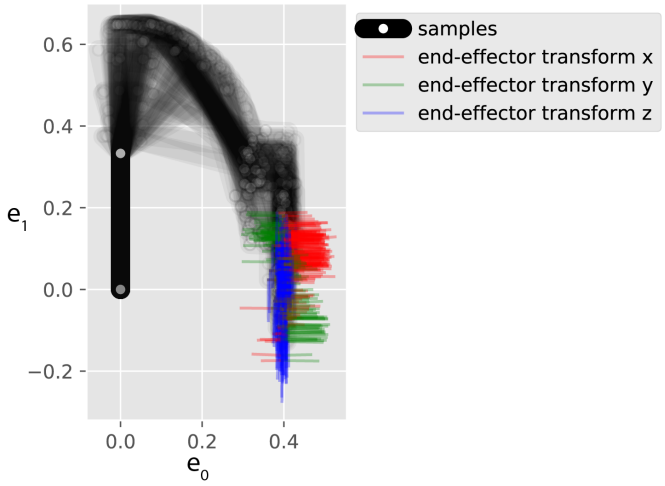



**Figure 3-18:** Samples where the height of the end-effector is correlated with its rotation along the $e_1$-axis. The transformation matrix of the end-effector is displayed.

Mises—Fisher distribution can be computed as a Gaussian on the vectorized matrix (see Sec. 3.3.2).

We produced three datasets, composed each one of $N = 3000$ independent samples from a ground-truth PoE model. The number of samples is much higher than required to understand the task but reduces the variance of the estimation of the parameters to provide precise comparisons. The PoE model used to sample has a correlation between the height of the end-effector ($e_2$) and the rotation along this axis. In the first case (a), the correlation is induced by penalizing deviations from a linear relationship between elements of the rotation matrix and the height of the end-effector. This dataset is shown in Fig. 3-18. In the two other cases (b) and (c), the correlation is induced with the $z$ Euler angle instead of the rotation matrix. In (b) and (c), the standard deviations of this angle is 0.2 and 1., respectively. The PoE model is defined as:

$$\begin{aligned} \text{state}: \quad & \boldsymbol{q} \in \mathbb{R}^7, \\ \text{task spaces}: \quad & \boldsymbol{y}_1 = \mathcal{T}_1(\boldsymbol{q}) = \boldsymbol{F}_{\boldsymbol{x},\text{vec}(\boldsymbol{R})}(\boldsymbol{q}) \in \mathbb{R}^{12}, \\ \text{experts}: \quad & \boldsymbol{y}_1 \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}) + \boldsymbol{L}\boldsymbol{L}^T), \\ \text{parameters to learn}: \quad & \boldsymbol{\mu}, \sigma, \boldsymbol{L}, \end{aligned}$$

where $\boldsymbol{y}_1$ is a concatenation of the position of the end-effector and its vectorized rotation matrix. The expert is a Gaussian with a structured covariance matrix. The covariance is composed of a diagonal component $\text{diag}(\boldsymbol{\sigma})$, whose diagonal is the vector $\boldsymbol{\sigma}$ and a low-rank component $\boldsymbol{L} \in \mathbb{R}^{12 \times n_{\text{axis}}}$ that encodes correlations. In this experiment, $n_{\text{axis}}$ is chosen to be 1 because there is only a single axis of covariance. The covariance matrix and the mean $\boldsymbol{\mu}$ can also be parametrized to have a marginal distribution of rotations in the form of a matrix Bingham—von Mises—Fisher distribution, a marginal distribution of position as a Gaussian and a covariance between the two (see (3.27)).

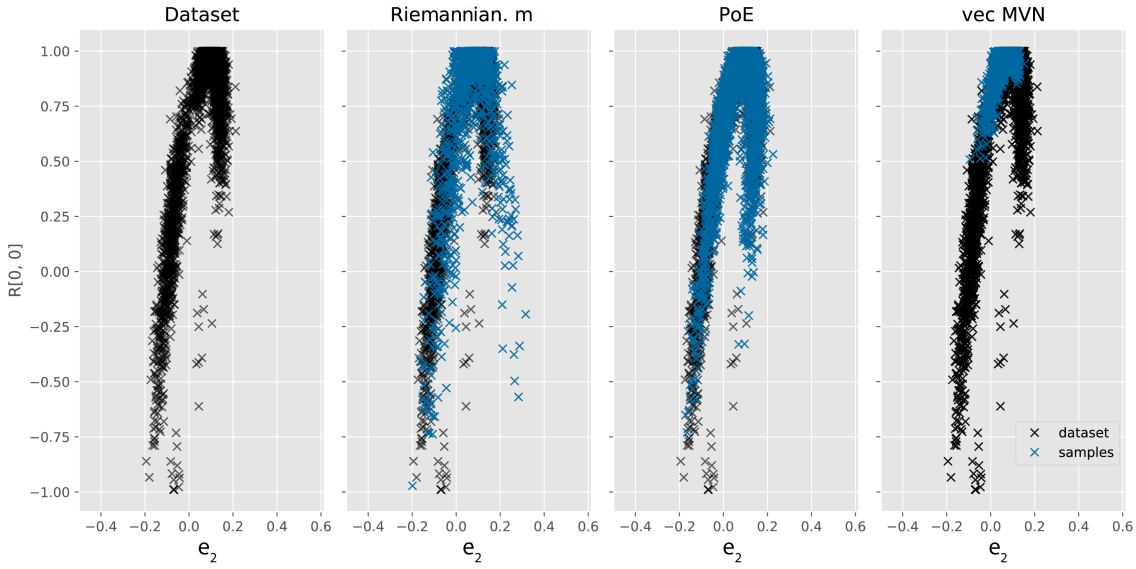



**Figure 3-19:** Scatter plot showing the correlation between the height of the end-effector and the first element of the first row of the rotation matrix for case (a). The dataset is shown in black and the samples from the different models in blue.

We compare our approach with two others, where only the concatenated position and orientation $\boldsymbol{y}_1$ is taken into account and without exploiting the kinematic structure of the robot. In the first alternative, we perform maximum likelihood estimation of a Gaussian with $\boldsymbol{y}_1$. In the second, we use a Gaussian on a Riemannian manifold as in [124]. The rotation matrices are converted to quaternions and the considered manifold is the product between a Euclidean and spherical 3-manifold. For the quantitative evaluation, we sampled 500 points from each model and computed $\text{MMD}_u^2$ with the dataset. Results are reported in Table 3.5. The vectorized Gaussian is the worst-performing model. Its best performance is in case (b), when the standard deviation of the angle is small, making the Euclidean approximation more valid. The PoE performs better in all cases. As expected, its advantage is the biggest in case (a), where the data was actually generated with a correlation between elements of the rotation matrix and height of the end-effector. Figure 3-19 shows the correlation between the height of the end-effector and the first element of the first row of the rotation matrix for case (a).

## 3.6 Conclusion

We proposed a framework based on products of experts to encode distributions in robotics. We demonstrated the pertinence of the model in various applications, and showed that this framework can be linked to many existing learning from demonstration representations and methods. By incorporating robot knowledge as task spaces, we showed that such approach is more data-efficient than general approaches like variational autoencoder. We also discussed the promises that such approach

| Case | (a) | | (b) | | (c) | |
|---|---|---|---|---|---|---|
| | $p_{\text{value}}$ | $\text{MMD}_u^2$ | $p_{\text{value}}$ | $\text{MMD}_u^2$ | $p_{\text{value}}$ | $\text{MMD}_u^2$ |
| Vectorized Gaussian | 0.0001 | $2.0\text{e}^{-2}$ | 0.10 | $2.15\text{e}^{-4}$ | 0.0011 | $1.0\text{e}^{-2}$ |
| Riemannian manifold | 0.10 | $1.03\text{e}^{-3}$ | 0.17 | $1.5\text{e}^{-4}$ | 0.29 | $2.03\text{e}^{-4}$ |
| **PoE** | **0.44** | $\mathbf{1.5e^{-3}}$ | **0.38** | $\mathbf{-3.9e^{-5}}$ | **0.68** | $\mathbf{-1.04e^{-3}}$ |

**Table 3.5:** Quantitative results for the task with a correlation between position and orientation. The table shows $p_{\text{value}}$ and maximum mean discrepancy $\text{MMD}_u^2$ measures between the dataset and the different models for the different cases.

hold to tackle wide-ranging data problems in robotics, by providing a framework that can start learning from few trials or demonstrations, but that it rich enough to progress once more data are available. Indeed, since PoEs offer the flexibility to learn parametrized task spaces as neural networks, the approach can span a wide range of problems, from small datasets with significant a priori knowledge to bigger datasets with less structured models. In the experiments, we validated that PoEs offer substantial improvements over approaches in which models are learned separately, emphasizing the capability of the approach to uncover tasks masked by kinematic limitations or by the resolutions of higher-level objectives.

# 4

# Product of policies

This chapter addresses the second part of the research question: how to preserve and exploit the information about the structure of the kinematic chain and the dynamics when learning movement primitives in several task spaces? The solution proposed in this chapter is to define a Gaussian controller in each task space. The robot is controlled with the fusion of these controllers, as a product of Gaussians. These controllers are optimized such that the distribution of trajectories executed on the robot is similar to the distributions of demonstrations, compared in the different task spaces.

In LfD, movements are commonly represented using movement primitives (MPs). They are used as building blocks of more complete skills in which they can be combined sequentially or simultaneously.

In LfD, the parameters of MPs are learned from a set of demonstrations. Ideally, motions synthesized from the MPs should match the distribution of demonstrations with the same variability [32, 91]. It can be later exploited for multiple usages, such as including additional constraints or objectives. Furthermore, keeping the variability is primordial when the demonstrations are used to initialize policy search [74]. Another desired feature of MPs is their adaptation to new situations or targets, such as moving objects. To that end, a common approach is to learn MPs in multiple parametric task spaces [16, 17, 92]. For example, the task spaces can be attached to objects of interest [16, 81, 86] such that the movements are analyzed under several coordinate systems. However, for computational reasons, many of these approaches make an assumption of independence between the MPs in the different spaces; those are learned independently and only combined at the controller level, which results in distortions in the synthesis.

A consistent framework for learning multiple models jointly is the product of experts (PoE) [52, 126]. Models with unnormalized likelihood like PoEs are used in robotics for inverse optimal control (IOC) [35, 61, 128]. However, they either rely on

expensive approximations of the normalizing constant [35] or learn only weights of predefined features [61]. Generative adversarial modelling [41] has been proposed as a more efficient approach with improved stability of the training [34, 49, 54].

In this chapter, we propose to train MPs within the generative adversarial framework which we call generative adversarial movement primitives (GAMP). We propose several adaptations of the discriminator and a particular parametrization of the policy to meet the requirements of LfD; as performing demonstrations on the physical systems is costly, we typically only have a few trajectories (from 5 to 20 depending on the complexity). Also, the training process should be interactive and thus relatively fast (from a few seconds to a few minutes). The proposed approach can be classified as model-based imitation learning [32]. Given a prior knowledge about the dynamics of the system, it uses model-based policy search to minimize an imitation cost. As shown in the experiments on the real robot, rough dynamic models make the process sample-efficient. We also propose a variant of the method to refine these models through executions on the real system. Our approach treats both epistemic uncertainty (coming from partial knowledge of the system) and aleatoric uncertainty (coming from stochasticity of the system), resulting in robust controllers. Finally, our framework aims to remain general and be compatible with multiple control strategies such as velocity, acceleration or torque control. It can also be used to train both time-dependent [17, 91] or time-independent policies [63].

**Organization of the chapter** In Sec. 4.1, the method to train MPs in the generative adversarial framework is described. In Sec. 4.1.1, the structure of a trajectory is presented as a state-space model, which includes a policy, an observation and a dynamic model. A policy structured as a product of Gaussian policies defined in different task space is presented. In Sec. 4.1.2, the additional discriminators are presented. In Sec. 4.2, we propose a method to treat uncertainties in the dynamic models. Some parametrization of the Gaussian policies are presented in Sec. 4.3. Finally, in Sec. 4.4, several experiments are proposed.

## 4.1 Generative adversarial training for product of policies

In the generative adversarial framework [41], a generator $G(\boldsymbol{z};\boldsymbol{\theta})$ is trained to transform input noise $p_z(\boldsymbol{z})$ into samples that look like the data distribution. To do this, a discriminator is trained in parallel to output the probability that a sample comes from the data rather than from the generator. On its side, the generator has to maximize the probability to mislead the discriminator. The generator and discriminator are typically neural networks trained with stochastic gradient descent (SGD). At each step of the training, the discriminator is optimized for a few steps of SGD and then one step is done for the generator.

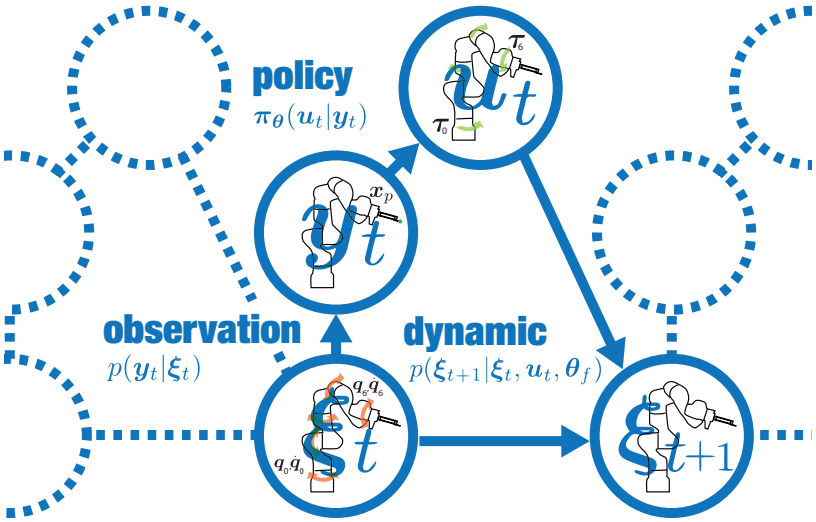



**Figure 4-1:** Dependencies in a trajectory model.

### 4.1.1 State-space generator

In order to generate trajectories, the considered generator is a state-space model defined by several components. We assume that we have access to a stochastic dynamic model of the robot $p(\boldsymbol{\xi}_{t+1}|\boldsymbol{\xi}_t, \boldsymbol{u}_t, \boldsymbol{\theta}_f)$ where $\boldsymbol{\xi}_t$ is the state at time t, $\boldsymbol{u}_t$ the control command and $\boldsymbol{\theta}_f$ the parameters of the dynamics model. If the robot is controlled with inverse dynamics, this model can be a simple integrator, but more complex models such as neural networks can be considered. If the dynamics model is not known or is uncertain, a distribution of parameters $p(\boldsymbol{\theta}_f)$ can be defined, under which the expected objective will be optimized, as we will see Sec. 4.2. The state of the system being sometimes not directly observed, an additional component that needs to be defined is a stochastic observation model $p(\boldsymbol{y}_t|\boldsymbol{\xi}_t)$ where $\boldsymbol{y}_t$ is an observation. Control commands are computed given this observation by a stochastic policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{y}_t)$ where $\boldsymbol{\theta}$ are the parameters of the policy. A distribution of trajectories $\boldsymbol{\tau} = \{\boldsymbol{y}_1, \boldsymbol{u}_1, \dots, \boldsymbol{y}_T, \boldsymbol{u}_T\}$ is then defined as a state-space model with

$$p(\boldsymbol{\tau}|\boldsymbol{\theta}_f, \boldsymbol{\theta}) = p(\boldsymbol{\xi}_1) \prod_{t=1}^{T} p(\boldsymbol{\xi}_{t+1}|\boldsymbol{\xi}_t, \boldsymbol{u}_t, \boldsymbol{\theta}_f)\, \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{y}_t)\, p(\boldsymbol{y}_t|\boldsymbol{\xi}_t). \tag{4.1}$$

The dependencies of this model are summarized in Fig. 4-1 as a graphical model.

An evident way to learn the MPs is to compute maximum likelihood estimation of $\boldsymbol{\theta}$ given a set of demonstrated trajectories. However, computing or maximizing this likelihood requires approximations or restricting assumptions [21, 26, 32]. In GANs, the likelihood is not modelled explicitly. It is just required to be able to draw samples from this density. Full sequences can be generated by forward sampling, by sampling each model after the other according to (4.1). Thus, great flexibility is

allowed for setting the dynamics, policy and observation models; they only have to be simple to sample from. Additionally, this process should be differentiable with respect to their respective parameters (e.g. $\boldsymbol{\theta}$ and $\boldsymbol{y}$ for the policy model) using the reparametrization trick [66]. If the observation model is not bijective, it might be impossible to retrieve the distribution of initial states of the demonstrations $p(\boldsymbol{\xi}_1)$. In this case, this distribution can be parametrized and optimized as well. For simplicity of the notation, an observable system with $\boldsymbol{\xi}_t = \boldsymbol{y}_t$ will be used for the derivations in the rest of the chapter, without loss of generality.

### Adaptation with products of Gaussian policies (PoGP)

While the $d_{\boldsymbol{\xi}}$-dimensional state $\boldsymbol{\xi}$ of a robotic manipulator is defined by its joint angles (and possibly velocities), movements are often best explained under several task spaces. Each task space $P$ is associated with a task map, which is a non-linear function $\mathcal{T}_{\boldsymbol{\xi},p} : \mathbb{R}^{d_{\boldsymbol{\xi}}} \to \mathbb{R}^{k^p_{\boldsymbol{\xi}}}$. Accordingly, a set of linear functions maps control commands $\boldsymbol{u}$ (joint velocities or acceleration) to their value in the different task spaces $\mathcal{T}_{\boldsymbol{u},p} : \mathbb{R}^{d_{\boldsymbol{u}}} \to \mathbb{R}^{k^p_{\boldsymbol{u}}}$. These transformations can be parametrized by the poses of an external object (which we will drop in the notation for simplicity) or by time, which we will denote with the superscript $t$. We propose to define a stochastic Gaussian policy in each of these task spaces as

$$\mathcal{T}^t_{\boldsymbol{u},p}(\boldsymbol{u}_t) \sim \mathcal{N}\Big(\boldsymbol{\mu}^t_p\big(\mathcal{T}^t_{\boldsymbol{\xi},p}(\boldsymbol{\xi}_t)\big), \boldsymbol{\Sigma}^t_p\big(\mathcal{T}^t_{\boldsymbol{\xi},p}(\boldsymbol{\xi}_t)\big)\Big). \tag{4.2}$$

In the most general case, $\boldsymbol{\mu}^t_p(\cdot)$ and $\boldsymbol{\Sigma}^t_p(\cdot)$ can be neural networks. In simpler cases, the policies can be proportional derivative controllers with a constant covariance. In Sec. 4.3, different parameterizations of the policy are given. The proposed overall policy is the fusion of the policies in the different task spaces, given as a product of linearly transformed Gaussians

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t|\boldsymbol{\xi}_t) \propto \prod_{p=1}^{P} \mathcal{N}\Big(\mathcal{T}^t_{\boldsymbol{u},p}(\boldsymbol{u}_t)\Big|\boldsymbol{\mu}^t_p\big(\mathcal{T}^t_{\boldsymbol{\xi},p}(\boldsymbol{\xi}_t)\big), \boldsymbol{\Sigma}^t_p\big(\mathcal{T}^t_{\boldsymbol{\xi},p}(\boldsymbol{\xi}_t)\big)\Big). \tag{4.3}$$

This product has a closed form expression as a Gaussian and can be sampled directly. Many works [17, 92, 95] have a final controller of this form, but it is computed by using expert policies that have been learned independently. In [95], we additionally proposed to exploit the uncertainty (coming from the lack of data) of each controller by using Bayesian models.

The same parametrization is also proposed as a Riemannian Motion Policy in [101]. They define a set of operators as *addition*, *pullback* and *pushforward* that are strictly equivalent to the product, inverse and forward linear transformations of Gaussian respectively, as presented in Sec. 2.4.1. However, their approach loses the statistical interpretation that is useful for developing learning algorithms and generating stochasticity.

### 4.1.2 Including LfD generative models as close-to-optimal discriminators

Besides training a discriminator as a neural network $D(\boldsymbol{\tau})$, we propose to include standard generative models used in LfD. They are usually trained in closed form or with very efficient procedures like EM. This addition is motivated by a dramatically increased stability and speed of the training procedure. We propose to include a second discriminator $D_q(\boldsymbol{\tau})$ which is multiplied to the original one. This discriminator consists of two approximate distributions $q_{\text{samples}}$ and $q_{\text{data}}$ learned with standard LfD generative models as [16, 17, 91]. In this chapter, this discriminator is called a *close-to-optimal discriminator* with

$$D_q(\boldsymbol{\tau}) = \frac{q_{\text{data}}(\boldsymbol{\tau})}{q_{\text{data}}(\boldsymbol{\tau}) + q_{\text{samples}}(\boldsymbol{\tau})}. \tag{4.4}$$

In an optimal discriminator, $q_{\text{data}}$ and $q_{\text{samples}}$ would be the exact distributions, not the approximate ones which are improperly normalized. But if we were able to model explicitly this likelihood, the generative adversarial approach would not be needed. The class of distributions $q$ we propose to use typically drops some dependencies or do not integrate to one on the space of trajectories. Another formulation of this problem is that the system is underactuated[1] [88]. Directly used as generative models, where the dropped dependencies are only restored at the synthesis phase [16, 90], these models induce distortions, as discussed in [126]. These distortions are extensively reduced if these models are used as classifiers in the context of generative adversarial learning; both the samples from the generator and the dataset are compared under the same approximations while the feasibility and dependencies are ensured by the generator (4.1).

We propose to train the approximate distribution $q_{\text{data}}$ once at the beginning of the learning process. The approximate distribution of samples $q_{\text{samples}}$ is updated with maximum likelihood before each step of gradient descent of the policy parameters, see Alg. 3. As it might be costly to generate many samples from the generator at each iteration, $q_{\text{samples}}$ can be learned incrementally with stochastic updates of maximum likelihood. Such updates can be derived for expectation maximization (EM), closed-form maximum likelihood (e.g. Gaussian distribution) or variational inference [55] (in the case where $q_{\text{data}}$ and $q_{\text{samples}}$ are Bayesian models whose posterior distribution is estimated).

Many possibilities are offered for choosing the family of approximate distributions $q$. A very simple choice, if the trajectories are all aligned in time, is to use a factorized Gaussian distribution as

$$q(\boldsymbol{\tau}) = \prod_{t=1}^{T} \mathcal{N}\Big( \begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{u}_t \end{bmatrix} \Big| \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t \Big). \tag{4.6}$$

Matching factorized Gaussian distributions is also done in [32] in a similar context.

[1] All the trajectories of the distribution are not feasible.

**Algorithm 3:** Stochastic gradient descent generative adversarial training of movement primitives

Compute maximum likelihood of $q_{\text{data}}$ on the $N$ demonstrations $\{\hat{\boldsymbol{\tau}}^{(i)}\}_{i=1}^{N}$
**for** *number of training iterations* **do**
  Sample from (4.1) $M$ trajectories $\{\boldsymbol{\tau}^{(i)}\}_{i=1}^{M}$
  Apply (stochastic) maximum likelihood updates on $q_{\text{samples}}$ given $\{\boldsymbol{\tau}^{(i)}\}_{i=1}^{M}$
  Update policy parameters $\boldsymbol{\theta}$ by descending the stochastic gradient :

$$\nabla_{\boldsymbol{\theta}} \frac{1}{M} \sum_{i=1}^{M} \Big( -\log\big(q_{\text{data}}(\boldsymbol{\tau}^{(i)})\big) + \log\big(q_{\text{data}}(\boldsymbol{\tau}^{(i)}) + q_{\text{samples}}(\boldsymbol{\tau}^{(i)})\big) \tag{4.5}$$

**end**

However, $q$ is represented explicitly by using Gaussian process dynamics models and moment matching approximations [21]. If the trajectories have particular correlations across time (that are not due to the dynamics), probabilistic movements primitives (ProMP)[91] can be used instead. In order to provide adaptation to parametrized task spaces, the discriminator can additionally compare the trajectories in these task spaces as

$$q(\boldsymbol{\tau}) = \prod_{t=0}^{T} \left( \mathcal{N}\Big( \begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{u}_t \end{bmatrix} \Big| \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t \Big) \prod_{p=1}^{P} \mathcal{N}\Big( \begin{bmatrix} \mathcal{T}^t_{\boldsymbol{\xi},p}(\boldsymbol{\xi}_t) \\ \mathcal{T}^t_{\boldsymbol{u},p}(\boldsymbol{u}_t) \end{bmatrix} \Big| \boldsymbol{\mu}^p_t, \boldsymbol{\Sigma}^p_t \Big) \right). \tag{4.7}$$

In this case, even if the approximate distributions $q$ are Gaussians, the discriminator would be able to distinguish between more complex distributions, as the comparison is done under non-linear transformations. In the case where the trajectories cannot be time-aligned or that the targeted distribution is multimodal, more complex models as hidden Markov models [16] or Gaussian mixture models [95] can be used, with the density

$$q(\boldsymbol{\tau}) = \prod_{t=0}^{T} \left( \sum_{k=0}^{K} \pi_k \mathcal{N}\Big( \begin{bmatrix} \boldsymbol{\xi}_t \\ \boldsymbol{u}_t \end{bmatrix} \Big| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \Big) \right). \tag{4.8}$$

They can be trained very efficiently with EM in a few milliseconds.

By using Gaussian models, which have quadratic log-likelihood, the gradients are well-behaved, leading to fast convergence. At initialization, our generator samples trajectories $\boldsymbol{\tau}^{(i)}$ very far from $q_{\text{data}}$ but close from $q_{\text{samples}}$. The term $\frac{1}{M}\sum_{i=1}^{M} -\log\big(q_{\text{data}}(\boldsymbol{\tau}^{(i)})\big)$ in Alg. 4 would dominate the gradient of the cost, which would be close to quadratic. For practical and stability reasons, we propose to train the policy first by considering only the additional classifier using approximate distributions. Then, the neural network classifier is only used for additional refinements and with smaller learning rates. In this case, the neural network is just helping the additional classifier

to distinguish features that are not encoded in the approximate distributions $q$.

## 4.2 Robustness and unknown dynamics

In this section, we propose a strategy to learn robust policies in case of changing or unknown dynamics. This problem is not directly related to the main research question of this thesis. However, providing an answer is necessary for the implementation of the proposed method on real systems. So far, we have considered fixed parameters $\boldsymbol{\theta}_f$ of the stochastic dynamic system, which is not realistic. It can result in a poor matching of the trajectory distribution in the case where the model of dynamics $\boldsymbol{\theta}_f$ does not match the real system. In the worst case, the distributions can completely diverge and executing the policy can be dangerous. In other cases, the problems can be less disastrous and more subtle. For example, if the model overestimates the stochasticity of the system, the rollouts on the real system would have lower variance than the demonstrations. In this case, the system will rely too much on the stochasticity of the environment to create variability. A robust policy has to match the distribution of trajectories for a distribution of parameters $p(\boldsymbol{\theta}_f)$. This distribution can be either a hand-tuned prior distribution, a posterior distribution if the dynamics are learned with Bayesian methods or a set of parameters $\{{\boldsymbol{\theta}_f}^{(j)}\}_{j=1}^{L}$ if they are learned with ensemble methods.

We propose to condition the discriminator on $\boldsymbol{\theta}_f$, which means that this value should be fed to it together with the samples. As the true system is not known when executing the policy, this latter should not depend on the parameters of the dynamics. Giving access to the model parameters on which are generated the samples to the discriminator only forces the policy to match the distribution of data under a distribution of dynamic parameters, ensuring robustness.

In order to use the additional discriminators $D_q(\boldsymbol{\tau})$ proposed in Section 4.1.2, two alternatives are possible. The choice mainly depends on the trade-off between robustness and computation time. In both cases, multiple approximate models $\{{q_{\text{samples}}}^{(j)}\}_{j=1}^{L}$ are learned on a batch of dynamic parameters $\{{\boldsymbol{\theta}_f}^{(j)}\}_{j=1}^{L}$. In the case of privileging robustness, the $L$ dynamic parameters are drawn from their distribution before each iteration of gradient descent. In this case, enough samples should be drawn from (4.1) in order to compute the maximum likelihood of the approximate distribution ${q_{\text{samples}}}^{(j)}$. The stochastic updates are not allowed as the $L$ model from the previous iterations do not correspond anymore. When it is too costly to sample enough trajectories to perform complete maximum likelihood of $q$, the $L$ models can be changed only after a given number of iterations or even kept fixed throughout the learning process. This solution is also natural if the parameters are learned by an ensemble method. The procedure is more formally presented in Alg. 4.

The parameters of the system $\boldsymbol{\theta}_f$ can be also learned or refined. For model-based policy search (from which our approach is a particular case), a key requirement of the dynamic model is its ability to produce good long-term predictions. Many approaches optimize a one-step-ahead model, for example by maximizing the likelihood of $p(\boldsymbol{\xi}_{t+1}|\boldsymbol{\xi}_t, \boldsymbol{u}_t)$. However, due to modelling errors, false assumptions on the model

**Algorithm 4:** Robust generative adversarial training of movement primitives

Compute maximum likelihood of $q_{\text{data}}$ on the $N$ demonstrations $\{\hat{\boldsymbol{\tau}}^{(i)}\}_{i=1}^{N}$
**for** *number of training iterations* **do**
  **for** $L$ *dynamic models* $\{{\boldsymbol{\theta}_f}^{(j)}\}_{j=1}^{L}$ **do**
    Sample from (4.1) $m$ trajectories $\{\boldsymbol{\tau}^{(i,j)}\}_{i=1}^{M}$
    Apply (stochastic) maximum likelihood update on ${q_{\text{samples}}}^{(j)}$ given $\{\boldsymbol{\tau}^{(i,j)}\}_{i=1}^{M}$
  **end**
  Update **global** policy parameters $\boldsymbol{\theta}$ by descending the stochastic gradient :

$$\nabla_{\boldsymbol{\theta}} \frac{1}{ML} \sum_{j=1}^{L} \sum_{i=1}^{M} \Big( -\log\left(q_{\text{data}}(\boldsymbol{\tau}^{(i,j)})\right) + \log\left(q_{\text{data}}(\boldsymbol{\tau}^{(i,j)}) + {q_{\text{samples}}}^{(j)}(\boldsymbol{\tau}^{(i,j)})\right) \tag{4.9}$$

**end**

and noisy or partial observation model, this approach tends to produce brittle predictions which diverge quickly from the real system [12]. Robust approaches as [26] optimize the likelihood of full sequences of observations $p(\boldsymbol{y}_1, \dots, \boldsymbol{y}_T)$ marginalized on the sequence of latent states $\{\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_T\}$. Our approach optimizes the same objective but in the generative adversarial framework, which does not require to model explicitly the marginal distribution of observation. It makes very little assumptions on the dynamic, observation and policy models at the expense of a higher computational cost.

The approach to refine the dynamic parameters is presented in the case of the close-to-optimal discriminator $D_q(\boldsymbol{\tau})$. The proposed process alternates between learning parameters of the policy using Alg. 4 and executing this policy on the real system to update the dynamic model. To do so, an additional approximate distribution $q_{\boldsymbol{\theta},\text{data}}$ is introduced. It models the distribution of trajectories executed on the real system with the inferred policy parameters $\boldsymbol{\theta}$ of the previous step. The distribution of trajectories $q_{\text{samples}}$ sampled with the inferred policy and model of the dynamics is optimized to match this new distribution $q_{\boldsymbol{\theta},\text{data}}$. This time, the gradient is computed with respect to the parameters of the dynamic model $\boldsymbol{\theta}_f$. For increased robustness, it is better to keep multiple dynamic parameters $\{{\boldsymbol{\theta}_f}^{(j)}\}_{j=1}^{l}$ and train them in parallel as an ensemble method, see Alg. 5. For example, if the inertia of the robot is not known, the multiple initial dynamic parameters could reflect this. The policy training at the first iteration (before executing on the real system) would be more conservative (high feedback terms) to accommodate this uncertainty. In the case where a single dynamic parameter was used in the policy optimization, the execution on the robot can be disastrous (for example if the inertia matrix was overestimated).

Our approach has the advantages to treat separately epistemic uncertainty (coming from partial knowledge of the system) and aleatoric uncertainty (coming from

**Algorithm 5:** Refining dynamic models with an ensemble method

Compute maximum likelihood of $q_{\text{data}}$ on the $N$ demonstrations $\{\hat{\boldsymbol{\tau}}^{(i)}\}_{i=1}^{N}$
**for** *number of real-system iterations* **do**
  Start with $L$ initial guesses $\{{\boldsymbol{\theta}_f}^{(j)}\}_{j=1}^{L}$ of system dynamics
  Update policy with **Algorithm 4**
  Sample $n$ trajectories $\{\hat{\boldsymbol{\tau}}_{\boldsymbol{\theta}}^{(i)}\}_{i=1}^{N}$ on the **real system** given current policy parameters $\boldsymbol{\theta}$
  Compute maximum likelihood of the distribution of new trajectories $q_{\boldsymbol{\theta},\text{data}}$
  **for** *number of training iterations* **do**
    **for** $L$ *dynamic models* $\{{\boldsymbol{\theta}_f}^{(j)}\}_{j=1}^{L}$ **do**
      Sample from (4.1) $M$ trajectories $\{\boldsymbol{\tau}^{(i,j)}\}_{i=1}^{M}$
      Apply (stochastic) maximum likelihood update on ${q_{\text{samples}}}^{(j)}$ given $\{\boldsymbol{\tau}^{(i,j)}\}_{i=1}^{M}$
      Update **dynamic model parameters** ${\boldsymbol{\theta}_f}^{(j)}$ using the stochastic gradient:

$$\nabla_{{\boldsymbol{\theta}_f}^{(j)}} \frac{1}{M} \sum_{i=1}^{M} -\log\left(q_{\boldsymbol{\theta},\text{data}}(\boldsymbol{\tau}^{(i,j)})\right) + \log\left(q_{\boldsymbol{\theta},\text{data}}(\boldsymbol{\tau}^{(i,j)}) + {q_{\text{samples}}}^{(j)}(\boldsymbol{\tau}^{(i,j)})\right) \tag{4.10}$$

    **end**
  **end**
**end**

stochasticity of the system). Also, by using neural networks to model the dynamics, environment with heteroscedastic noise can be modeled, as opposed to standard Gaussian process models.

## 4.3 Robotic policies

In this section, we propose several convenient parameterizations of policies that can be used in our framework. As proposed in Sec. 4.1.1, the policies used in this chapter are defined in $P$ task spaces $\mathcal{T}_p : \mathbb{R}^d \to \mathbb{R}^{k_p}$. We denote $\boldsymbol{x}_p = \mathcal{T}_p(\boldsymbol{q})$ the value in task space $p$ and $\boldsymbol{J}_p = \partial\mathcal{T}_p/\partial\boldsymbol{q}$ its Jacobian. The velocity $\dot{\boldsymbol{x}}_p$ and acceleration $\ddot{\boldsymbol{x}}_p$ in the task space are

$$\dot{\boldsymbol{x}}_p = \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4.11}$$

$$\ddot{\boldsymbol{x}}_p = \boldsymbol{J}_p(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_p(\boldsymbol{q})\dot{\boldsymbol{q}} \approx \boldsymbol{J}_p(\boldsymbol{q})\ddot{\boldsymbol{q}}. \tag{4.12}$$

The relation between the joint torque $\boldsymbol{\tau}$ and the generalized force is

$$\boldsymbol{J}_p^{\top}(\boldsymbol{q})\,\mathbf{f}_p = \boldsymbol{\tau}. \tag{4.13}$$

| Control strategy | Velocity | Acceleration | Torque |
|---|---|---|---|
| State $\boldsymbol{\xi}$ | $\boldsymbol{q}$ | $\begin{bmatrix}\boldsymbol{q}\\ \dot{\boldsymbol{q}}\end{bmatrix}$ | $\begin{bmatrix}\boldsymbol{q}\\ \dot{\boldsymbol{q}}\end{bmatrix}$ |
| Control command $\boldsymbol{u}$ | $\hat{\dot{\boldsymbol{q}}}$ | $\hat{\ddot{\boldsymbol{q}}}$ | $\boldsymbol{\tau}$ |
| Transform $\mathcal{T}_{\boldsymbol{\xi},p}$ | $\mathcal{T}_p(\boldsymbol{q})$ | $\begin{bmatrix}\mathcal{T}_p(\boldsymbol{q})\\ \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}}\end{bmatrix}$ | $\begin{bmatrix}\mathcal{T}_p(\boldsymbol{q})\\ \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}}\end{bmatrix}$ |
| Transform $\mathcal{T}_{\boldsymbol{u},p}$ | $\boldsymbol{J}_p(\boldsymbol{q})\hat{\dot{\boldsymbol{q}}}$ | $\boldsymbol{J}_p(\boldsymbol{q})\hat{\ddot{\boldsymbol{q}}}$ | $\boldsymbol{J}_p^{\top\dagger}(\boldsymbol{q})\boldsymbol{\tau}$ |

**Table 4.1:** Equivalences between abstract state $\boldsymbol{\xi}$, control command $\boldsymbol{u}$, task-spaces transform and robotic variables.

These relations are used to define an equivalence between variables used in the above and the different control strategies. The equivalences are reported in Table 4.1 for different control strategies. For velocity and acceleration control, $\hat{\dot{\boldsymbol{q}}}$ and $\hat{\ddot{\boldsymbol{q}}}$ are reference values that are tracked by lower-level controller, as inverse dynamics [84].

These relations do not need to be exact. They are just parameterizations of the policy which give a better structure to the problem to facilitate the training phase and increase generalization capabilities. Simplifications, such as dropping $\dot{\boldsymbol{J}}_p(\boldsymbol{q})$ for acceleration control, can be done to speed up the computation of the stochastic gradient while training.

The policies can be parametrized in different ways. For time-dependent policies, a solution is a feedback controller with time-varying gains and feed-forward terms. These controllers are very usual in LfD [16, 91] and are also solutions of linear-quadratic tracking problems [15]. They can be used both for velocity control

$$\boldsymbol{\mu}_p^t(\mathcal{T}_{\boldsymbol{\xi},p}^t(\boldsymbol{\xi})) = -\boldsymbol{K}_p(t)\mathcal{T}_p^t(\boldsymbol{q}) + \boldsymbol{d}_p(t), \tag{4.14}$$

$$\boldsymbol{\Sigma}_p^t(\mathcal{T}_{\boldsymbol{\xi},p}^t(\boldsymbol{\xi})) = \hat{\boldsymbol{\Sigma}}_p(t), \tag{4.15}$$

or acceleration and force with

$$\boldsymbol{\mu}_p^t(\mathcal{T}_{\boldsymbol{\xi},p}^t(\boldsymbol{\xi})) = -\boldsymbol{K}_p(t)\mathcal{T}_p^t(\boldsymbol{q}) - \boldsymbol{K}_p^v(t)\boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{d}_p(t). \tag{4.16}$$

Continuous values for the parameters depending on time $t$ can be induced by linear basis functions or simple multilayer perceptrons (MLP). Gains $\boldsymbol{K}$ can be parametrized in several restrictive ways depending on the assumptions on the system. It can help at stabilizing and speeding up the training phase as well as at providing more safety on the robot.

Time-independent policy can be defined with MLPs that output the parameters

of the Gaussian policy

$$\boldsymbol{\mu}_p^t(\mathcal{T}_{\boldsymbol{\xi},p}^t(\boldsymbol{\xi})) = \mathrm{F}_{\boldsymbol{\mu}}\big(\mathcal{T}_p^t(\boldsymbol{q})\big), \quad \boldsymbol{\Sigma}_p^t(\mathcal{T}_{\boldsymbol{\xi},p}^t(\boldsymbol{\xi})) = \mathrm{F}_{\boldsymbol{\Sigma}}\big(\mathcal{T}_p^t(\boldsymbol{q})\big). \tag{4.17}$$

Covariance matrices can be parametrized by their Cholesky decomposition or using the matrix exponential of another symmetric matrix. Time-independent policies have also been modeled by computing conditional distributions in Gaussian mixture models [63, 95]. This latter approach is extremely fast to train from data but its gradient is not well-behaved for optimization.

## 4.4 Experiments

### 4.4.1 Time-dependent policy

In this first experiment with synthetic systems, we consider a simulated 2D unit mass system with a discretization of time $dt = 0.01$ s. The state $\boldsymbol{\xi} \in \mathbb{R}^4$ is composed of its position and velocity, and the control command $\boldsymbol{u} \in \mathbb{R}^2$ is the simulated force. The dataset are letters from the alphabet [16]. For each letter, $N = 13$ time-aligned demonstrations of $T = 200$ timesteps are given. Demonstrations are shown in Fig. 4-2-(*left*) for letter "N". A time-dependent feedback Gaussian policy as (4.16) is used. The gain matrices and feed-forward terms are parametrized with time-dependent basis functions. Gains are parametrized in several ways, which has some influence on training time but none on the final quantitative results. For the evaluations, $\boldsymbol{K}_p(t)$ and $\boldsymbol{K}_p^v(t)$ were chosen as diagonal matrices with positive elements. In this first experiment, the policy is not a PoGP as it is defined only in the original state space. The approximate distributions used for the discriminator are factorized Gaussians as (4.6). Each letter was trained for 10 s. The approach presented in Sec. 4.2 was used on a distribution of dynamic parameters $p(\boldsymbol{\theta}_f)$ which include different values of Gaussian perturbations acting on the control command and on the initial state.

We compare our approach with ProMP [91] and hybrid approaches that learn a time-dependent distribution of states with either Gaussian mixture regression (GMR + LQT) [17] or hidden Markov models (HMM + LQT) [16] and use linear quadratic tracker to regenerate continuous trajectories.

Given the controller computed for each model, we evaluated rollouts in 3 situations. In the first case, ${\boldsymbol{\theta}_f}^{(1)}$, the system is deterministic. In the second case, ${\boldsymbol{\theta}_f}^{(2)}$, uncorrelated Gaussian perturbations in force of standard deviation of 10 N are injected. In the third case, ${\boldsymbol{\theta}_f}^{(3)}$ Gaussian noise on the initial position of standard deviation of 0.035 m is applied. Two metrics are used to compare the demonstrations with the synthesized samples. The first one evaluates if the mean motion is well reproduced. For each letter, the mean squared error (MSE) is computed between the mean trajectories (position and velocity only) over the $N = 13$ demonstrations and the mean over 20 samples from the model. The second metric evaluates if the full distribution is well reproduced. The Bhattacharyya distance (BD) is computed over a Gaussian approximation of the distribution of demonstrations and samples.

The results are reported in Table 4.2 for each case of dynamics, each model and the

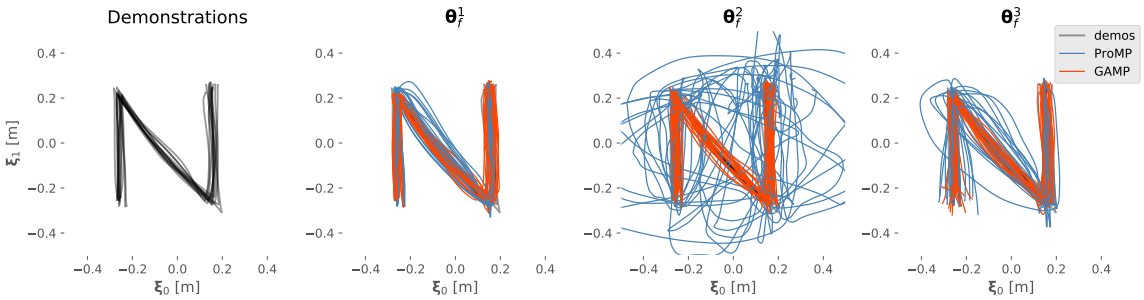



**Figure 4-2:** Demonstrations and reproductions using ProMP and GAMP for different stochasticity of the environment, parametrized with ${\boldsymbol{\theta}_f}^{(\cdot)}$. The same controllers are used in the 3 situations.

| **Metrics** | **Mean squared error** | | | **Bhattacharyya distance** | | |
|---|---|---|---|---|---|---|
| Environment $\boldsymbol{\theta}_f$ | ${\boldsymbol{\theta}_f}^1$ | ${\boldsymbol{\theta}_f}^2$ | ${\boldsymbol{\theta}_f}^3$ | ${\boldsymbol{\theta}_f}^1$ | ${\boldsymbol{\theta}_f}^2$ | ${\boldsymbol{\theta}_f}^3$ |
| GAMP (ours) | $\mathbf{0.18 \pm 0.06}$ | $\mathbf{0.19 \pm 0.08}$ | $\mathbf{0.23 \pm 0.11}$ | $\mathbf{0.13 \pm 0.04}$ | $\mathbf{0.13 \pm 0.04}$ | $\mathbf{0.11 \pm 0.04}$ |
| ProMP [91] | $0.38 \pm 0.27$ | $1.15 \pm 0.55$ | $0.60 \pm 0.42$ | $0.24 \pm 0.10$ | $0.90 \pm 0.30$ | $0.29 \pm 0.10$ |
| GMR + LQT [17] | $0.31 \pm 0.14$ | $0.34 \pm 0.13$ | $0.31 \pm 0.13$ | $0.40 \pm 0.07$ | $0.30 \pm 0.06$ | $0.40 \pm 0.07$ |
| HMM + LQT [16] | $1.58 \pm 0.48$ | $1.20 \pm 0.47$ | $1.20 \pm 0.48$ | $0.76 \pm 0.21$ | $0.58 \pm 0.21$ | $0.75 \pm 0.21$ |

**Table 4.2:** Mean squared error and Bhattacharyya distance between demonstrations and samples of different models. The samples are generated with three different dynamic parameters.

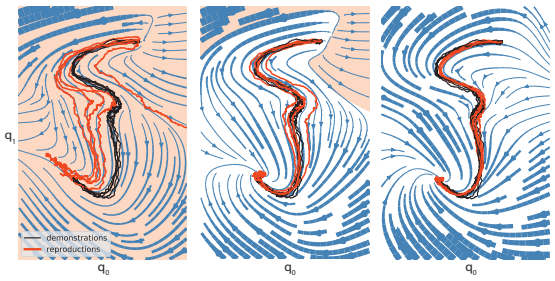
two metrics. The mean value and standard deviation of these metrics over the whole alphabet is given. Demonstrations and synthesized samples from ProMP and GAMP are shown in Fig. 4-2 for each stochasticity. ProMP and GAMP have similar results in terms of MSE and BD in case of no stochasticity ${\boldsymbol{\theta}_f}^{(1)}$. The ProMP controller derived in [91] assumes known dynamics and stochasticity in order to match the distribution of demonstrations. Our approach can generate a controller that matches the distribution, for a distribution of stochasticity of the environment. A more robust controller for ProMP can be derived in our framework by using $q_{\text{data}}$ and $q_{\text{samples}}$ as ProMP distributions. The approach using GMR + LQT is robust to perturbations and performs well in terms of MSE. However, the distribution is not matched very well. The velocity of the rollouts have the same variance as the demonstrations but the positions tend to shrink on the mean trajectory. This is due to the assumption of independence between two consecutive states that are ignored in GMR and HMM and that are later restored with LQT. The proper way to generate the matching distribution would be to use IOC with LQT [34] or trajectory HMM [125].

### 4.4.2 Time-independent policy in two task spaces

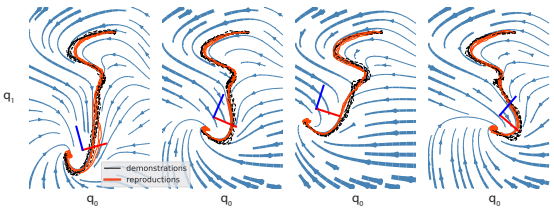
In the second experiment, we consider a time-independent policy that should adapt to a moving object, as shown in Fig. 4-3b. The system considered is a simple integrator

with possible perturbations. The dataset consists of a smooth blending between the letters "S" and "J". The letter "J" should move according to an object, displayed as a coordinate system in Fig. 4-3b. Ten different positions and orientations of the object are given, and for each of which $N = 10$ demonstrations of $T = 400$ timesteps are performed. The dataset was randomly split into 5 situations to train and 5 to test the generalization. The policy is the product of two time-independent Gaussian policies given by an MLP as (4.17). These two policies are defined in a different task space: the first one in a fixed task space, and the second one, projected in the coordinate system of the moving object. The MLPs have both 2 hidden layers of 150 units with *tanh* activation and output a state-dependent Gaussian with a full covariance. The covariance is parametrized by its matrix logarithm. Even if the demonstrations are full and aligned, we discard this information for training, and randomly split them in small chunks. The approximate distributions used for the discriminator are Gaussian mixture models with $K = 20$ as (4.8) in each task space. Before each update of policy parameters $\boldsymbol{\theta}$, 10 steps of EM are performed with 1000 points each. Every 50 steps of policy parameters update, the mixture models are reinitialized with k-means to avoid local minima. The policy parameters are initialized by maximum likelihood of the policy density on pairs of $\{\boldsymbol{\xi}, \boldsymbol{u}\}$ for 5 s of stochastic gradient descent. This initialization corresponds to a policy imitation objective, which is known to produce brittle policy [103]. Fig. 4-3a-*(left)* shows the policy after initialization and the dangers of drifting away from the training data. The models are further trained for 15 s in the generative adversarial network. Two alternatives are considered. In the first (GAMP), the system is assumed to be deterministic during training. In the second (GAMP + noise), small perturbations in the initial state of the chunks are simulated. The policy is then trained to look like the demonstrations, even with noise, which results in more robustness. The differences between the policy learned in these two cases are shown in Fig. 4-3a-*(middle and right)*. We also consider another policy, where the adaptation to the moving object is given by the neural network instead of the usage of multiple task spaces. In this case, the MLP defining the Gaussian policy has an additional input. It is the position and vectorized rotation matrix of the object.

As an evaluation, we produce full rollouts from the initial states of the demonstrations. We compute the mean absolute error (MAE) over the position of the whole rollout and the closest demonstration. The mean value and its standard deviation over the training and testing situations are given in Table 4.3. The adaptation with the use of task spaces and MLP perform the same on the training set but the former generalizes better. In robotics, many adaptations of movements can be understood easily by projecting them into several coordinate systems. The two approaches can also be combined in the case where the definition of multiple task spaces is not sufficient. The addition of noise in the initial state makes the GAMP more robust. They also generalize better to new situations. When using the imitation cost only, the trajectories diverge, as shown in Fig. 4-3a-(*left*). In the case of MLP adaptation, they diverge extremely fast, leading to an enormous cost. When using the imitation cost only, the benefits of defining two policies on low-dimensional task space instead of a unique policy with an additional vector of inputs are important. The fusion of two




(a) Demonstrations and samples with several learning strategies. The area leading to divergence are highlighted. (*left*) The policy is trained only by maximizing its own likelihood, as in policy imitation. (*middle*) The policy is trained with the method presented in this chapter, to maximize the likelihood of the trajectories. (*right*) Additional perturbations are added in the initial state of each chunk, leading to a more robust policy.




(b) Demonstrations and samples in several situations. The transformation of the moving object is indicated with a coordinate system. The flow field displays only the mean value of the policy.

**Figure 4-3:** Illustration of time-independent policies as flow fields.

| | **Training** | **Testing** |
|---|---|---|
| *Task spaces adaptation* | | |
| GAMP + noise | $\mathbf{0.016 \pm 0.003}$ | $\mathbf{0.025 \pm 0.009}$ |
| GAMP | $0.019 \pm 0.008$ | $0.075 \pm 0.13$ |
| Imitation | $0.251 \pm 0.346$ | $1.629 \pm 2.124$ |
| *MLP adaptation* | | |
| GAMP + noise | $0.017 \pm 0.002$ | $0.078 \pm 0.014$ |
| GAMP | $0.025 \pm 0.008$ | $0.086 \pm 0.028$ |
| Imitation | $3.2e6 \pm 8.7e6$ | $1.3e6 \pm 6.4e6$ |

**Table 4.3:** Mean absolute error (MAE) between the whole rollout and the closest demonstrations for situations in the training and testing set. The red colour indicates huge errors because of divergence.

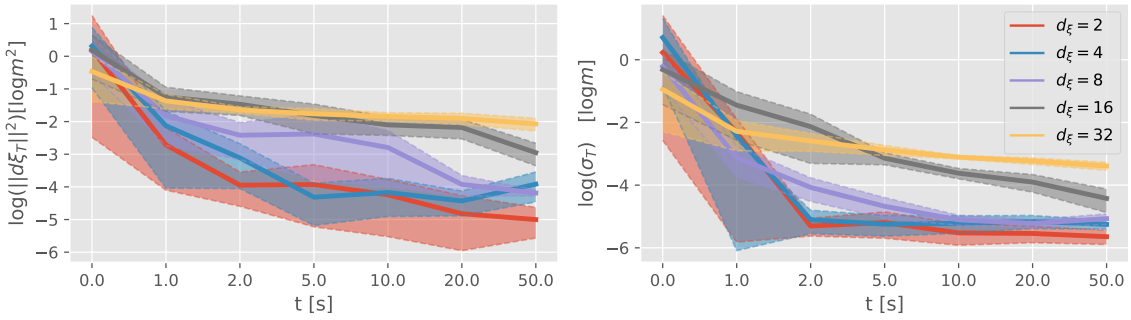


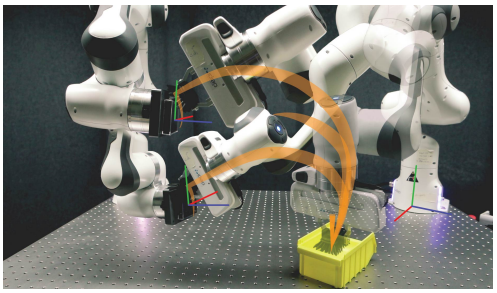

**Figure 4-4:** Evaluations of the robustness for different dimensionality of state $d_{\boldsymbol{\xi}} \in \{2, 4, 8, 16, 32\}$. (*left*) Log mean distance between the final state of the demonstrations and of 10 rollouts. (*right*) Log mean standard deviation of the final state of the 10 rollouts.

robust policies will tend to be more robust than a policy that can change completely for each new vector defining the situation.

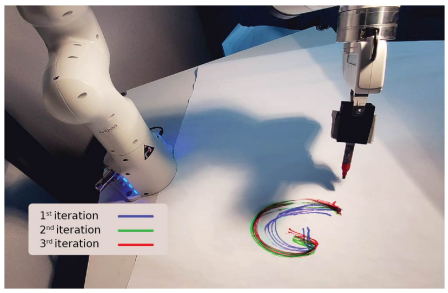
The problem of learning robust policy is very difficult [103], [63]. Our approach gives no guarantees that the system cannot diverge, but the cost of mimicking the distribution of trajectories greatly increases the robustness. By injecting noise and training with stochastic gradient descent for a sufficient amount of time, the system is expected not to diverge. We performed an additional test for showing that this also applies to higher-dimensional systems, given slightly longer optimization. We created higher-dimensional dataset by randomly concatenating letters up to $\boldsymbol{\xi} \in \mathbb{R}^{32}$. Two metrics were used to check the divergence. The first one is the mean distance between the final state of the demonstrations and 10 rollouts. The second one is the standard deviation of the final state for each rollout. The rollouts were executed for twice the horizon of the demonstrations, to check if the system drifts further and by adding perturbation on the initial state. These two metrics were evaluated for 5 random concatenations of letters for each dimension. The metrics were evaluated just after initialization using the policy imitation cost and several times during 50 s of training. Results are reported in Fig. 4-4. They show that, as expected, high-dimensional systems tend to be more difficult to train. However, after a few seconds of optimization, no more trajectories were diverging even for high-dimensional system. They all converged within an area of at worse 0.05 m of standard deviation, while the scale of the workspace is about 1 m.

### 4.4.3 Acceleration control with adaptation

In this experiment, the robot has to paint a box held by another robot. It needs to dip the brush in a paint container that is always at the same place and then wipe the box whose orientation and position can vary, see Fig. 4-5a. The dataset consists of $N = 7$ time-aligned demonstrations of 4.5 s with a discretization of time $dt = 0.02$ s. In each demonstration, the box has a different pose. We consider that the control commands are the joint accelerations $\ddot{\boldsymbol{q}} \in \mathbb{R}^7$ and the state $\boldsymbol{\xi} \in \mathbb{R}^{14}$ consists



(a) Painting task with adaptation to varying poses.




(b) Drawing task in varying environments.

**Figure 4-5:** Two illustrative tasks are performed on the robot to demonstrate the adaptation and robustness of the approach. A 7-DoF Panda robot controlled with acceleration *(a)* and torque *(b)* control is used.

| | **Training** | **Testing** $\sigma_c = 1$ | **Testing** $\sigma_c = \sqrt{2}$ | **Testing** $\sigma_c = 2$ |
|---|---|---|---|---|
| ProMP conditioning [91] | $\mathbf{0.35 \pm 0.13}$ | $1.72 \pm 2.29$ | $8.18 \pm 15.86$ | $13.21 \pm 22.66$ |
| GAMP | $0.60 \pm 0.23$ | $\mathbf{0.79 \pm 0.81}$ | $\mathbf{1.20 \pm 0.84}$ | $\mathbf{7.02 \pm 17.78}$ |

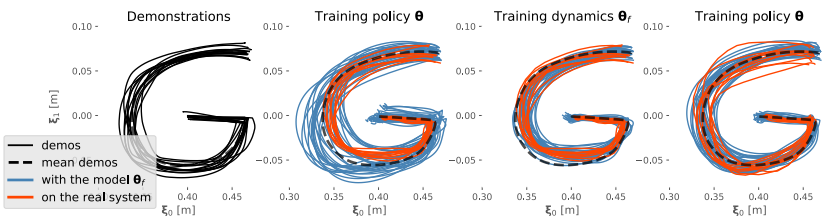
**Table 4.4:** Bhattacharyya distance as quantitative evaluation for the painting task.

of joint angles and velocities of the robot holding the brush. In this configuration, the dynamic model is thus given as a double integrator.

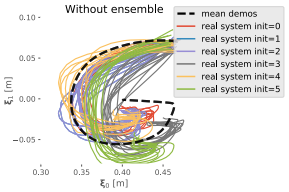
In order to provide adaptation, the policy and discriminator are defined in joint space and two task spaces. The first task space is the position and orientation of the end-effector in a fixed coordinate system and the second is in a coordinate system attached to the box to paint. In each of these three spaces, a Gaussian feedback controller with time-varying gains, feed-forward terms and covariances are defined. More details are given in Section 4.3, see Equation (4.16). The discriminator consists of a factorized Gaussian distribution in each task space as (4.7).

As evaluation, we compare the adaptation capabilities with a ProMP conditioned on the 6-DoF pose of the box. As metric, we compute the Bhattacharyya distance[2] (BD) for the distribution of final position and orientation in the coordinate system of the box between the demonstrations and the reproductions. These final poses are shown in Fig. 4-5a. Results are reported in Table 4.4. They first are performed on the 7 different contexts of the demonstrations. In this case, ProMP conditioning gives better results. Generalization is then tested by sampling 20 contexts from the Gaussian distribution of poses in the demonstrations. To analyze the extrapolation capabilities, the standard deviation is multiplied by $\sigma_c \in \{1, \sqrt{2}, 2\}$. In every case unseen in the demonstrations, the product of policies generalizes better.

[2]This distance is computed by approximating the final distribution with a Gaussian on 10 trajectory samples.




(a) Iterations of Alg. 5 using an ensemble method to learn the dynamics.




(b) First execution without the ensemble method.

**Figure 4-6:** Learning to reproduce the distribution of "G" letter on a 7-DoF Panda robot.

### 4.4.4 Force control and dynamic model learning

In this experiment, we reproduce 2D handwritten letters from [16] in different environments. For each letter, $N = 13$ time-aligned demonstrations of $T = 200$ timesteps are given. With a discretization of time $dt = 0.01$ s, the trajectories last 2 s. The policy learned is then run at 1000 hz on the robot. A third dimension is added to the letters as a fixed height. We alternate between optimizing the policy and refining the model of the system, as proposed in Alg. 5. We consider that the control commands are the forces $\mathbf{f} \in \mathbb{R}^3$ applied at the end-effector as $\boldsymbol{\tau} = \boldsymbol{J}^\top \mathbf{f}$. The state $\boldsymbol{\xi} \in \mathbb{R}^6$ is the position and velocity of the end-effector. In this experimental setup, the configuration of the robot is considered as a hidden variable that influences the dynamics. This uncertainty has to be learned by the identification of the dynamic parameters and the policy robust to the unknown configuration.

A time-dependent feedback controller is used as in the first experiment. The dynamics are learned by an ensemble method. We consider that the system is a mass of 3 kg on which a non-linear state-dependent perturbation is added. This non-linear term is modeled as a MLP with two hidden layers of 20 units, *tanh* activation and the last layer linear. At initialization, the neural networks generate perturbations of a standard deviation of 5 N. This initialization is important: if the true system is in the distribution of systems defined by the initialization of the $L$ models, then the first policy executed on the robot will be already quite good. We demonstrate this by performing the same task with a unique neural network instead of an ensemble.

After initialization of the dynamic parameters, a robust policy is learned for 10 s. This policy is run on the robot for $M = 10$ times, by starting at a random initial state of the demonstrations, and with a random configuration. With the initial guesses about the system, the first computed policy already leads to a very similar distribution, see Fig. 4-6a (second column). The $L$ dynamic parameters are optimized in parallel for 10 s given these new trajectories. The trajectories of the generator now match the trajectories on the true system (third column). The whole process can be repeated until convergence. In this experiment, the policy has been updated only once more for $10s$ and tested on the robot with good results (fourth column).

As a comparison, Fig. 4-6b shows the execution of the first policy on the system when no ensemble method are used. Several sets of trajectories are displayed corre-

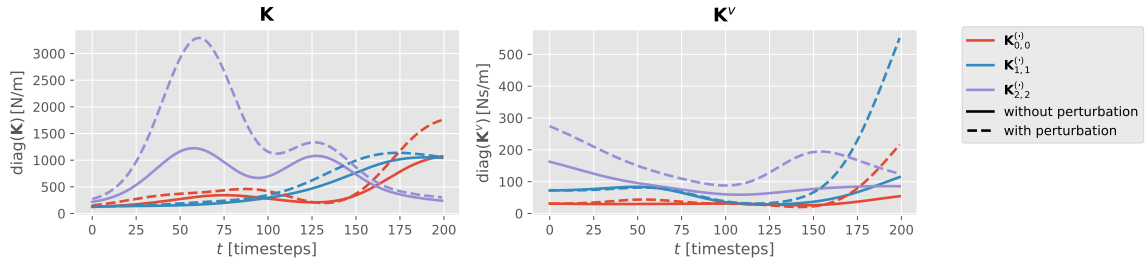



**Figure 4-7:** Increase of the feedback gains resulting from the identification of external perturbations. *Left:* Diagonal values of the time-dependent proportional gains $\boldsymbol{K}(t) \in \mathbb{R}^{3\times3}$. *Right:* Diagonal values of the derivative gains $\boldsymbol{K}^v(t) \in \mathbb{R}^{3\times3}$.

sponding to different initializations of $\boldsymbol{\theta}_f$. In this case, the trajectories are worse than the ensemble method because the epistemic uncertainty is not taken into account, which results in an overconfidence on the dynamic model. As a comparison, the first policy computed using the ensemble method (Fig. 4-6a) has higher feedback gains, resulting in a lower sensitivity to uncertainties.

These first sets of trajectories were produced without any other perturbations than the unknown configuration. To assess for the generality of the method, we tested the process in three different environments. In the first, a user applies short (around 0.2 s) perturbations of around 10 N all along the trajectories and in every direction. After an update of the dynamic model, a higher stochasticity of the environment is inferred. The following update of the policy results in higher feedback terms (see Fig. 4-7).

In a second environment, we tested if the approach is able to learn that an obstacle is on its path, which should be pushed. This time, the letter "U" was chosen and a moving plastic block of around 1 kg put on the table to block the lower part of the letter. Iterations of Alg. 5 are shown for this environment in Fig. 4-8. The first trajectories executed on the robot are truncated (second column). The friction between the end-effector of the robot and the obstacle also prevents motion along $\boldsymbol{\xi}_0$. After updating the dynamics model with $M = 10$ rollouts, the prediction of the generator matches the real system (third column). The following update of the policy leads to a much better reproduction of the distribution (fourth column).

In the third case, the robot needs to draw the letter on a paper. A pen was placed in the gripper of the robot and the end-effector redefined as the tip of the pen. The policy was constrained to apply a constant force of 8 N on the paper, as this information was not in the dataset. The contact of the table was explicitly modeled in the dynamics model as a spring-damper system with high gains. The ensemble method still had to learn the additional friction induced by the tip of the pen on the paper. This dynamics was harder to train and the system needed three iterations of the whole process instead of one in the previous experiments. The trajectories of these iterations are shown in Fig. 4-5b.

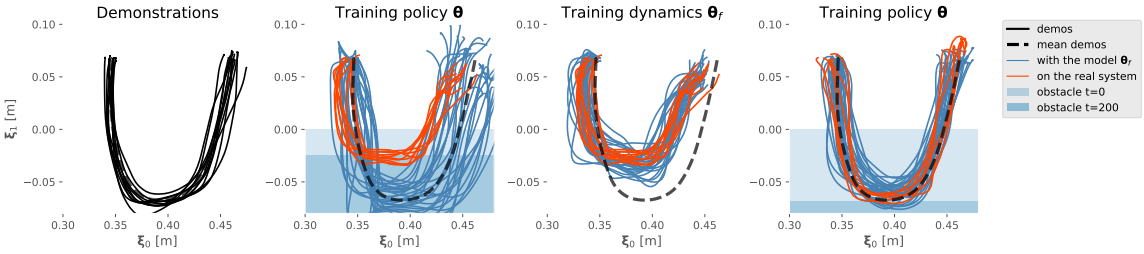



**Figure 4-8:** Iterations of Alg. 5 for reproducing a distribution of "U" shaped trajectories with an obstacle that should be pushed.

## 4.5 Conclusion

The generative adversarial framework is promising for learning movement primitives. It can bring together numerous classical techniques from LfD with the computation power and flexibility of modern machine learning architecture [1]. The approach is easy to be adapted to a wide range of problems. The practitioner, vaguely familiar with machine learning, is only required to define a function for the dynamic system, one for the policy and possibly multiple relevant task spaces. Future work will focus on learning more complex policies and dynamics models, also from raw pixel observations. More efficient model-based policy search methods should also be incorporated in the framework, to cope with longer horizon problems.

# 5

# Summary and recommendations

Inferring intentions from demonstrations is quite a challenge. Having prior knowledge about the system, the environment and the task can significantly increase the efficiency of the process. In robotics however, this knowledge can often be reduced to the use of multiple task spaces in which the configurations or motions are meaningful.

Two approaches to reintroduce mathematical consistency and coherence in models learning distributions in different task spaces have been developed in this thesis. The core idea is to treat the data in their original space - the configuration space for Chapter 3 and the space of trajectories for Chapter 4 - and consider distributions with these proper supports. In Chapter 3, this has been done by using complex approximations of the normalizing constant. This normalizing constant ensures that the multiplication of distributions defined in different task spaces is a proper distribution in the configuration space. When computing normalizing constant of trajectories, the computational complexity becomes of high importance. In Chapter 4, the consistency is ensured through a structured sampling of trajectories. The task spaces are used to define a controller in each of them, and also to compare demonstrations with reproductions.

These two theoretical contributions provide the foundation for many other robotic applications and extensions. In Sec. 3.2, we propose a novel technique that leverages the PoE formulation combined with null space operators to learn task priorities from demonstrations with minimal prior knowledge compared to the state of the art. Our technique is particularly adapted at recovering masked secondary constraints. The objective of manipulability is one of those; it is often of lower importance than task-space objectives. We proposed also an alternative approach to represent tasks with hierarchy, by using uni-Gauss distributions. Thanks to this formulation, each secondary task has a probability to be completely abandoned. Therefore, if the tasks are incompatible, the robot would not any longer try to fulfil secondary tasks by aiming in the direction to minimize their cost, as it is done with the nullspace formu-

lation.

In Sec. 3.4, we proposed an approach to compute torque controllers in different task spaces by using optimal control. By separating the low-level control loop and the updates of the impedance controllers, this technique is seen to be effective with small computation power. This approach is useful to set up virtual guides and assistance in a shared control strategy. We also proposed to use invertible neural networks to learn more complex virtual guides of arbitrary shapes. Another control strategy was suggested by drawing a link between ergodic control and variational inference. This strategy is therefore applicable when the robot needs to generate random motions while satisfying some constraints learned from demonstrations. Alternatively, motion can also be learned and synthesized in the PoE framework by using ProMP experts, which are fully compatible. By mixing ProMP experts with the previous experts specified, our approach offers the opportunity to encode both time-dependent and time-independent objectives.

Additionally, the product of experts allows more flexibility in choosing experts distributions, as a proper normalization is not required any longer. The use of intractable orientation statistics distributions, and the definition of new ones (to encode correlation between positions and orientations, for instance) is then possible.

In Sec. 4.1.2, the approach to stabilize the training of generative adversarial models for motion primitives has been illustrated. It reduces the need for data, and learning dexterous motion with only few examples is then possible. The technique provides also a theoretically grounded way to adapt the movement to objects position and orientation.

In Sec. 4.2, we proposed to treat differently the stochasticity of the environment and the uncertainty coming from our partial knowledge of it. Tested on a real robot, the approach effectively allows the control gains to be adapted according to the external perturbations and the partial knowledge, resulting in very robust and safe controllers. Applied to the learning of time-independent policy as an autonomous dynamic system, this approach provides robust policies. Their stability is highly increased without imposing any further constraints. The approach also scales well to high-dimensional spaces.

In Sec. 4.2, another data-efficient, robust and safe approach was developed, in order to iteratively optimize an imitation policy and refine dynamic models. This technique leverages prior knowledge about the system and the flexibility to learn non-linear perturbation models. The approach was validated on dynamic drawing and pushing tasks, while facing external perturbations produced by a user.

## 5.1 Suggestions for future research

We conclude by providing suggestions for open research directions. The frameworks we developed in this thesis is particularly flexible and is undeniably of help for robotic practitioners to tackle a wide range of problems. By defining relevant task spaces, intelligent policy parameterizations and prior knowledge about the system dynamic, roboticists can focus on modelling the learning problem adequately. The

inference is then performed in a unified manner, without requiring more assumptions than the differentiability of the different components. The inference exploits the power of modern machine learning ecosystems with automatic differentiation [1], similarly as automatic statistic inference libraries as [107].

### 5.1.1 Selection of task spaces

Nevertheless, the selection of relevant task spaces is still an open question. One perspective is that this selection can be framed as a Bayesian inference problem where each task space is associated with a binary random variable indicating its usage. A prior distribution on these variables could indicate to how common they are; for example, the position of the end-effector is more likely to be important than the elbow. However, inference on these discrete variables is hard and computation time increases linearly according to the number of experts. Multiple alternative sources of information other than the samples recorded by manipulating the robot can be exploited; the task spaces are indeed easily interpretable by human users. For example, in an industrial application, an interface could offer the opportunity to the demonstrator to select a set of task spaces represented visually.

For an interface-free application, the robot could exploit user intention hints like gaze or speech interaction. Finally, the task spaces could be selected in an active learning scenario; the robot could propose postures or interactively apply perturbations during the demonstrations to know if a task space is relevant. For instance, if the demonstrator is holding the end-effector, a redundant manipulator could move its elbow to see if the latter would try to prevent its motion. In this case, a higher likelihood would be assigned to the task space related to the elbow. An active learning scenario would need to provide a systematic way to produce these perturbations. The same principle could also be used to learn the hierarchy between the tasks.

### 5.1.2 Active learning

Besides the selection of task spaces, using active learning could be investigated to reduce uncertainties of the policy and the dynamic models. Instead of training a single policy, as proposed in Sec. 4.1, multiple policies could be optimized in parallel as an ensemble method. A measure of divergence between the distribution of control command given by each of these policies provides a significant estimation of the uncertainty. This uncertainty empowers the robot with the ability to request demonstrations either in a particular state or context (external task parameters). This process is assumed to greatly increase the stability of global time-independent policies as discussed in the experiment in Sec. 4.4.2. We already investigated the use of active learning for policies in [40] but using a product of independent policies as proposed in [95].

In Sec. 4.2, we proposed an approach to iteratively optimize the policy and refine the dynamic models. In our experiments, dynamics models are learned robustly by repeating approximately 10 times the same trajectories with variations generated by the stochastic policy. These variations are extremely important to learn robust

dynamic models. As efficient as the system could be, this iterative process is however not completely optimal; the policy is optimized to imitate the demonstrations the best way possible given the actual knowledge about the dynamics. Therefore, it acts as if it would be judged on the next execution, without a possibility to improve. A more optimal process should consider its iterative nature and would produce a policy that does not only try to imitate better but that also collect useful information about the dynamic models.

### 5.1.3 Rhythmic patterns

Many tasks include rhythmic patterns, like music playing, cleaning, cutting or shaking. An open research direction is the encoding of these tasks, which are often the combination of a discrete motion with a rhythmic component. Such movements could be well represented in the framework designed in Chapter 4; the frequencies analysis could be included as an additional task space, looking at the trajectories as a whole. The discriminator could, for example, compare the discrete cosine components of the demonstrations and the reproductions.

The controller should also be adapted to generate rhythmic patterns. For example, time-correlated noise with parametric frequency filters could be used instead of sampling control commands independently.

# Bibliography


[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016. 41, 89, 92

[2] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. *Proc. Intl Conf. on Machine Learning (ICML)*, pages 1–8, 2004. doi: 10.1145/1015330.1015430. 21

[3] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29 (13):1608–1639, 2010. 21

[4] I. Abraham, A. Prabhakar, and T. D. Murphey. Active area coverage from equilibrium. *arXiv preprint arXiv:1902.03320*, 2019. 17

[5] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, volume 1, pages 113–120, 1996. 19

[6] O. Arenz, G. Neumann, and M. Zhong. Efficient gradient-free variational inference using policy search. In *Proc. Intl Conf. on Machine Learning (ICML)*, volume 80, pages 234–243. PMLR, 2018. 26

[7] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. In *Lazy learning*, pages 75–113. Springer, 1997. 19

[8] E. Ayvali, H. Salman, and H. Choset. Ergodic coverage in constrained environments using stochastic trajectory optimization. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 5204–5210, 2017. 17, 53

[9] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 226–234, 2014. 22

[10] D. Berenson, S. Srinivasa, and J. Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011. 44, 45

[11] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59:1371–1394. 20

[12] S. A. Billings. Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. *John Wiley and Sons*, 2013. 78

[13] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, Secaucus, NJ, USA, 2006. 65

[14] C. M. Bishop, N. D. Lawrence, T. Jaakkola, and M. I. Jordan. Approximating posterior distributions in belief networks using mixtures. In *Advances in Neural Information Processing Systems (NIPS)*, pages 416–422, 1998. 26

[15] M. Bohner and N. Wintz. The linear quadratic tracker on time scales. *International Journal of Dynamical Systems and Differential Equations*, 3(4):423–447, 2011. 51, 80

[16] S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016. 17, 18, 19, 21, 28, 29, 30, 36, 42, 47, 71, 75, 76, 80, 81, 82, 87

[17] S. Calinon and A. Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23(15):2059–2076, 2009. 17, 18, 19, 20, 47, 71, 72, 74, 75, 81, 82

[18] S. Calinon, A. Pistillo, and D. G. Caldwell. Encoding the time and space constraints of a task in explicit-duration hidden Markov model. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 3413–3418, San Francisco, CA, USA, September 2011. 17

[19] G. Canal, E. Pignat, G. Alenyà, S. Calinon, and C. Torras. Joining high-level symbolic planning with low-level motion primitives in adaptive hri: application to dressing assistance. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 1–9, 2018. 13, 14

[20] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 1683–1691, 2014. 25, 41

[21] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 465–472, 2011. 21, 73, 76

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. 22

[23] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. In *Proc. Intl Conf. on Learning Representation (ICLR)*, 2017. 45, 65

[24] P. R. Dixon. *Speech Pattern Processing Using Products of Experts*. PhD thesis, University of Birmingham, 2006. 28

[25] M. Djioua and R. Plamondon. Studying the variability of handwriting patterns using the kinematic theory. *Human movement science*, 28(5):588–601, 2009. 17

[26] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian. Probabilistic recurrent state-space models. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 1280–1289, 2018. 73, 78

[27] A. D. Dragan, K. C. Lee, and S. S. Srinivasa. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 301–308, 2013. 17

[28] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017. 19

[29] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987. 41

[30] D. Ehlers, M. Suomalainen, J. Lundell, and V. Kyrki. Imitating human search strategies for assembly. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. 17

[31] M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *IEEE access*, 2:56–77, 2014. 50

[32] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters. Probabilistic model-based imitation learning. *Adaptive Behavior*, 21(5):388–403, 2013. 16, 20, 71, 72, 73, 75

[33] J. Falkoff. Democratization of automation: The next generation of industrial robotics. https://xconomy.com/boston/2018/01/22/democratization-of-automation-the-next-generation-of-industrial-robotics/. Accessed: 2020-08-20. 11

[34] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NeurIPS Workshop on Adversarial Training*, 2016. 72, 82

[35] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 49–58, 2016. 21, 24, 71, 72

[36] O. for Economic Co-operation and Development. *Enhancing SME competitiveness: the OECD Bologna Ministerial Conference*. OECD proceedings. Organisation for Economic Co-operation and Development, 2001. URL https://books.google.ch/books?id=n8m6AAAAIAAJ. 11

[37] B. Fowkes. *The rise and fall of communism in Eastern Europe*. Springer, 1995. 11

[38] S. Füssel. *Gutenberg and the Impact of Printing*. Routledge, 2020. 11

[39] M. Gaebel. *MOOCs: Massive open online courses*. EUA, 2014. 11

[40] H. Girgin, E. Pignat, N. Jaquier, and S. Calinon. Active improvement of control policies with Bayesian Gaussian mixture model. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2020. 92

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014. 22, 65, 72

[42] A. H. Gottlieb. *1,000 years, 1,000 people: Ranking the men and women who shaped the millennium*. Kodansha Amer Inc, 1998. 11

[43] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012. 62

[44] D. B. Grimes, R. Chalodhorn, and R. P. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Robotics: science and systems*, pages 199–206. Cambridge, MA, 2006. 20

[45] D. B. Grimes, D. R. Rashid, and R. P. Rao. Learning nonparametric models for probabilistic imitation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 521–528, 2007. 20

[46] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396, 2017. 12

[47] F. Guo, X. Wang, K. Fan, T. Broderick, and D. B. Dunson. Boosting variational inference. *Advances in Neural Information Processing Systems (NIPS)*, 2016. 26

[48] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375–396, 1999. 27

[49] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1235–1245, 2017. 72

[50] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Trans. on Robotics*, 24(6):1463–1467, 2008. 20

[51] G. E. Hinton. Products of experts. *Proc. Intl Conf. on Artificial Neural Networks. (ICANN)*, 1999. 28, 33, 34, 38, 50

[52] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 22, 28, 33, 38, 57, 71

[53] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 22

[54] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4565–4573, 2016. 72

[55] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013. 75

[56] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 7087–7094, 2018. 19

[57] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1547–1554, 2003. 12

[58] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon. Geometry-aware manipulability learning, tracking and transfer. *International Journal of Robotics Research (IJRR)*, 2020. 43, 44, 49, 60

[59] N. Jetchev and M. Toussaint. Task space retrieval using inverse feedback control. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 449–456, 2011. 22

[60] D. D. Johnson, R. M. Keller, and N. Weintraut. Learning to create jazz melodies using a product of experts. In *International Conference on Computational Creativity*, pages 151–158, 2017. 28

[61] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal. Learning objective functions for manipulation. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 1331–1336, 2013. 22, 43, 71, 72

[62] R. E. Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960. 29

[63] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Trans. on Robotics*, 27(5):943–957, 2011. 19, 20, 72, 81, 85

[64] C. Khatri and K. Mardia. The von Mises-Fisher matrix distribution in orientation statistics. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 95–106, 1977. 47

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proc. Intl Conf. on Learning Representation (ICLR)*, 2015. 26

[66] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *Proc. Intl Conf. on Learning Representation (ICLR)*, 2013. 22, 63, 74

[67] M. Kopicki, S. Zurek, R. Stolkin, T. Moerwald, and J. L. Wyatt. Learning modular and transferable forward models of the motions of push manipulated objects. *Autonomous Robots*, 41(5):1061–1082, 2017. 28

[68] A. Kume, S. Preston, and A. T. Wood. Saddlepoint approximations for the normalizing constant of Fisher–Bingham distributions on products of spheres and stiefel manifolds. *Biometrika*, 100(4):971–984, 2013. 47

[69] M. Laskey, J. Lee, R. Fox, A. D. Dragan, and K. Y. Goldberg. DART: Noise injection for robust imitation learning. In *Conference on Robot Learning (CoRL)*, pages 143–156, 2017. 19

[70] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006. 22

[71] P. Lehner and A. Albu-Schäffer. Repetition sampling for efficiently planning similar constrained manipulation tasks. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 2851–2856, 2017. 19

[72] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon. Memory of motion for warm-starting trajectory optimization. *IEEE Robotics and Automation Letters*, 5(2):2594–2601, 2020. 17

[73] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 475–482, 2012. 24

[74] S. Levine and V. Koltun. Guided policy search. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 1–9, 2013. 17, 24, 71

[75] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proc. Intl Conf. on Informatics in Control, Automation and Robotics. (ICINCO)*, pages 222–229, 2004. 53

[76] H.-C. Lin, M. Howard, and S. Vijayakumar. Learning null space projections. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 2613–2619, 2015. 40

[77] R. Lober, V. Padois, and O. Sigaud. Variance modulated task prioritization in whole-body control. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 3944–3949, 2015. 19

[78] J. Manavalan, Y. Zhao, P. Ray, H.-C. Lin, and M. Howard. A library for constraint consistent learning. *Advanced Robotics*, 34(13):845–857, 2020. 40

[79] G. Mathew and I. Mezić. Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Physica D: Nonlinear Phenomena*, 240(4-5):432–442, 2011. 53

[80] A. C. Miller, N. J. Foti, and R. P. Adams. Variational boosting: Iteratively refining posterior approximations. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 2420–2429, 2017. 26

[81] M. Mühlig, M. Gienger, J. J. Steil, and C. Goerick. Automatic selection of task spaces for imitation learning. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 4996–5002, 2009. 18, 19, 30, 42, 71

[82] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. 17, 20

[83] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987. 39, 40, 50

[84] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008. 31, 80

[85] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. *Proc. Intl Conf. on Machine Learning (ICML)*, pages 663–670, 2000. ISSN 00029645. doi: 10.2460/ajvr.67.2.323. 21

[86] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015. 18, 30, 42, 71

[87] M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009. 26

[88] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018. 20, 75

[89] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish. Recent progress on programming methods for industrial robots. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, 2010. 11

[90] A. Paraschos. *Robot Skill Representation, Learning and Control with Probabilistic Movement Primitives*. PhD thesis, Technische Universität Darmstadt, 2017. 75

[91] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2616–2624, 2013. 20, 47, 48, 71, 72, 75, 76, 80, 81, 82, 86

[92] A. Paraschos, R. Lioutikov, J. Peters, and G. Neumann. Probabilistic prioritization of movement primitives. *IEEE Robotics and Automation Letters*, 2(4): 2294–2301, 2017. 17, 18, 20, 71, 74

[93] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962. 22

[94] E. Pignat and S. Calinon. Learning adaptive dressing assistance from human demonstration. *Robotics and Autonomous Systems*, 93:61–75, 2017. 13

[95] E. Pignat and S. Calinon. Bayesian Gaussian mixture model for robotic policy imitation. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):4452–4458, 2019. doi: 10.1109/LRA.2019.2932610. 74, 76, 81, 92

[96] R. Pini. Invexity and generalized convexity. *Optimization*, 22(4):513–525, 1991. 46

[97] C. Pradalier, F. Colas, and P. Bessiere. Expressing Bayesian fusion as a product of distributions: application in robotics. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 1851–1856, 2003. 28

[98] G. Raiola, X. Lamy, and F. Stulp. Co-manipulation with multiple probabilistic virtual guides. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 7–13, 2015. 45

[99] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.049. 20

[100] R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014. 26

[101] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018. 17, 74

[102] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 1530–1538, 2015. 26

[103] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. Intl Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011. 19, 83, 85

[104] L. Rozo, N. Jaquier, S. Calinon, and D. G. Caldwell. Learning manipulability ellipsoids for task compatibility in robot manipulation. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 3183–3189, 2017. 18

[105] R. Salakhutdinov and G. Hinton. Deep Boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009. 22

[106] T. Salimans, D. A. Knowles, et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013. 26

[107] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016. 92

[108] K. Sangsuvan. Small business in the WTO: Small fish in a big pond or globalization 3.0. *Mich. St. Int'l L. Rev.*, 23:341, 2014. 11

[109] A. Savini and G. Savini. A short history of 3d printing, a technological revolution just started. In *2015 ICOHTEC/IEEE international history of high-technologies and their socio-cultural contexts conference (HISTELCON)*, pages 1–8, 2015. 11

[110] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Intl Journal of Robotic Research*, pages 561–572. Springer, 2005. 20

[111] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell. Learning task priorities from demonstrations. *IEEE Transactions on Robotics*, 35(1):78–94, 2018. 18

[112] C. Sminchisescu, M. Welling, and G. Hinton. A mode-hopping MCMC sampler. Technical report, University of Toronto, 2003. 25

[113] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3483–3491, 2015. 19

[114] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998. 12

[115] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel. Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 2074–2081, 2010. 21

[116] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proc. Intl Conf. on Machine Learning (ICML)*, volume 1, pages 288–293, 2000. 19

[117] C. Voss, M. Moll, and L. E. Kavraki. Atlas+x: Sampling-based planners on constraint manifolds. Technical report, RICE computer science, 2017. 44

[118] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2): 1–305, 2008. 25, 26

[119] M. Welling, A. Mnih, and G. E. Hinton. Wormholes improve contrastive divergence. In *Advances in Neural Information Processing Systems (NIPS)*, pages 417–424, 2004. 38

[120] O. Wilde. *The soul of man under socialism*. Castrovilli Giuseppe, 1910. 11

[121] C. Williams, F. V. Agakov, and S. N. Felderhof. Products of Gaussians. In *Advances in Neural Information Processing Systems (NIPS)*, number 1, pages 1017–1024, 2002. 30

[122] Y. Yang, V. Ivan, and S. Vijayakumar. Real-time motion adaptation using relative distance space representation. In *Proc. Intl Conf. on Advanced Robotics (ICAR)*, pages 21–27, 2015. 44

[123] T. Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985. 43

[124] M. J. Zeestraten, I. Havoutis, J. Silverio, S. Calinon, and D. G. Caldwell. An approach for imitation learning on Riemannian manifolds. *IEEE Robotics and Automation Letters*, 2(3):1240–1247, 2017. 18, 49, 69

[125] H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech & Language*, 21(1):153–173, 2007. 18, 82

[126] H. Zen, M. J. Gales, Y. Nankaku, and K. Tokuda. Product of experts for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):794–805, 2012. 18, 28, 71, 75

[127] Y. Zhang, K. Hauser, and J. Luo. Unbiased, scalable sampling of closed kinematic chains. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 2459–2464, 2013. 44

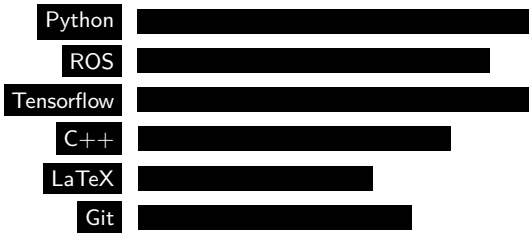
[128] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 21, 23, 71

# EMMANUEL PIGNAT

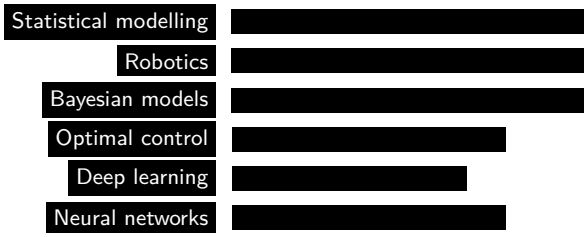
PhD Student

Ch. des Rottes 7
+41 78 721 85 30
emmanuel.pignat@gmail.com
1893 Muraz, Switzerland
Emmanuel Pignat

## IT SKILLS




## TECHNICAL SKILLS




## EXPERIENCE

2016 – 2020 **Research assistant** **Idiap Research Institute** Development of imitation learning algorithm for robotics.
`Statistical modelling / Robotics / Neural networks`

2015 – 2015 **Research intern** **GTX medical** Developments of algorithms to detect gait events of pathological users.
`Python / C++ / Vicon / Wearable sensors`

2012 – 2015 part time **Independent scientific 3d animator** Realization of various 3d animation for scientific vulgarization.
`Blender / After-effect / Illustrator`

## EDUCATION

2016 – 2020 **Doctor of Philosophy - PhD** **Ecole polytechnique fédérale de Lausanne**

2014 – 2016 **Master's degree in Micro-engineering** **Ecole polytechnique fédérale de Lausanne**

2011 – 2014 **Bachelor's degree in Micro-engineering** **Ecole polytechnique fédérale de Lausanne**

2006 – 2011 **Maturité fédérale gymnasiale** **Collège Saint Maurice**

## SELECTED PUBLICATIONS

2020 Pignat, E., Girgin, H. and Calinon, S.. **Generative Adversarial Training of Product of Policies for Robust and Adaptive Movement Primitives**. In Proc. Conference on Robot Learning (CoRL).

2019 Pignat, E. and Calinon, S.. **Bayesian Gaussian Mixture Model for Robotic Policy Imitation.** IEEE Robotics and Automation Letters (RA-L), 4:4, 4452-4458.

2018 Canal, G., Pignat, E., Alenya, G., Calinon, S. and Torras, C.. **Joining high-level symbolic planning with low-level motion primitives in adaptive HRI: application to dressing assistance.** In Proc. of the IEEE Intl Conf. on Robotics and Automation (ICRA), pp. 3273-3278.

2017 Pignat, E. and Calinon, S.. **Learning adaptive dressing assistance from human demonstration.** Robotics and Autonomous Systems, 93, 61-75.

## LANGUAGES

**French** - native
**English** - proficient
**German** - rudimentary

## HOBBIES

I love biking, climbing, mountaineering, skating, playing the piano, drawing, painting, filming, photographing, reading, ... maybe just learning new skills actually...