Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

Learning Task-Parameterized Riemannian Motion Policies

An T. Le

Course of Study:	Information Technology
Examiner:	Ph. D. Jim Mainprice
Supervisor:	Dr. Meng Guo
Commenced:	April 19, 2021
Completed:	September 9, 2021

Abstract

Nowadays, robots gradually have more autonomy to operate alongside people not only on assembly lines, but also in daily spaces such as kitchens, museums, or hospitals. In these scenarios, a robot must demonstrate a high degree of adaptability in real-time dynamic situations while satisfying task compliance objectives such as collision avoidance. The robot skill also needs to be programmed with ease to cope with an enormous variety of task behaviors. To achieve this, we propose Task-parameterized Riemannian Motion Policy (TP-RMP) framework to address the challenges associated with learning and reproducing the skills under multiple task objectives and situations. Specifically, the task objectives are viewed as multiple subtasks, learned as stable policies from demonstrations. The learned policies are also task conditioned and able to cope with real-time changing task situations. Under the RMPflow framework, our approach synthesizes a stable global policy in the configuration space that combines the behaviors of these learned subtasks. The resulting global policy is a weighted combination of the learned policies satisfying the robot's kinematic and environmental constraints. Finally, we demonstrate the benchmarks of TP-RMP under increasing task difficulties in terms of external disturbances and skill extrapolation outside of the demonstration region.

Contents

List of Figures

List of Algorithms

List of Abbreviations

BIC Bayesian Information Criterion. 48

DMP Dynamic Movement Primitives. 17

EM Expectation-Maximization. 7

GDS Geometric Dynamical System. 22

GMM Gaussian Mixture Model. 7

LfD Learning from Demonstrations. 7

LP Linear Program. 37

LQC Linear Quadratic Control. 17

MSE Mean squared error. 8

PbD Programming by Demonstrations. 15

QP Quadratic Program. 34

RMP Riemannian Motion Policy. 7

SMS Simple Mechanical System. 20

TP-DMP Task-parameterized Dynamic Movement Primitives. 17

TP-GMM Task-parameterized Gaussian Mixture Model. 7

TP-RMP Task-parameterized Riemannian Motion Policy. 3

UMIC Unified Motion and variable Impedance Control. 18

1 Introduction

Nowadays, there is an increasing interest in programming robot behaviors that are not constrained in static and controlled environment settings. Due to the dynamic characteristics of a practical workspace, the robot behaviors should be reusable in different real-time task situations and robust to external disturbances. Moreover, the robot must also exhibit compliance and collision avoidance behaviors for safe execution. Indeed, as an example of a handover task working with humans, handover action cannot be constrained at a specific point for natural interaction. The robot arm at the handover phase should not be stiff and may be further subjected to disturbance from the human. Another example is the pouring task, which requires satisfying multiple objectives to perform it properly. The robot should reach the targets (i.e., cup mouths) with the kettle, avoid pouring water on the floor while approaching, and avoid colliding with other objects in the scene.

To address the above motivation, LfD, which is also referred to as Programming by Demonstrations (PbD) [BCDS08], provides a paradigm for robot skill learning required for executing complex trajectories. Manually programming such skills can otherwise be challenging or infeasible. On the other hand, the tasks may need to satisfy multiple objectives as the mentioned pouring task. RMP [RIK+18] and RMPflow [CMI+19] provides a framework to synthesize control policy satisfying multiple objectives by a divide and conquer manner, rendering a combined desired behavior when following the generated policy.

In this thesis, we aim to design a skill learning model, namely TP-RMP, that utilizes these paradigms. The model will exhibit three properties:

- **Reactive**: is a time-invariant property. It enables the skill model to be robust under timing drifts or external disturbances during the execution phase.
- **Adaptation**: adapts the skill behaviors to new task situations in real-time, which are different from the demonstration phase.
- **Composable**: combines multiple desired objectives as policies associated with different subtask spaces while ensuring that the combined overall policy is Lyapunov stable.

To be concrete, TP-RMP, which encodes complex demonstrations, is an adapting RMP dependent to real-time task situations, which is composable other RMPs serving for other task objectives in the RMPflow framework.

Structure

The thesis consists of the following chapters:

Chapter 2 – Related works: presents the related works to the thesis topic.

Chapter 3 – Preliminary: describes the preliminary foundation works that the thesis is based upon.

Chapter 4 – Task-parameterized Riemannian Motion Policies: formulates the main model of the thesis.

Chapter 5 – Experimentation: presents the TP-RMP benchmark results.

Chapter 6 – Conclusion and Outlook: gives conclusions and future outlooks.

2 Related works

This thesis is at the intersection of multiple research directions, such as motion primitives, impedance control, composable motion policies, and robot learning. Here, we review the most relevant works to this thesis, addressing the mentioned research directions.

Movement primitives are essentially viewed as the basis for more complex robot tasks by sequencing or superimposing them in parallel. Each movement primitive encodes a "primitive" behavior such as point-to-point movement (e.g., reaching for a cup, swinging a golf club, etc.) with a set of differential equations [BCDS08; INS02]. This line of research is often referred to as Dynamical System-based approaches since they directly define a robot motion with a differential equation. A notable work is the Dynamic Movement Primitives (DMP) [MKKP12; UGAM10] which learns new motor tasks from physical interactions. DMP provides real-time motion primitive adaptation to new target states in dynamic environments. However, DMP still suffers from the extrapolation problem where it cannot generalize outside of the demonstration region. Task-parameterized Dynamic Movement Primitives (TP-DMP) [PL18] improves extrapolation problem of DMP by formulating learning as a density estimation problem. This approach utilizes GMM to model the forcing term of DMP from data, which is closely related to our work.

Another line of work providing real-time adaptation is to combine variable impedance control with density estimation problems. [CSC10] proposes an imitation learning approach to model robot skills with dynamical systems and formulate the stiffness matrix to be inversely proportional to the observed covariance in the demonstrations. Later, TP-GMM [Cal16] treats frames of reference as task situations (i.e. *task parameters*) providing a strong extrapolation capability out of demonstration region. Recently, a body of works [LGD+21; RGK+20], which based on TP-GMM modeling, are successful in sequencing skill models and reproducing multi-modal forceful demonstration based on real-time task conditions. However, these approaches suffer from high computation costs in skill retrieval since they rely on Linear Quadratic Control (LQC) to track a full trajectory following the TP-GMM components.

On the view of composable motion policies, RMP [RIK+18] is a mathematical object to modular robotic behaviors. It allows constructing primitive behavioral components separately in different subtask spaces and then combining them using the RMP-*algebra* formulated in the RMPflow algorithm [CMI+19]. These approaches avoid conflicting behaviors when summing policy contributions together by treating subtask space geometry metrics as weightings in the sum. An effort of introducing LfD to RMPflow framework is presented in [RLR+20], where an RMP is learned from demonstration data in form of learning parametric potential field. However, there is no clear adaptability in their method since they rely on manually-designed RMP for goal conditioning. Another notable mention is that, recently, [ULL+21] proposes Composable Energy Policies framework, which combines motion policies by formulating an optimization problem over the product of these stochastic policies.

Perhaps this work is mostly related to [RLR+20], where we also attempt to learn a reactive motion policy as a RMP in the RMPflow framework to benefit from its Lyapunov stability property when combining with other objectives RMP. However, we design TP-RMP to have adaptability under different task situations while maintaining its composability in the RMPflow framework. Our work is inspired from Unified Motion and variable Impedance Control (UMIC) [KK17] method, in which we explicitly learn a task-parameterized potential field to guide the system and a task-parameterized dissipative field to damp the system optimally under real-time changing task situations. To realize that, similar to TP-DMP, we model the RMP's dynamical system and subtask space metric with TP-GMM to enable skill extrapolation out of demonstration region.

3 Preliminary

The preliminary is presented in this section, which are the theoretical foundations that this thesis works are built upon.

3.1 Robot state space representations

The theories in this thesis are designed upon the robot arm state-space representations on both *configuration space* and *task space*.

We consider a robot arm with its configuration space $\mathcal{C}$ represented by a smooth Euclidean manifold, admitting a global generalized coordinate $\boldsymbol{q} \in \mathbb{R}^d$. It is usually more easy and intuitive to describe the behaviors (e.g., trajectory tracking, collision avoidance) of the robot arm end-effector motion on another manifold representing the task space, denoted as $\mathcal{T}$. The task space of the end-effector could be the concatenation of 3D positional space and 3D orientational unit quaternion space. However, flat manifolds like Euclidean manifolds may be inadequate to represent such space, as they rely on rough approximations to account for the constraints imposed by the unit quaternion representation. These approximations may lead to inaccurate skill models or unstable controllers. Instead, we could endow the task space with a Riemannian manifold $\mathcal{M} = \mathbb{R}^3 \times \mathbb{S}^3$ [Zee18]. Briefly, for each point $\boldsymbol{x} \in \mathcal{M}$, there exists a tangent space $\boldsymbol{T}_{\boldsymbol{x}}\mathcal{M}$ as a vector space $\mathbb{R}^m$ allowing local Euclidean operations, while being geometrically consistent with manifold constraints. Notably, there are exponential and logarithmic maps that project points between $\boldsymbol{T}_{\boldsymbol{x}}\mathcal{M}$ and $\mathcal{M}$. Indeed, we can exploit these maps to correctly compute statistics over $\mathcal{M}$ using Riemannian normal distributions [Zee18] that encode full end-effector motion patterns.

Naturally, there exists a smooth task map $\psi : \mathcal{C} \to \mathcal{T}$ that maps the configuration space to the task space as forward kinematics, of which there also exists the task map Jacobian $\boldsymbol{J}(\boldsymbol{q})$ and its time derivative $\dot{\boldsymbol{J}}(\boldsymbol{q})$ for the RMP-*algebra* operations introduced in Section 3.3.

3.2 Riemannian Motion Policies

RMP [RIK+18] describes dynamic motions on Riemannian manifolds. Considering a smooth Riemannian manifold $\mathcal{M}$, the natural form of the RMP on the manifold $\mathcal{M}$ is a mathematical object $\mathcal{R} = (\boldsymbol{f}, \boldsymbol{M})^{\mathcal{M}}$, where the dynamical system $\boldsymbol{f} : \boldsymbol{x}, \dot{\boldsymbol{x}} \to \ddot{\boldsymbol{x}}$ is described by the second-order differential equation that maps the pose $\boldsymbol{x} \in \mathcal{M}$ and velocity $\dot{\boldsymbol{x}} \in \boldsymbol{T}_{\boldsymbol{x}}\mathcal{M}$ to the desired acceleration $\ddot{\boldsymbol{x}}_d = \boldsymbol{f}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ on the manifold. The second component $\boldsymbol{M} \in \mathbb{R}_+^{m \times m}$ is the endowed Riemannian metric of $\mathcal{M}$, which is a positive definite matrix defining the geometric structure of the underlying manifold. Note that m is the dimension of the tangent space $\boldsymbol{T}_{\boldsymbol{x}}\mathcal{M}$.

In the Geometric Mechanic [BL04] view, RMP can be represented as the canonical form $(\boldsymbol{a}, \boldsymbol{M})^{\mathcal{M}}$. An realization of this canonical form is the Simple Mechanical System (SMS) [BL04]:

$$\boldsymbol{a} = \boldsymbol{M}(\boldsymbol{x})^{-1}(-\nabla\Phi(\boldsymbol{x}) - \Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \boldsymbol{\xi}_{\boldsymbol{M}}(\boldsymbol{x}, \dot{\boldsymbol{x}})) \tag{3.1}$$

where $\boldsymbol{M}(\boldsymbol{x}) : \mathcal{M} \to \mathbb{R}_+^{m \times m}$ is view as *inertia matrix*, $\Phi(\boldsymbol{x}) : \mathcal{M} \to \mathbb{R}_+$ is the potential field, and $\Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) : \mathcal{M}, \boldsymbol{T}_{\boldsymbol{x}}\mathcal{M} \to \boldsymbol{T}_{\boldsymbol{x}}\mathcal{M}$ is a non-conservative generalized force derived from a dissipative field (i.e. damping force). The Riemannian metric $\boldsymbol{M}$ induces the curvature term $\boldsymbol{\xi}_{\boldsymbol{M}}$, which bends the trajectories to follow geodesics on the manifold $\mathcal{M}$ in the absence of the potential and damping forces. An important connection is that the curvature term $\boldsymbol{\xi}_{\boldsymbol{M}}$ is equivalent to Coriolis force proven using the Christoffel symbol in curvilinear spaces [CMI+19]. In this case, for the natural form $(\boldsymbol{f}, \boldsymbol{M})^{\mathcal{M}}$, the dynamical system is $\boldsymbol{f} = -\nabla\Phi(\boldsymbol{x}) - \Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \boldsymbol{\xi}_{\boldsymbol{M}}(\boldsymbol{x}, \dot{\boldsymbol{x}})$.

In general, the acceleration policy (3.1) dictates the motion on a manifold under the influence of a damped virtual potential field, which grounds the system model that this thesis develops in the next chapter.

3.3 RMPflow

This section provides a brief introduction of RMPflow [CMI+19], the computational framework for policy generation with RMPs. The idea of RMP formulation assumes that the overall complex task can be decomposed into a set of subtasks defined on different *subtask spaces*. The subtasks could be goal-reaching or trajectory tracking for the end-effector, collision avoidance for the robot links, etc. Thus, the task space $\mathcal{T}$ can be represented as the union of multiple subtask spaces $\mathcal{T} = \bigcup_{n=1}^{L} \mathcal{T}_{l_n}$. In general, RMPflow utilizes the dynamical systems and the metrics of RMPs to generate an acceleration policy on the configuration space $\mathcal{C}$ such that the transformed policies exhibit the combined behaviors in each of the subtask spaces.

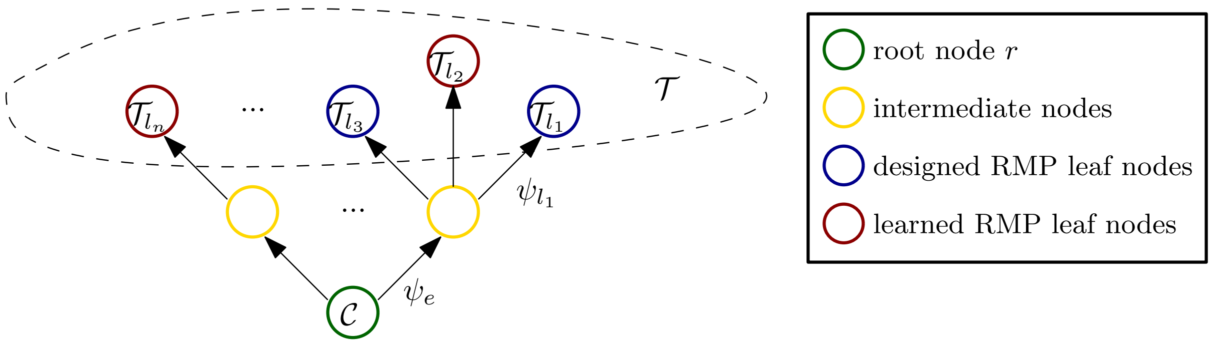


Figure 3.1: An example diagram of RMP-tree.

RMPflow is designed as the RMP-tree, a directed tree encoding the structure of the task map. The specifications of the RMP-tree (V, E) are as follows:

- Each node $v \in V$ associates with a state $(\boldsymbol{x}_v, \dot{\boldsymbol{x}}_v)$ on a manifold $\mathcal{M}_v$ along with its canonical RMP $(\boldsymbol{a}_v, \boldsymbol{M}_v)^{\mathcal{M}_v}$.
- Each edge $e = (u, v) \in E$ corresponds to a smooth map $\boldsymbol{\psi}_e : \mathcal{M}_u \to \mathcal{M}_v$ from the given parent node u manifold to the child node v manifold.
- The root node r associates with the state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ in the configuration space $\mathcal{C}$ and its $(\boldsymbol{a}, \boldsymbol{M})^{\mathcal{C}}$.
- The leaf nodes $\{l_n\}_{n=1}^{L}$ associate with the subtask RMPs $\{(\boldsymbol{a}_{l_n}, \boldsymbol{M}_{l_n})^{\mathcal{M}_{l_n}}\}_{n=1}^{L}$. Each subtask RMP encodes the behavior as acceleration policy $\boldsymbol{a}_{l_n}$, while its Riemannian metric contributes a state-dependent importance weight to the policy when combined with other policies.

An example RMP-tree is shown in Figure 3.1. Note that, in this work, some subtask leaf nodes are associated with manually-designed RMPs, while other leaf nodes are subjected to the learned RMPs presented in the next chapter.

The subtask policies defined on the leaf nodes are combined using RMP-*algebra*. Consider a node u in the RMP-tree with N child nodes $\{v_j\}_{j=1}^{N}$ and their edge $\{e_j\}_{j=1}^{N}$ between a parent node u and its child nodes, RMP-*algebra* is formulated with three operators:

- **pushforward**: propagates the state of a node in the RMP-tree to update the states of its child nodes. Let $(\boldsymbol{x}_u, \dot{\boldsymbol{x}}_u)$ and $\{(\boldsymbol{x}_{v_j}, \dot{\boldsymbol{x}}_{v_j})\}_{j=1}^{N}$ be the state associated with the parent node and the child nodes, respectively. The state of its j-th child node v_j is computed as $(\boldsymbol{x}_{v_j}, \dot{\boldsymbol{x}}_{v_j}) = (\boldsymbol{\psi}_{e_j}(\boldsymbol{x}_u), \boldsymbol{J}_{e_j}(\boldsymbol{x}_u)\dot{\boldsymbol{x}}_u)$, where $\boldsymbol{J}_{e_j} = \frac{\partial \boldsymbol{\psi}_{e_j}}{\partial \boldsymbol{x}_u}$ is the Jacobian matrix.

- **pullback**: propagates the natural form of RMPs $\{(\boldsymbol{f}_{v_j}, \boldsymbol{M}_{v_j})^{\mathcal{M}_{v_j}}\}_{j=1}^N$ from the child nodes to the parent node. The natural form RMP associated with the parent node $(\boldsymbol{f}_u, \boldsymbol{M}_u)^{\mathcal{M}_u})$ is computed as:

$$\boldsymbol{f}_u = \sum_{j=1}^{N} \boldsymbol{J}_{e_j}^{\mathsf{T}}(\boldsymbol{f}_{v_j} - \boldsymbol{M}_{v_j}\dot{\boldsymbol{J}}_{e_j}\dot{\boldsymbol{x}}_u), \quad \boldsymbol{M}_u = \sum_{j=1}^{N} \boldsymbol{J}_{e_j}^{\mathsf{T}}\boldsymbol{M}_{v_j}\boldsymbol{J}_{e_j} \tag{3.2}$$

- **resolve**: maps an RMP from its natural form $(\boldsymbol{f}_u, \boldsymbol{M}_u)^{\mathcal{M}_u}$ to its canonical form $(\boldsymbol{a}_u, \boldsymbol{M}_u)^{\mathcal{M}_u}$ with $\boldsymbol{a}_u = \boldsymbol{M}_u^{\dagger}\boldsymbol{f}_u$, where † denotes Moore-Penrose inverse.

Recursive applications of **pushforward** operations in the forward pass followed by **pullback** operations in a backward pass along the RMP-tree compute a weighted combination of the leaf node RMPs as the natural form RMP $(\boldsymbol{f}_r, \boldsymbol{M}_r)^{\mathcal{C}}$ at the root r. Finally, a **resolve** operation generates a global configuration space policy $\boldsymbol{a}_r = \pi(\boldsymbol{q}, \dot{\boldsymbol{q}})$.

RMPflow preserves the convergence property of leaf node policies: if all subtask RMPs are in the form of (3.1) which is a simplified version of Geometric Dynamical System (GDS), the combined policy in the configuration space is also in the form of (3.1) and thus Lyapunov stable. In the following chapter, we rely on the stability property of RMPflow to combine the learned TP-RMPs with other manually-designed RMPs into a stable global policy. For further detailed theoretical analysis, we refer the readers to [CMI+19].

3.4 Task-parameterized Gaussian Mixture Model

This section introduces the TP-GMM, which is a density estimation model for LfD. It provides an elegant probabilistic representation of motion skills in the task space. First, we define the term *model parameters* refers to the learned parameters of a model describing the movement or skill. In contrast, the external parameters representing the task situation (such as poses of the objects of interest or the actors) will be denoted *task parameters*. The latter are essentially frames of reference used as inputs to transform the learned model parameters according to the current situation.

From some demonstrations, TP-GMM can be used to learn the skill spatial distribution of the demonstration space. Assuming given N demonstrations in task space $\mathcal{T}$, each demonstration i contains T_i observations $\boldsymbol{Z}_i = \{\boldsymbol{\zeta}_{i,t}\}_{t=1}^{T_i}, \boldsymbol{\zeta} \in \mathcal{T}$. The same demonstrations are projected to the perspective of P different coordinate systems (e.g. objects of interest frame, end-effector frame). The projected demonstrations are transformed from the static global frame to frame p by:

$$\boldsymbol{\zeta}^{(p)} = \boldsymbol{A}^{(p)-1}(\boldsymbol{\zeta} - \boldsymbol{b}^{(p)}) \tag{3.3}$$

where $\mathcal{F} = \{\boldsymbol{A}^{(p)}, \boldsymbol{b}^{(p)}\}_{p=1}^{P}$ is the set of the rotation and translation of frame p with regard to the global frame.

Task parameterization of GMMs incorporates observations from the perspective of different frames of reference, thus allowing the skill to adapt its motion to new frames representing task situations. Hence, the TP-GMM is described by the model parameters $\Theta = \{\pi_k, \{(\boldsymbol{\mu}_k^{(p)}, \boldsymbol{\Sigma}_k^{(p)})\}_{p=1}^{P}\}_{k=1}^{K}$ where K represents the number of Gaussian components in the mixture model, π_k is the mixing coefficient (i.e. the prior probability) of each component, and $\{(\boldsymbol{\mu}_k^{(p)}, \boldsymbol{\Sigma}_k^{(p)})\}_{p=1}^{P}$ are the parameters of the k-th Gaussian component within frame p. Note that the mixture model above can not be learned independently for each frame, as the mixing coefficients π_k are shared by all frames and the k-th component in frame p must map to the corresponding k-th component in the global frame.

EM [Cal16] is a well-established algorithm to learn TP-GMM. Learning of the model parameters is achieved by maximizing the log-likelihood, under the constraint that, the data in the different reference frames are generated from the same source, resulting in an EM process with the E-step:

$$\gamma_{t,k} = \frac{\pi_k \prod_{p=1}^{P} \mathcal{N}(\boldsymbol{\zeta}_t^{(p)} | \boldsymbol{\mu}_k^{(p)}, \boldsymbol{\Sigma}_k^{(p)})}{\sum_{j=1}^{K} \pi_j \prod_{p=1}^{P} \mathcal{N}(\boldsymbol{\zeta}_t^{(p)} | \boldsymbol{\mu}_j^{(p)}, \boldsymbol{\Sigma}_j^{(p)})} \tag{3.4}$$

and then the M-step:

$$\begin{aligned} \pi_k &= \frac{\sum_{t=1}^{T} \gamma_{t,k}}{T} \\ \boldsymbol{\mu}_k^{(p)} &= \frac{\sum_{t=1}^{T} \gamma_{t,k} \boldsymbol{\zeta}_t^{(p)}}{\sum_{t=1}^{T} \gamma_{t,k}} \\ \boldsymbol{\Sigma}_k^{(p)} &= \frac{\sum_{t=1}^{T} \gamma_{t,k} (\boldsymbol{\zeta}_t^{(p)} - \boldsymbol{\mu}_k^{(p)})(\boldsymbol{\zeta}_t^{(p)} - \boldsymbol{\mu}_k^{(p)})^{\mathsf{T}}}{\sum_{t=1}^{T} \gamma_{t,k}} \end{aligned} \tag{3.5}$$

that iteratively update the model parameters until convergence. The model parameters can be initialized with a k-means procedure or dividing the demonstration equally through time for each component to speed up the EM process. Note that in a standard GMM, EM estimates the Gaussian parameters $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Here, EM is used to estimate task-parameterized model parameters $\{(\boldsymbol{\mu}_k^{(p)}, \boldsymbol{\Sigma}_k^{(p)})\}_{p=1}^{P}$ by incrementally modeling the local importance of the reference frames (see Figure 3.2 for illustration).

After learning, the learned model can be used to reproduce movements in new situations (new positions and orientations of reference frames) during reproduction. Given the current observed frames $\mathcal{F} = \{\boldsymbol{A}^{(p)}, \boldsymbol{b}^{(p)}\}_{p=1}^{P}$, a new GMM $\hat{\Theta}$ is computed in global

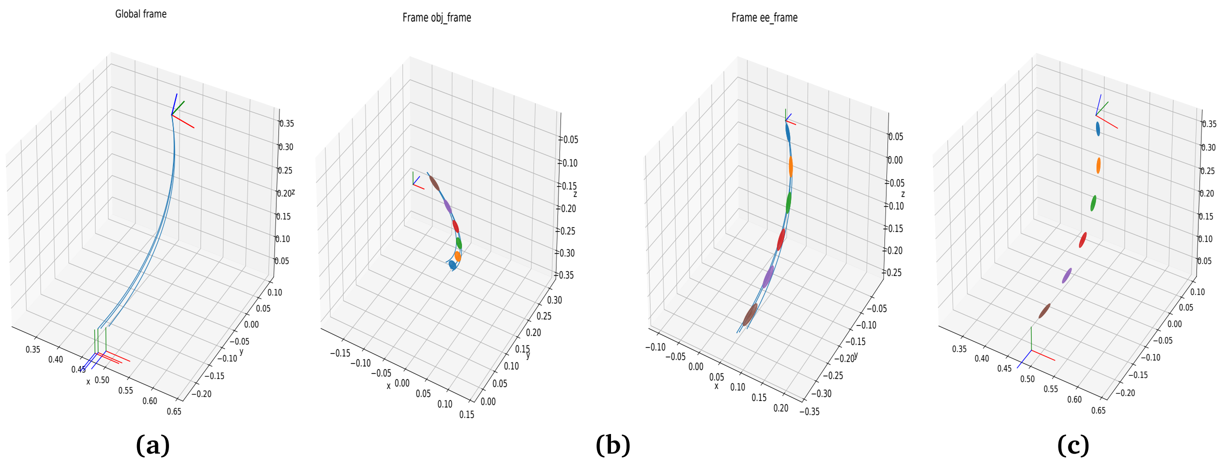


Figure 3.2: Learning TP-GMM for picking skill in 3D Euclidean manifold. The ellipsoids represent the standard deviation region of the Gaussian variances. The small and narrow ellipsoids reflect the high local invariance of the trajectories observed from the different frames. The more round ellipsoids represent regions where precision is not required. Units are in meters. **(a)** Provided 3 demonstrations with 3 associated frame sets. **(b)** The demonstrations are transformed in local frame perspectives, e.g. the object frame and end-effector frame, then the EM process learns the local GMMs in local frame perspectives. **(c)** Given new task parameter frames $\mathcal{F}$, a new GMM $\hat{\Theta}$ is computed accordingly.

frame with the parameters $\{\pi_k, (\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k)\}_{k=1}^K$ by a product of the transformed Gaussian components across different frames in the global frame:

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_k &= (\sum_{p=1}^{P} \hat{\boldsymbol{\Sigma}}_k^{(p)-1})^{-1} \\ \hat{\boldsymbol{\mu}}_k &= \hat{\boldsymbol{\Sigma}}_k (\sum_{p=1}^{P} \hat{\boldsymbol{\Sigma}}_k^{(p)-1} \hat{\boldsymbol{\mu}}_k^{(p)}) \end{aligned} \tag{3.6}$$

where the parameter of the k-th Gaussian is transform into the global frame as $\hat{\boldsymbol{\mu}}_k^{(p)} = \boldsymbol{A}^{(p)} \boldsymbol{\mu}_k^{(p)} + \boldsymbol{b}^{(p)}$ and $\hat{\boldsymbol{\Sigma}}_k^{(p)} = \boldsymbol{A}^{(p)} \boldsymbol{\Sigma}_k^{(p)} \boldsymbol{A}^{(p)\intercal}$. The task parameters could be varied with time, but we drop the time index t for the ease of notation.

4 Task-parameterized Riemannian Motion Policies

We present the TP-RMP model design as concrete mathematical formulations in this section.

4.1 Problem statement

This chapter is dedicate to formulate and learn a skill with the *reactive*, *adaptive* and *composable* properties as presented in Chapter 1. To the best of our knowledge, no existing work learns stable reactive policies from demonstrations capable of reproducing the desired combined behaviors in different task conditions.

In this light of motivation, we consider a subtask space $\mathcal{T}_{l_n}$ corresponding to leaf node l_n in the RMP-tree, we assume the availability of N demonstrations $\{\boldsymbol{Z}_i\}_{i=1}^N$. Here the i-th demonstration has T_i data points $\boldsymbol{Z}_i = \{\boldsymbol{\zeta}_{i,t}\}_{t=1}^{T_i}$, where each state $\boldsymbol{\zeta} \in \mathcal{T}_{l_n}$ in the demonstration may have generalized coordinate $\mathbb{R}^d$. For each demonstration i, there are also P coordinate frames as task parameters $\mathcal{F}_i = \{\boldsymbol{A}_i^{(p)}, \boldsymbol{b}_i^{(p)}\}_{p=1}^P$ at the time recording the demonstration. Note that while we assume only the data points as system poses, the demonstrated system velocities and accelerations can be easily approximated using linear approximation.

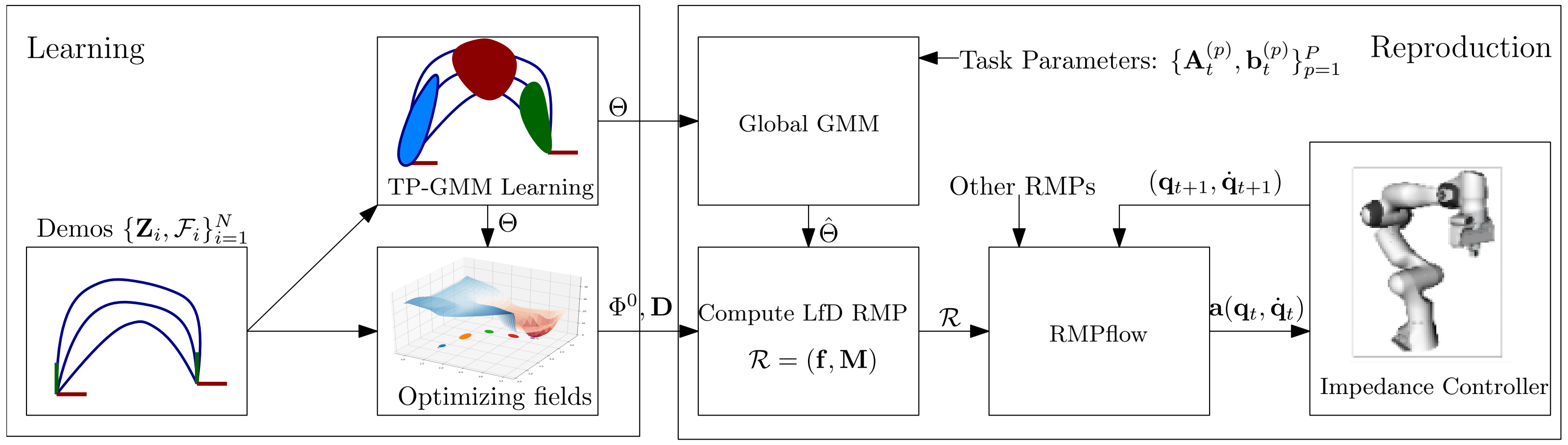
Our goal is to learn a TP-RMP $\mathcal{R} = (\boldsymbol{f}, \boldsymbol{M})^{\mathcal{M}}$ for gravity-compensated systems, where $\boldsymbol{M}$ is the Riemannian metric that will be learned to describe the smooth manifold $\mathcal{M}$ embedded in the ambient manifold of the task space $\mathcal{T}_{l_n}$. $\boldsymbol{f}$ is the learning second-order dynamic system. $\mathcal{R}$ will be conditioned on the task parameter frames $\mathcal{F}$. The learned TP-RMP will be composable to other RMPs, which are associated with the leaf nodes $\{l_n\}_{n=1}^L$ in the RMP-tree.

4.2 Model formulation

To begin, we aim to realize the RMP learning model as the form of Equation (3.1), where the potential field has to be learned as the main governing factor generating the desired motion. At the same time, the damping force in Equation (3.1) also needs to be optimized so that the generated motion reaches the potential minima without oscillating. Thus, the learning concept is somewhat similar to UMIC [KK17]. However, in this work, we utilize the GMM statistics learned from the demonstrations to set the parameters of the potential field and the dissipative field. Moreover, the Riemannian metric and its curvature are also induced based on the Gaussian covariances of the GMM. Hence, the learned fields and metrics benefit from the task parameterization of TP-GMM.

In general, after learning the TP-GMM Θ using N demonstrations, the potential field, and the damping parameters will be optimized from the same N demonstrations. The general learning pipeline of the framework is presented in Figure 4.1.

The following subsections describe concrete mathematical formulation deriving TP-RMP from TP-GMM.



Learning
Demos $\{\mathbf{Z}_i, \mathcal{F}_i\}_{i=1}^N$
TP-GMM Learning
Θ
Optimizing fields
Θ
$\Phi^0, \mathbf{D}$
Reproduction
Task Parameters: $\{\mathbf{A}_t^{(p)}, \mathbf{b}_t^{(p)}\}_{p=1}^P$
Global GMM
$\hat{\Theta}$
Compute LfD RMP
$\mathcal{R} = (\mathbf{f}, \mathbf{M})$
$\mathcal{R}$
Other RMPs
$(\mathbf{q}_{t+1}, \dot{\mathbf{q}}_{t+1})$
RMPflow
$\mathbf{a}(\mathbf{q}_t, \dot{\mathbf{q}}_t)$
Impedance Controller

Figure 4.1: The overview diagram of the proposed framework. In the learning phase, given multiple kinesthetic demonstrations, we encode the skill dynamic variations into TP-GMM Θ and then optimize the potential field $\boldsymbol{\Phi}^0$ and dissipative field $\boldsymbol{D}$ parameters. During reproduction, given the current task parameter frames $\mathcal{F}_t$, an instance of global GMM $\hat{\Theta}$ is computed. Then, with the field parameters, the LfD RMP is defined. Finally, together with other RMPs encoding other behaviors, e.g. collision avoidance, the final acceleration policy at the configuration space $\mathcal{C}$ is computed for the current robot state.

4.2.1 Lagrange energy $\mathcal{L}$

In this section, we derive the realization of RMP in Equation (3.1) from the Lagrange energy of a virtual system on a smooth manifold $\mathcal{M}$. The Lagrangian is defined as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, \dot{\boldsymbol{x}}) &= T - V \\ &= \frac{1}{2}\dot{\boldsymbol{x}}^{\intercal}\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}} - \Phi(\boldsymbol{x})\end{aligned} \tag{4.1}$$

where T is the kinematic energy and V or $\Phi(\boldsymbol{x})$ are the potential energy. In Geometric Mechanic [BL04] view, this kinematic energy guarantees that the Riemannian metric is equivalent to the mass matrix $\boldsymbol{M} = \frac{\partial^2 \mathcal{L}}{\partial \dot{x}^2}$. By applying Euler-Lagrange equation on $\mathcal{L}$, we get:

$$\begin{aligned}\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{x}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} &= \mathbf{0} \\ \boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}}(\dot{\boldsymbol{x}}^{\intercal}\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}}) + \frac{\partial \Phi}{\partial \boldsymbol{x}} &= \mathbf{0} \\ \boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{\partial \Phi}{\partial \boldsymbol{x}} &= \mathbf{0}\end{aligned} \tag{4.2}$$

where the curvature term $\boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}}(\dot{\boldsymbol{x}}^{\intercal}\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}})$ is equivalent to the Coriolis force [CMI+19]. Then, the canonical form RMP $(\boldsymbol{a}, \boldsymbol{M})^{\mathcal{M}}$ derived from the Lagrangian $\mathcal{L}$ is defined as:

$$\boldsymbol{a} = \boldsymbol{M}(\boldsymbol{x})^{-1}(-\frac{\partial \Phi}{\partial \boldsymbol{x}} - \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}})) \tag{4.3}$$

This RMP resides in Lagrangian RMPs family, which are derived from Lagrange energies. The dynamic system evolved by this RMP is conservative, whose Hamiltonian $\mathcal{H}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = T + V$ (i.e. total energy) is constant [RWX+21].

However, convergence property is required in skill reproduction. Consider the object picking task, in which the task parameters are the frames associated with the end-effector and the object of interest, the end-effector needs to accelerate from its idle pose and de-accelerate when reaching the object. We need a form of dissipative field to siphon the system energy so that, by the end of the motion generation given the task situations, the system converges at the object of interest. A direct way to realize this behavior is adding a damping force $\Psi(\boldsymbol{x}, \dot{\boldsymbol{x}})$ to the RMP:

$$\boldsymbol{a} = \boldsymbol{M}(\boldsymbol{x})^{-1}(-\frac{\partial \Phi}{\partial \boldsymbol{x}} - \Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}})) \tag{4.4}$$

We aim to learn the potential field $\Phi(\boldsymbol{x})$ and the dissipative field/damping force $\Psi(\boldsymbol{x}, \dot{\boldsymbol{x}})$ so that the Hamiltonian of the system $\mathcal{H}_t(\boldsymbol{x}, \dot{\boldsymbol{x}}) \to 0$ as $t \to \infty, \boldsymbol{x} \to \boldsymbol{\zeta}^*, \dot{\boldsymbol{x}} \to \mathbf{0}$, where $\boldsymbol{\zeta}^*$

is the designate goal pose during skill reproduction. Note that the designate goal can be conditioned on the task parameters $\mathcal{F}$.

The remainder of this section describes TP-RMP dynamical system formulations and learning processes.

4.2.2 Riemannian metric & curvature

The Riemannian metric can be induced on the spatial distribution of the demonstrations, which are represented by the Gaussian covariances in the GMM. The idea is to utilize the variation of the demonstrations to shape the learning metric geometry and thus constrain the motion generation. In intuition, this Riemannian metric expands or contracts the field forces conforming with the Gaussian covariances.

Given the task parameter frames $\mathcal{F} = \{\mathbf{A}^{(p)}, \mathbf{b}^{(p)}\}_{p=1}^{P}$ and the learned TP-GMM Θ, an instance of GMM with parameters $\hat{\Theta} = \{\pi_k, (\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k)\}_{k=1}^{K}$ is computed in the static global frame using Equation (3.6), the Riemannian metric is then defined as the weighted sum of the inverse of Gaussian covariances of $\hat{\Theta}$:

$$\boldsymbol{M}(\boldsymbol{x}) = \gamma \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) \hat{\boldsymbol{\Sigma}}_k^{-1} \tag{4.5}$$

where γ is a scaling parameter and $\tilde{w}_k$ is the normalized probability density function of the GMM:

$$\tilde{w}_k(\boldsymbol{x}) = \frac{\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k)}{\sum_{j=1}^{K} \mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)}, \quad 0 \leq \tilde{w}_k \leq 1, \quad \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) = 1 \tag{4.6}$$

The scaling parameter γ is a hyperparameter regulating the system inertia, e.g., the lower γ, the less system inertia and thus the more system sensitivity to the acting forces. Therefore, varying γ also has the effect of varying the importance weight of this RMP when combining with other RMPs using RMPflow.

Similar to our work, [RLR+20] learns the Riemannian metric encoded in a neural network that warps the field forces constraining the system to follow the demonstration direction of motion. However, their approach is limited in two aspects:

- their Riemannian metric does not capture the local dynamic variations of the demonstrations. Despite adding a small positive offset ϵ at the diagonal, the Riemannian metric is prone to be sharply narrow along the nominal direction, and thus the system evolves with a significant stiffness to follow the demonstrated direction motion during execution. This effect is undesirable for compliance applications when working with humans.

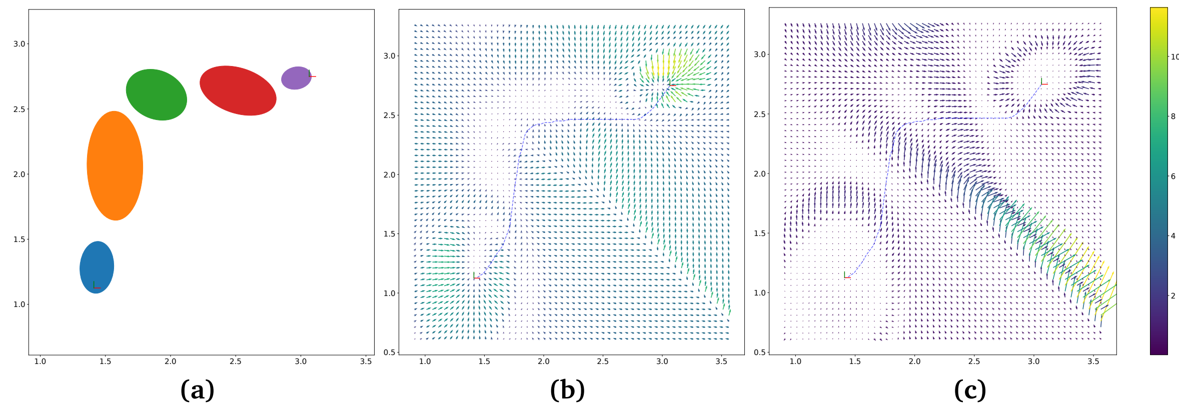


Figure 4.2: The effect of Riemannian metric warping the potential gradient field. The blue dashed line is the nominal trajectory that the system should follow. The color bar depicts the magnitude of the vectors. The **(a)** The GMM $\hat{\Theta}$ computed by given frames $\mathcal{F}$. **(b)** Learned potential gradient field with parameters set by $\hat{\Theta}$. **(c)** The potential gradient field warped by the Riemannian metric conditioned on $\hat{\Theta}$. The vectors in the region around nominal trajectory are expanded along the nominal direction, while being contracted in the orthogonal directions.

- the neural network may suffer from the curse of dimensionality and thus require many demonstrations to generalize in the task space adequately.

Our approach overcomes these issues and provides an analytically Riemannian metric rather than black-box learning. An illustration of the Riemannian metric that warps the potential gradient field is presented in Figure 4.2.

This Riemannian metric formulation induces a curvature term $\boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}})$ as in Equation (4.4). The curvature term is also know as the Coriolis force in the view of Geometric Mechanic [BL04]:

$$\boldsymbol{c}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{x}}(\dot{\boldsymbol{x}}^\intercal \boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}}) \tag{4.7}$$

where the derivation of first term is:

$$\begin{aligned}\dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} &= \frac{\partial}{\partial t}(\gamma \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x})\hat{\boldsymbol{\Sigma}}_k^{-1})\dot{\boldsymbol{x}} \\ &= \gamma \sum_{k=1}^{K} \frac{\partial \tilde{w}_k(\boldsymbol{x})}{\partial t}\hat{\boldsymbol{\Sigma}}_k^{-1}\dot{\boldsymbol{x}} \\ &= \gamma \sum_{k=1}^{K} [(\frac{\partial \tilde{w}_k(\boldsymbol{x})}{\partial \boldsymbol{x}})^{\mathsf{T}}\dot{\boldsymbol{x}}]\hat{\boldsymbol{\Sigma}}_k^{-1}\dot{\boldsymbol{x}} \\ &= \gamma \sum_{k=1}^{K} \tilde{w}_k[\sum_{j=1}^{K} \tilde{w}_j(\boldsymbol{x}-\hat{\boldsymbol{\mu}}_j)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_j^{-1}\dot{\boldsymbol{x}} - (\boldsymbol{x}-\hat{\boldsymbol{\mu}}_k)^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_k^{-1}\dot{\boldsymbol{x}}]\hat{\boldsymbol{\Sigma}}_k^{-1}\dot{\boldsymbol{x}}\end{aligned} \tag{4.8}$$

and the second term is:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{x}}(\dot{\boldsymbol{x}}^{\mathsf{T}}\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}}) &= \dot{\boldsymbol{x}}^{\mathsf{T}}(\frac{\partial \boldsymbol{M}(\boldsymbol{x})}{\partial \boldsymbol{x}})\dot{\boldsymbol{x}} \\ &= \gamma \sum_{k=1}^{K} \frac{\partial \tilde{w}_k(\boldsymbol{x})}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_k^{-1}\dot{\boldsymbol{x}} \\ &= \gamma \sum_{k=1}^{K} \tilde{w}_k[\sum_{j=1}^{K} \tilde{w}_j\hat{\boldsymbol{\Sigma}}_j^{-1}(\boldsymbol{x}-\hat{\boldsymbol{\mu}}_j) - \hat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{x}-\hat{\boldsymbol{\mu}}_k)]\dot{\boldsymbol{x}}^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_k^{-1}\dot{\boldsymbol{x}}\end{aligned} \tag{4.9}$$

In intuition, the term $-\boldsymbol{M}(\boldsymbol{x})^{-1}\boldsymbol{\xi}_M(\boldsymbol{x},\dot{\boldsymbol{x}})$ in Equation (4.4) simply ensures stable and geometrically consistent behavior to this Riemannian metric formulation. Indeed, the manifold $\mathcal{M}$ is completely characterized embedded in the ambient task space manifold by the Riemannian metric $\boldsymbol{M}(\boldsymbol{x})$.

4.2.3 Potential field

Despite the Riemannian metric amplifies the forces in the demonstrated direction of motion corresponding to the GMM $\hat{\Theta} = \{\pi_k, (\hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)\}_{k=1}^{K}$ parameters, the desired motion of the Equation (4.4) is mainly guided by the the negative potential gradient $-\frac{\partial \Phi}{\partial x}$. Warping with the Riemannian metric, the term $-\boldsymbol{M}(\boldsymbol{x})^{-1}\frac{\partial \Phi}{\partial x}$ can be view as the negative *natural* gradient of the potential on the manifold $\mathcal{M}$.

Similar to the Riemannian metric formulation as Equation (4.5), the potential function is also a weighted sum of the local potentials modelled by $\hat{\Theta}$:

$$\Phi(\boldsymbol{x}) = \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x})\phi_k(\boldsymbol{x}) \tag{4.10}$$

where each local potential ϕ_k is associated with the k-th Gaussian of the GMM $\hat{\Theta}$:

$$\phi_k(\boldsymbol{x}) = \phi_k^0 + \begin{cases} \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k\|^2_{\hat{\boldsymbol{\Sigma}}_k^{-1}}, & \|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k\|_{\hat{\boldsymbol{\Sigma}}_k^{-1}} \leq \delta \\ \delta(\|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k\|_{\hat{\boldsymbol{\Sigma}}_k^{-1}} - \frac{1}{2}\delta), & otherwise \end{cases} \tag{4.11}$$

where δ is the threshold dividing the quadratic and norm sections of the potential. ϕ_k^0 is the bias potential level. This design of local potential is similar to the well-known Huber loss [Hub64] in Machine Learning literature. Note that in this case we use Mahalanobis norm to reflect the local dynamics variation encoded in the GMM covariances, and thus, with the same distance, the negative potential gradients pull differently across motion directions. This potential function also ensures that the attractive acceleration always has a unit magnitude except in the neighborhood δ of the Gaussian mean $\hat{\boldsymbol{\mu}}_k$ where it smoothly decreases to zero. A simple alternative to this design using only quadratic is introduced in [KK17]. Their drawback is that the gradient of a quadratic function increases linearly with the distance, which can cause undesirably large accelerations far away from the GMM.

Then the potential gradient is derived as follows:

$$\frac{\partial \Phi}{\partial \boldsymbol{x}} = \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) \frac{\partial \phi_k(\boldsymbol{x})}{\partial \boldsymbol{x}} + \sum_{k=1}^{K} \frac{\partial \tilde{w}_k(\boldsymbol{x})}{\partial \boldsymbol{x}} \phi_k(\boldsymbol{x}) \tag{4.12}$$

The two terms in the potential gradient can be interpreted as:

- **Attraction force** corresponds to the first term:

$$\sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) \frac{\partial \phi_k(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{cases} \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) \hat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k), & \|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k\|_{\hat{\boldsymbol{\Sigma}}_k^{-1}} \leq \delta \\ \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) \frac{\hat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)}{\|\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k\|_{\hat{\boldsymbol{\Sigma}}_k^{-1}}}, & otherwise \end{cases} \tag{4.13}$$

  The attraction force pulls the system towards the nearest Gaussian mean, which attracts the system back to nominal motion. It mainly reflects the *reactive* property of the acceleration policy. As aforementioned, outside the threshold δ, the attraction force is a normalized vector towards to nearest Gaussian, which renders a stable pull independent of the distance towards the attraction point.

- **Nominal force** corresponds to the second term:

$$\sum_{k=1}^{K} \frac{\partial \tilde{w}_k(\boldsymbol{x})}{\partial \boldsymbol{x}} \phi_k(\boldsymbol{x}) = \sum_{k=1}^{K} \tilde{w}_k [\sum_{j=1}^{K} \tilde{w}_j \hat{\boldsymbol{\Sigma}}_j^{-1}(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_j) - \hat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{x} - \hat{\boldsymbol{\mu}}_k)] \phi_k(\boldsymbol{x}) \tag{4.14}$$

  Considering the system pose is close to the current Gaussian mean $\hat{\boldsymbol{\mu}}_k$, this nominal force pulls the system from the current Gaussian mean to the next Gaussian

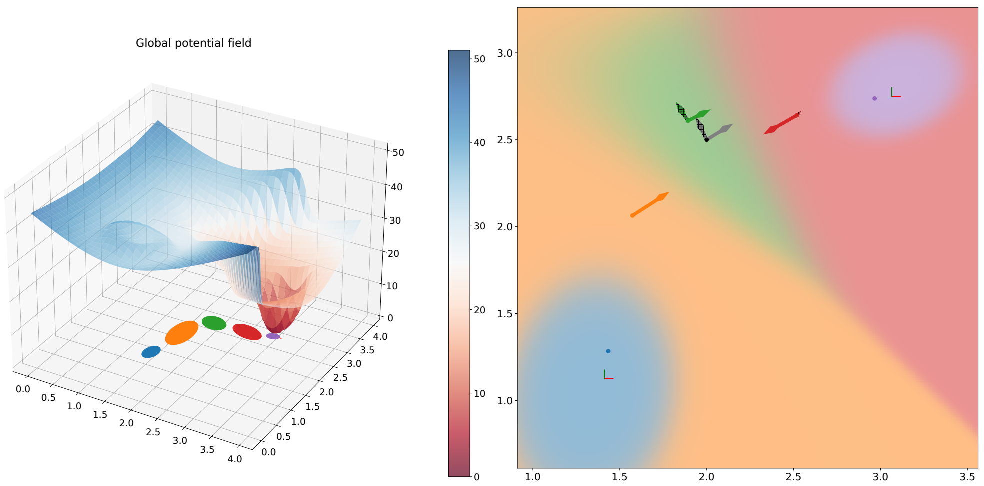


Figure 4.3: (a) The potential Φ that is conditioned on a GMM projected on x-y plane. **(b)** An instance of attraction forces (shaded vectors) and nominal forces applied at the (black) system state coordinate $(2., 2.5)$. The background is colored following the weight values $\{\tilde{w}_k\}_{k=1}^K$, depicting the dominant influence of the nearest Gaussian distribution. For clarity, the component forces are colored and placed at their respective Gaussians, while the weighted sum forces are colored black and placed at the system state. As expected, the weighted sum nominal force currently guides the system to the red Gaussian mean, while the weighted sum attraction force pulls the system the nearest green Gaussian mean. Note that other forces at further components are too small to visualize.

mean $\hat{\boldsymbol{\mu}}_{k+1}$ (see Figure 4.3 for visual explanation). Note that if all the Gaussian distributions in the GMM $\hat{\Theta}$ become standard distributions, this nominal force reduces to the nominal force similar to [KK17]. In this case, however, we utilize the local dynamics captured in the Gaussian covariance shapes. Thus, the system is expected to track the sequence of Gaussian means and stop at $\hat{\boldsymbol{\mu}}_K$. In an optimal control sense, this tracking behavior is the same as using LQC to generate a sequence of control acceleration tracking the Gaussian means as in [LGD+21; RGK+20].

With this potential formulation, the remaining learning parameters are the set of K bias potential levels $\boldsymbol{\Phi}^0 = \{\phi_k^0\}_{k=1}^K$, which regulate the energy level of the potential field. The following section explains the learning process of $\boldsymbol{\Phi}^0$ to match the desired demonstrated motion.

4.2.4 Learning potential field from demonstrations

The LfD problem in this case is viewed as learning the mentioned natural gradient descent $-\boldsymbol{M}(\boldsymbol{x})^{-1}\frac{\partial \Phi}{\partial \boldsymbol{x}}$, and then at the second step, learning the dissipative field described in Section 4.2.5.

In fact, the solution of the natural gradient descent learning problem is not unique. There are many different potentials and metric combinations that can result in the same desired acceleration policy. Hence, together with the designed Riemannian metric in Equation (4.5), we bias the learning process so that, by learning the potential level $\boldsymbol{\Phi}^0$, the negative natural gradient is aligned with the demonstrated motion direction $\hat{\dot{\boldsymbol{\zeta}}} = -\boldsymbol{M}(\boldsymbol{x})^{-1}\frac{\partial \Phi}{\partial \boldsymbol{x}}$. Given the N demonstrations $\{\boldsymbol{Z}_i\}_{i=1}^N$ used to optimize the TP-GMM Θ, the potential learning is designed as minimizing the following constrained Quadratic Program (QP) accounted for N demonstrations:

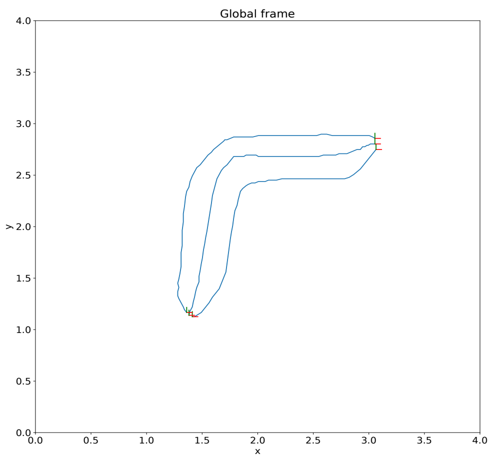
$$\underset{\boldsymbol{\Phi}^0}{min} \sum_{i=1}^{N}\sum_{t=1}^{T_i} \left\| \hat{\dot{\boldsymbol{\zeta}}}_{i,t} - (-\boldsymbol{M}_i(\boldsymbol{\zeta}_{i,t})^{-1}\frac{\partial \Phi_i(\boldsymbol{\zeta}_{i,t})}{\partial \boldsymbol{x}}) \right\| \tag{4.15}$$

s.t.

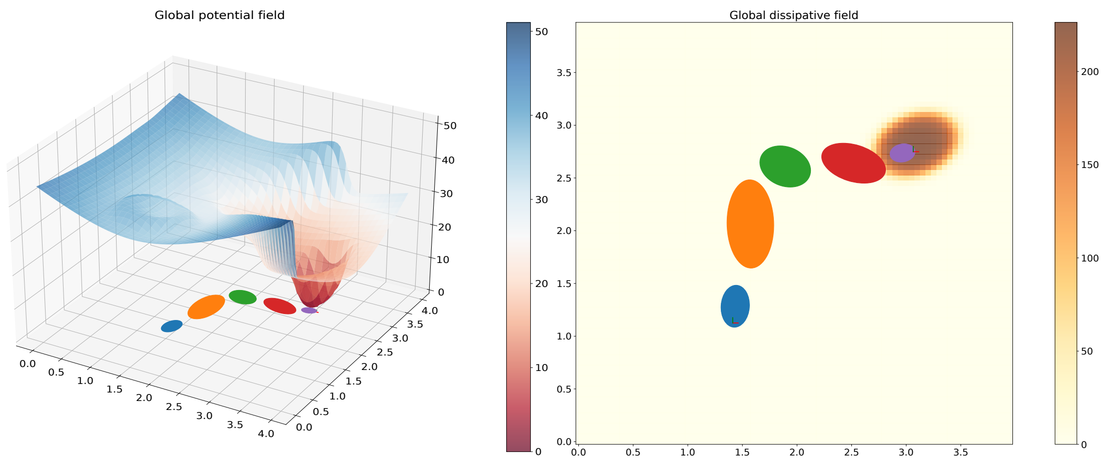
$$0 \leq \phi_K^0 \leq \phi_{k+1}^0 \leq \phi_k^0 \quad \forall k = 1, ..., K-1 \tag{4.16}$$

where for each demonstrations i, a GMM with parameters $\hat{\Theta}_i = \{\pi_k, (\hat{\boldsymbol{\mu}}_{i,k}, \hat{\boldsymbol{\Sigma}}_{i,k})\}_{k=1}^K$ is computed using Equation (3.6) from the associated task parameter frames $\mathcal{F}_i$, i.e. the frames at the time recording the demonstration i. By the GMM $\hat{\Theta}_i$, the corresponding potential gradient $\frac{\partial \Phi_i}{\partial \boldsymbol{x}}$ and Riemannian metric $\boldsymbol{M}_i$ are computed by Equation (4.12) and (4.5), respectively. Note that the constraints in QP (4.15) ensure that the energy always positive and consistent, i.e. the potential field has the shape of descending valley having a plateau at the end of the demonstrations, in accordance of the GMM $\hat{\Theta}$ shape (see Figure 4.4).

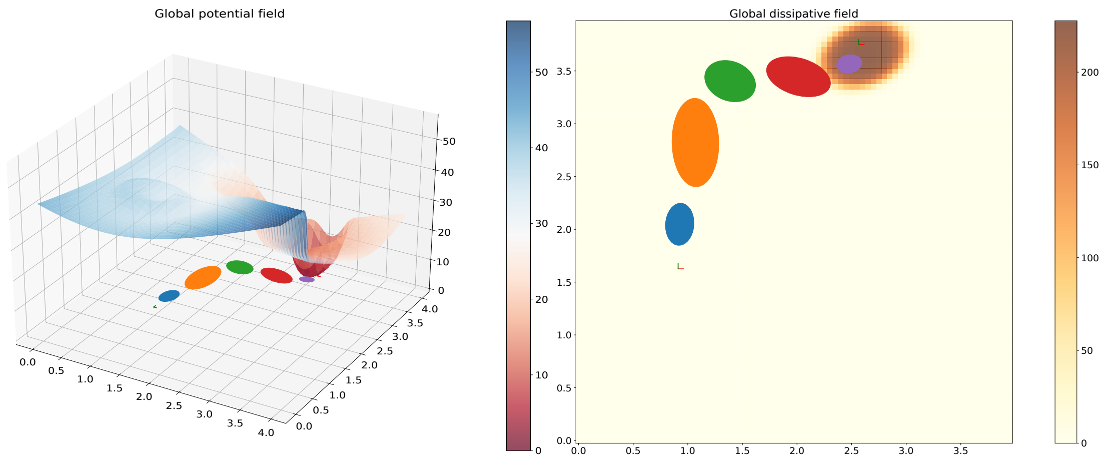
This constrained QP formulation is convex and has a unique global solution, which also has significantly fewer parameters and constraints to optimize than [KK17; RLR+20], i.e., K parameters and K inequality constraints. In this case, we only need to optimize the number of potential field parameters corresponding to the number of Gaussians K, while in [KK17; RLR+20] they optimize the number parameters equal to the number of points in the demonstrations. In fact, the potential field specified in Equation (4.10) utilizes the GMM statistics that encodes local dynamic variations, thus reducing the optimizing parameters to reproduce the desired motion. In practice, it takes a few second to optimize the convex QP (4.15) using the popular solver *cvxpy* [DB16].



(a) Demonstrations



(b) **(c)**



(d) **(e)**

Figure 4.4: Shape shifting potential field and dissipative field in accordance to task parameter frames $\mathcal{F}$. **(a)** The demonstrations that TP-RMP is learned from. **(b),(c),(d),(e)** Illustrations of shape shifting potential field and dissipative field. Between the two cases, *Start* frame origin is shifted by $(-0.5, 0.5)$, while *end* frame origin is shifted by $(-0.5, 1.0)$

4.2.5 Dissipative field

Similar to previous sections, given the GMM $\hat{\Theta}$, the dissipative field in Equation (4.4) is defined as the weighted sum:

$$\begin{aligned}\Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) &= \boldsymbol{B}(\boldsymbol{x})\dot{\boldsymbol{x}} \\ &= \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x})\psi_k(\dot{\boldsymbol{x}})\end{aligned} \tag{4.17}$$

where the k-th local dissipative field $\psi_k(\dot{\boldsymbol{x}}) = d_k \boldsymbol{I}\dot{\boldsymbol{x}}$ is associated with the k-th Gaussian. In this context, the damping matrix is $\boldsymbol{B}(\boldsymbol{x}) = \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) d_k \boldsymbol{I}$. Hence, the learning parameters of the field is the set of K damping coefficients $\boldsymbol{D} = \{d_k\}_{k=1}^{K}$ associated with K Gaussian components.

Note that the uniform dissipative field is chosen for each Gaussian for stability. In practice, this simple dissipative field does not limit the model capacity to perform various motion dynamics. Indeed, similar to the potential field (4.10), the dissipative field model is varied with the K number of Gaussian components: the larger K, the more granularity of the model.

4.2.6 Learning dissipative field from demonstrations

As mentioned, the objective is to learn the dissipative field to decrease the system's total energy approaching zero as it is approaching the goal pose during execution. Note that we can explicitly choose a zero coordinate of a task parameter frame as the designated goal.

Given a demonstration $\boldsymbol{Z}_i = \{\boldsymbol{\zeta}_{i,t}\}_{t=1}^{T_i}$ and its associated task parameter frames $\mathcal{F}_i = \{\boldsymbol{A}_i^{(p)}, \boldsymbol{b}_i^{(p)}\}_{p=1}^{P}$, the GMM $\hat{\Theta}_i$ and its corresponding potential Φ_i and Riemannian metric $\boldsymbol{M}_i$ are computed similar to Section 4.2.4. We then try to match the system total energy defined at the beginning of the demonstration $t = 1$:

$$\begin{aligned}\mathcal{H}_i &= T + V \\ &= \frac{1}{2}\dot{\boldsymbol{\zeta}}_{i,1}^{\intercal}\boldsymbol{M}_i(\boldsymbol{\zeta}_{i,1})\dot{\boldsymbol{\zeta}}_{i,1} + \Phi_i(\boldsymbol{\zeta}_{i,1})\end{aligned} \tag{4.18}$$

and the dissipated energy over the demonstrated trajectory:

$$\mathcal{D}_i = \sum_{t=1}^{T_i} \Psi(\boldsymbol{\zeta}_{i,t}, \dot{\boldsymbol{\zeta}}_{i,t}) \cdot \dot{\boldsymbol{\zeta}}_{i,t}\Delta t \tag{4.19}$$

where $\cdot$ is the dot product, and Δt is the sampling interval of the demonstration.

Then, the dissipative field parameters $\boldsymbol{D}$ are optimized with the following Linear Program (LP) holistically accounted for N demonstrations:

$$\min_{\boldsymbol{D}} \sum_{i=1}^{N} \|\mathcal{H}_i - \mathcal{D}_i\| \tag{4.20}$$

s.t.

$$\epsilon \leq d_k \leq d_{k+1} \leq d_K \quad \forall k = 1, ..., K-1 \tag{4.21}$$

where ϵ is the positive lower bound of the damping coefficient. In fact, there are many solutions of the dissipative field that realize the matching energy. However, we bias the learning process by defining the constraints for the parameters $\boldsymbol{D}$ so that the non-negative damping coefficients are monotonically increasing as the system approaching the goal pose. These parameters ensure the consistent behavior for the reactive property that, for all initial conditions, the system motion is gradually increasingly damped until reaching the goal pose.

Similar to the QP (4.15), this LP is convex and has unique solution. It takes less than a second to optimize with *cvxpy*.

Overall, with the TP-RMP formulation in this section, the TP-RMP parameters are defined as the tuple $(\Theta, \boldsymbol{\Phi}^0, \boldsymbol{D})$, where Θ is the TP-GMM parameters, and $\boldsymbol{\Phi}^0, \boldsymbol{D}$ are the bias potential levels and the damping coefficients, respectively.

4.3 Stability Analysis

In this section, we analyze the stability of the TP-RMP under different task parameters $\mathcal{F}$ and initial system states $(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0)$. Note that in terms of reactive acceleration policy, i.e., unifying motion generation and impedance control [KK17], tracking a reference trajectory is not a primary objective. Instead, our focus is to show that under different system states, by evolving with acceleration policy (Equation (4.4)) derived from the TP-RMP, the system converges to the goal conditioned on the task parameters.

Lemma 4.3.1
Given the TP-GMM $\Theta = \{\pi_k, \{(\boldsymbol{\mu}_k^{(p)}, \boldsymbol{\Sigma}_k^{(p)})\}_{p=1}^{P}\}_{k=1}^{K}$, suppose $\forall 1 \leq p \leq P, 1 \leq k \leq K : \boldsymbol{\Sigma}_k^{(p)}$ are positive-definite. Then, with any GMM $\hat{\Theta}$ in the global frame computed by task parameters $\mathcal{F}$, the Riemannian metric $\boldsymbol{M}(\boldsymbol{x})$ defined in Equation (4.5) is positive-definite with $\forall x \in \mathcal{M}$.

Proof. If $\forall 1 \leq p \leq P, 1 \leq k \leq K : \mathbf{\Sigma}_k^{(p)}$ are positive-definite, and thereby invertible, then by any affine transformation of $\mathcal{F}$, $\forall 1 \leq k \leq K : \hat{\mathbf{\Sigma}}_k$ computed by Equation (3.6) are positive-definite. It follows that, by Equation (4.5), the computed Riemannian metric $\forall \boldsymbol{x} \in \mathcal{M}, \boldsymbol{M}(\boldsymbol{x})$ is positive-definite by construction $0 \leq \tilde{w}_k \leq 1, \sum_{k=1}^{K} \tilde{w}_k(\boldsymbol{x}) = 1$. □

This lemma ensures that, if the GMM covariances represented the demonstration variation do not have a null-space, the designed Riemannian metric is strictly positive-definite for every point in the task space. In physical view, it follows that the kinematic energy reserved in the inertia matrix $\boldsymbol{M}$ is ensured to exist at every point. Hence, there must be sufficient statistics of the skill behavior by having a reasonable number of demonstrations. Otherwise, it incurs highly stiff behaviors along the demonstrated motion direction, and the system may be unstable outside of the demonstration region.

To analyze the TP-RMP stability, we present the following theorem:

Theorem 4.3.2
Given the learned TP-RMP $(\Theta, \mathbf{\Phi}^0, \boldsymbol{D})$, *suppose* $\forall 1 \leq p \leq P, 1 \leq k \leq K : \mathbf{\Sigma}_k^{(p)}$ *are positive-definite. Then, with any GMM* $\hat{\Theta}$ *in the global frame computed by task parameters* $\mathcal{F}$, *the system converges to a positive invariant set* $\mathcal{O}_\infty = \{(\boldsymbol{x}, \dot{\boldsymbol{x}}) : \nabla\Phi(\boldsymbol{x}) = \boldsymbol{0}, \dot{\boldsymbol{x}} = \boldsymbol{0}\}$ *starting from any initial state* $(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0)$.

Proof. By Lemma 4.3.1, $\boldsymbol{M}(\boldsymbol{x}) \succ 0, \forall \boldsymbol{x} \in \mathcal{M}$ under any GMM $\hat{\Theta}$. We thereby consider a Lyapunov candidate our system:

$$\boldsymbol{V}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \Phi(\boldsymbol{x}) + \frac{1}{2}\dot{\boldsymbol{x}}^\intercal \boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{x}} \tag{4.22}$$

Taking time derivative of $\boldsymbol{V}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ yields:

$$\dot{\boldsymbol{V}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \dot{\boldsymbol{x}}^\intercal \frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}} + \dot{\boldsymbol{x}}^\intercal \boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \frac{1}{2}\dot{\boldsymbol{x}}^\intercal \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} \tag{4.23}$$

The term $\boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}}$ can be obtained by rearranging Equation (4.4). Then substitute to previous equation yields:

$$\begin{aligned}\dot{\boldsymbol{V}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) &= \dot{\boldsymbol{x}}^\intercal \frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}} + \dot{\boldsymbol{x}}^\intercal \Big(-\frac{\partial \Phi(\boldsymbol{x})}{\boldsymbol{x}} - \Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}})\Big) + \frac{1}{2}\dot{\boldsymbol{x}}^\intercal \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} \\ &= -\dot{\boldsymbol{x}}^\intercal \Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) - \dot{\boldsymbol{x}}^\intercal \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{1}{2}\dot{\boldsymbol{x}}^\intercal \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}}\end{aligned}$$

By Equation (4.8) and (4.9), the term $-\dot{\boldsymbol{x}}^\intercal \boldsymbol{\xi}_M(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \frac{1}{2}\dot{\boldsymbol{x}}^\intercal \dot{\boldsymbol{M}}(\boldsymbol{x})\dot{\boldsymbol{x}} = 0$. Hence, (4.23) is simplified to:

$$\begin{aligned}\dot{\boldsymbol{V}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) &= -\dot{\boldsymbol{x}}^\intercal \Psi(\boldsymbol{x}, \dot{\boldsymbol{x}}) \\ &= -\dot{\boldsymbol{x}}^\intercal \boldsymbol{B}(\boldsymbol{x})\dot{\boldsymbol{x}}\end{aligned}$$

Note that, the damping matrix in Equation (4.17) is $\boldsymbol{B}(\boldsymbol{x}) \succ 0, \forall \boldsymbol{x} \in \mathcal{M}$ by construction for any GMM $\hat{\Theta}$. Hence, $\dot{\boldsymbol{V}}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ is negative-definite. By LaSalle's invariance principle [LaS60], the system is globally asymptotic stable. □

Furthermore, it is known that the system (4.4) is a simplified version of GDS, where the Riemannian metric $\boldsymbol{M}(\boldsymbol{x})$ is not velocity dependent. However, the stability property when combining this kind of system is still applicable with Theorem 2 in [CMI+19]. Indeed, if all subtask RMPs are generated by systems in the form of (4.4) (or in form of more general GDSs), the combined policy in the configuration space $\mathcal{C}$ is also in the form of (4.4) and hence Lyapunov stable.

4.4 Skill reproduction

In manipulation tasks, a skill requires coordinated and constrained motion of different parts of a robot. The problem with LfD methods in recent literature is that it is hard to combine other behaviors to satisfy dynamic task requirements incrementally. On the other hand, RMPflow [CMI+19] provides a framework to modular a complex task into subtasks that are easier to encode the motion behaviors, then combines them to realize the complex motion behaviors. However, it is usually not straightforward to design some motion behaviors, e.g., tracking a specific trajectory. Moreover, the skill may require conditioning on real-time task situations; for example, the picking skill needs to adapt for moving object targets.

To address these problems, we first construct an RMP-tree with root node in the configuration space $\mathcal{C}$, in particular, the joint space of the robot. The relevant robot body parts are added as child nodes of the root in the RMP-tree with edges given by the forward kinematics of the robot. Branching out further from these nodes are leaf nodes, corresponding to various subtask spaces encoding different motion objectives. At some leaf nodes, we can encode multiple human demonstrations with associated task parameters by learning TP-RMPs, while in other leaf nodes, the manually-designed RMPs such as collision avoidance or joint limiting can be handcrafted. At each control iteration, the overall configuration space policy $\boldsymbol{a}_r$ at root node is found using RMPflow as described in Section 3.3. To ensure stability, all manually-designed RMPs should have the form of Equation (4.4).

After learning phase, the high-level procedures for reproduction phase of a learned TP-RMP $(\Theta, \boldsymbol{\Phi}^0, \boldsymbol{D})$ described in Figure 4.1 is presented in Algorithm 4.1. The operations **pushforward**, **pullback**, **resolve** are introduced in Section 3.3.

Algorithmus 4.1 Skill reproduction

```
procedure COMPUTERMP(($\Theta, \boldsymbol{\Phi}^0, \boldsymbol{D}$), $\mathcal{F}$)
    Compute GMM $\hat{\Theta}$ from $\mathcal{F}$ using Equation (3.6).
    Compute LfD RMP $(\boldsymbol{f}, \boldsymbol{M})$ with $\hat{\Theta}, \boldsymbol{\Phi}^0, \boldsymbol{D}$ using Equation (4.10, 4.17, 4.5, 4.7).
return $(\boldsymbol{f}, \boldsymbol{M})$
end procedure
procedure REPRODUCE(($\Theta, \boldsymbol{\Phi}^0, \boldsymbol{D}$))
                                                            // TP-RMP initialization.
    Compute initial RMP $\mathcal{R}_0$ = ComputeRMP(($\Theta, \boldsymbol{\Phi}^0, \boldsymbol{D}$), $\mathcal{F}_0$)
    Set initial configuration state $(\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0)$.
                                                            // Main execution.
    while The system is not converged do
        if Received task parameters $\mathcal{F}_t$ then
            $\mathcal{R}$ = ComputeRMP(($\Theta, \boldsymbol{\Phi}^0, \boldsymbol{D}$), $\mathcal{F}_t$)
            Update new RMP $\mathcal{R}$ to corresponding leaf node.
        end if
        Retrieve current configuration state $(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$.
        Execute pushforward operation using forward kinematics recursively from the
root to leaf nodes.
        Compute leaf node RMPs in the subtask spaces.
        Execute pullback operation recursively from leaf nodes to the root.
        Execute resolve operation to compute combined policy $\boldsymbol{a}_r(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$.
        Track reactive acceleration $\boldsymbol{a}_r$ by impedance controller.
    end while
end procedure
```

As a remark, regarding the task conditioning, we can design a goal-directing RMP to adapt the skill to new targets as argued in [RLR+20]. However, only goal-directing behavior may be insufficient as the skill may need to satisfy a specific motion pattern before reaching the goal. For instance, some inserting skills require top-down insertion; the end-effector is constrained to move downward or execute some wiggling behavior before inserting. These kinds of behaviors are hard to design manually and can be effectively captured by learning TP-RMP. Moreover, task parameterization can be extended to multiple frames, which can be viewed as waypoints to shape the computed GMM in the global frame. In turn, the GMM shapes the potential field, the dissipative field, and the structure of the Riemannian metric given the current task situations. Thus, this mechanism creates great flexibility of task conditioning, while manual design may need multiple RMPs to realize.

Another important remark of TP-RMP design is that, in Algorithm 4.1, the method does not require the task parameters to be static during execution. Indeed, the task parameters as a set of frame transformations $\mathcal{F}_t = \{\boldsymbol{A}_t^{(p)}, \boldsymbol{b}_t^{(p)}\}_{p=1}^P$ can be varied with time. Hence, an instance of GMM computed from $\mathcal{F}_t$ is also varied with time, and thus the induced fields are time-conditioned. This property implies that the skill behavior can be adapted in real-time, as long as TP-RMP receives new task parameters from vision or sensing systems. We will show the real-time adaptability of TP-RMP in the experiment section.

5 Experimentation

This section presents the experiment outline that supports the claims and theories developed in this thesis. All measurement units in this section are SI units, e.g., distances are in meters (m), and forces are in Newton (N).

5.1 Experiment setups

To demonstrate the *reactive*, *adaptive*, and *composable* properties of TP-RMP, we test the model in two settings:

- a simulated 2-DoFs point system that freely moves omni-directional in 2D task space. We also create a dataset of 15 skills, each having three demonstrations and their associated two task parameter frames (see Figure 5.1). The shape of the demonstrations are created to contain a great variety of curvatures but are constrained to a limited task space region for the ease of visualization. In all skills, each demonstration contains around 600 data points.
- a simulated 6-DoFs UR5 robot arm on *pyBullet* [CB21]. We test TP-RMP explicitly with the picking skill on $\mathbb{R}^3 \times \mathbb{S}^3$ task space with only three demonstrations, where each demonstration contains around 400 data points. The task parameter frames, in this case, are attached to the end-effector suction gripper and the object of interest (see Figure 5.2).

The control loop of the 2D point system is simply applying a double integrator to the acceleration policy, and then the next system state is fed back to the policy. In the case of the simulated UR5 robot arm, we track the acceleration policy using an impedance controller (see Algorithm 4.1). For all experiments, the control frequency is set to $f = 100Hz$, the same as the demonstration sampling frequency.

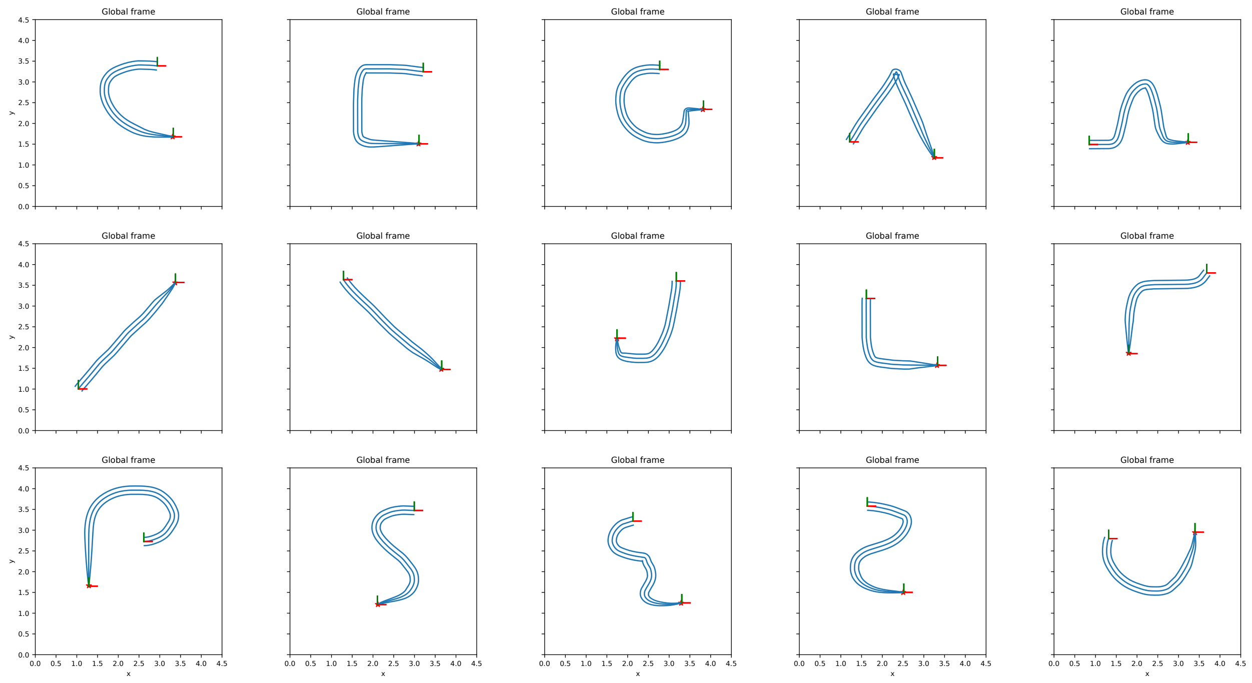


Figure 5.1: Illustration of 15 2D skills, each consists of 3 demonstrations. The task parameters are the *start* and *end* frames plotted in each skills as red and green lines. The small red start shows the goal point that the skill should reach, which is also at the zero coordinate of the *end* frame. All skills are recorded with 100Hz frequency.

5.1.1 Criteria

The performance of TP-RMP is evaluated with two schemes:

- **Tracking MSE**: evaluates how the learned skill reproduces the reference demonstration as well as the demonstration variation with the same recorded task parameter frames. In other words, this criteria shows how the learned model reproduces the nominal behavior of the skill given same demonstrated situations. The tracking performance is defined as the MSE evaluated between all the sampling points of the reference trajectory $\boldsymbol{Z} = \{\boldsymbol{\xi}_t\}_{t=1}^T$ and the reproduced trajectory $\{\boldsymbol{x}_t\}_{t=1}^T$:

$$MSE = \frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{\xi}_t - \boldsymbol{x}_t\|^2$$

  Note that in this case, the reference trajectory is the middle demonstration in Figure 5.1.

- **Goal error**: evaluates how robust that the learned TP-RMP guides the system reaching the designated goal points, under increasing task difficulties in terms of moving goal, disturbance, and when combining with other RMPs. We rely explicitly on the task parameter frame to define the goal point, in which the goal is conditioned to be the zero coordinate of the *end* frame (see Figure 5.1). We could define more frames to condition the skill behavior (e.g., as waypoints). However, for the clarity of the experiment, we define only the *start* and *end* frames for 2D skills, or *end-effector* and *object* frames for picking skill.

5.1.2 Baseline

In Section 5.2 of the tracking experiment, the tracking performance of TP-RMP is compared to a low-gain PD controller, which the control law is described by:

$$\boldsymbol{a} = \boldsymbol{K}_p(\boldsymbol{x} - \boldsymbol{x}_d) + \boldsymbol{K}_d(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_d) + \boldsymbol{\tau}_{ff}$$

where $(\boldsymbol{x}, \dot{\boldsymbol{x}})$ si the current system state. $\boldsymbol{K}_p, \boldsymbol{K}_d$ is the gain and damping matrices, respectively. $\boldsymbol{\tau}_{ff}$ is the feed-forward torque to compensate for the system dynamics, and $\boldsymbol{x}_d$ is the consecutive desired point on the reference trajectory. Both $\boldsymbol{\tau}_{ff}$, $\boldsymbol{x}_d$ are updated with a frequency of $f_d = 50Hz$, while the control loop frequency is set to 100Hz. Note that for the case of a 2D point system, the compensate torque is simply zero, while the $\boldsymbol{\tau}_{ff}$ is computed using forward dynamics of simulated UR5 robot.

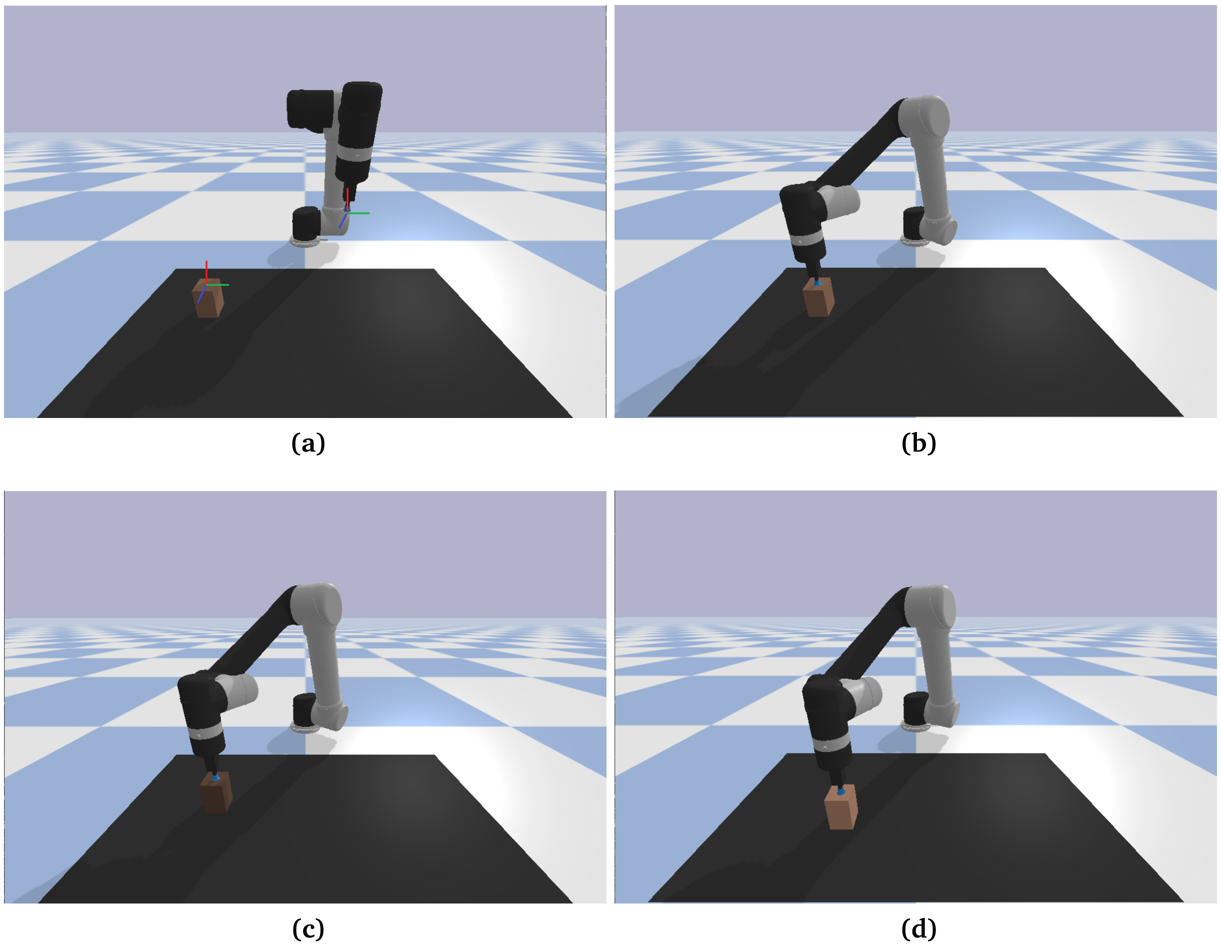
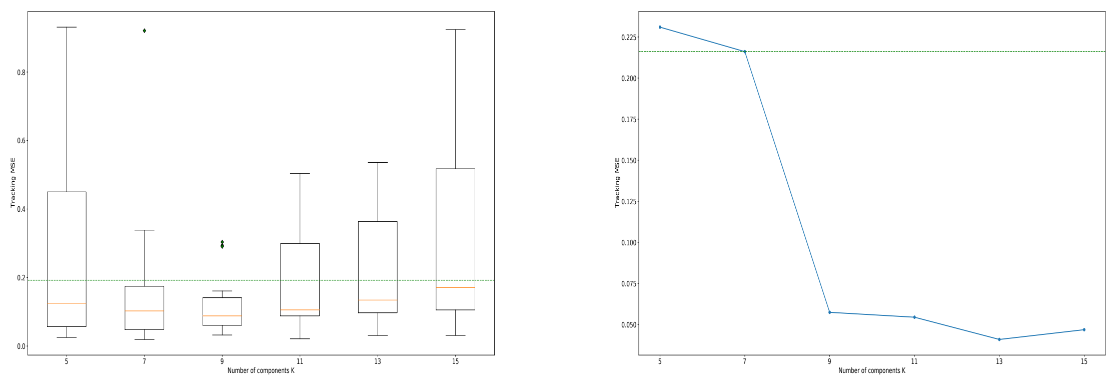


Figure 5.2: Picking skill demonstrations on *pyBullet* simulation with UR5 robot arm. For each demonstration with different object placement, we control the robot arm to reach the grasping point with point-position controller. We record the trajectory of the demonstration with 100Hz frequency and the task parameter frames attached the the end-effector and the object. The demonstration data can be seen in Figure 3.2a.



(a) 2D skills

(b) picking skill

Figure 5.3: Graph of tracking MSE over number of Gaussian components K.

5.1.3 Learning setups

For all learning TP-RMP processes, we set $\gamma = 0.25$ for 2D skill learning and $\gamma = 0.04$ for picking skill learning. γ is tuned so that the mass matrix $\boldsymbol{M}(\boldsymbol{x})$ has small enough eigenvalues for adequate system sensitivity in the range $1 - 100N$ applied forces.

The typical total training time of the EM process, optimizing potential and dissipative field for 600 data points, is under a minute with $K \leq 100$. The Python implementation of TP-RMP used for the experiments is published online[1].

5.2 Tracking experiment

The tracking experiment evaluates the skill reproducibility of the learned TP-RMP given the same task conditions as in demonstration. We argue that:

- in LfD context, the TP-RMP should at least reproduce the trajectory that bears a resemblance to the nominal demonstration given the same demonstrated task condition at the reproduction phase. **Tracking MSE** metric can be used to measure the resemblance between the reproduced trajectory and nominal demonstration.
- Then, given new task parameters, the trajectory reproduced by TP-RMP retains the nominal skill behavior.

[1]https://github.com/humans-to-robots-motion/tp-rmp

For 2D skills, for each chosen K, we learn 15 TP-RMPs corresponding to the 15 2D skills, then we collect 15 tracking MSEs for these models and present them as a box plot column. For picking skill, we learn a TP-RMP of chosen K and record its tracking MSE. The test data of both cases are presented in Figure 5.3. The dashed green line represents the average tracking MSE of the baseline PD controller. The green diamonds are the outliners for each column.

Figure 5.3a shows the tracking MSE evaluated for 15 2D skills over model selection K. The tracking performance varies over the number of Gaussian components K. For all number K, the medians of TP-RMP tracking MSEs are below the PD controller tracking performance, where $K = 9$ performs the best with 75% percentiles below the baseline. As expected, tracking performance is worse with very high 75% percentiles for small number $K = 5$, since the model capacity is low and there may not be enough waypoints to guide the system. However, surprisingly, there is a valley such that increasing K does not make tracking MSE decreases monotonically. The reason is that, by construction, the induced RMP is based on the shape of Gaussian covariances. As in Figure 5.4, there are two effects when increasing K:

- despite having more attractors as waypoints, when learning with EM process (see Section 3.4), the Gaussian component shapes are contracted in the demonstrated nominal direction and more expanded in the orthogonal directions. Hence, the region around the nominal demonstration has steeper potential gradients and thus induces instability.
- the Riemannian metric conditioned on these ill-formed Gaussian shapes also contracts the nominal force along the nominal direction.

These effects are more prominent if the curvature variation of the demonstrations is higher. Note that, for the case of picking skill in Figure 5.3b, the valley of suitable number K is much wider. The reason is that the recorded demonstration of picking skill has less curvature variation than the 2D skills.

Hence, for complex skill behaviors, it is imperative to select a reasonable number K. We can apply Bayesian Information Criterion (BIC) [Mur12] method for model selection. Another technique is to compute K is Dirichlet processes for infinite Gaussian mixture models [KTNS12].

Figure 5.5 shows the execution of the learned TP-RMPs models with $K = 9$ under the same task parameters as in demonstrations to assess the tracking performance. It can be seen that the reproduced trajectories bear a resemblance to the nominal demonstrations, with every point in the reproduced trajectories stay within 2σ to their closest Gaussian. Note that it is not a hard constraint to follow the nominal trajectory closely since the model also encodes the demonstration variation. Further illustrations of the underlying GMMs and the induced fields are presented in Appendix A.

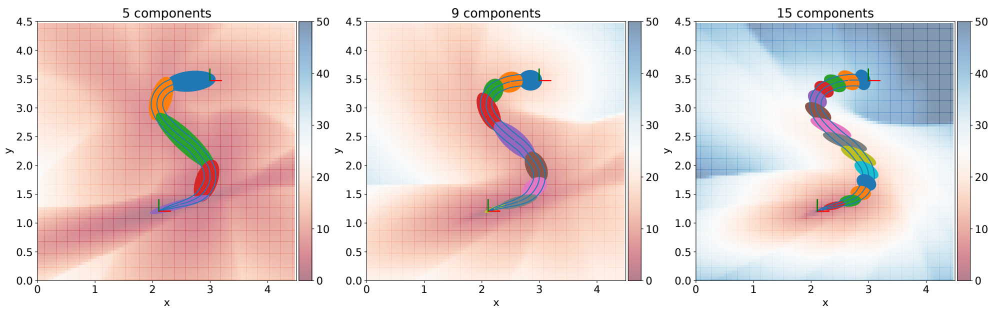


Figure 5.4: The shape of the GMM variances of 5, 9 and 15 components given the same task parameters. The ellipsoids represents the standard deviation region of each Gaussian. The background is the induced potential field of the GMM.

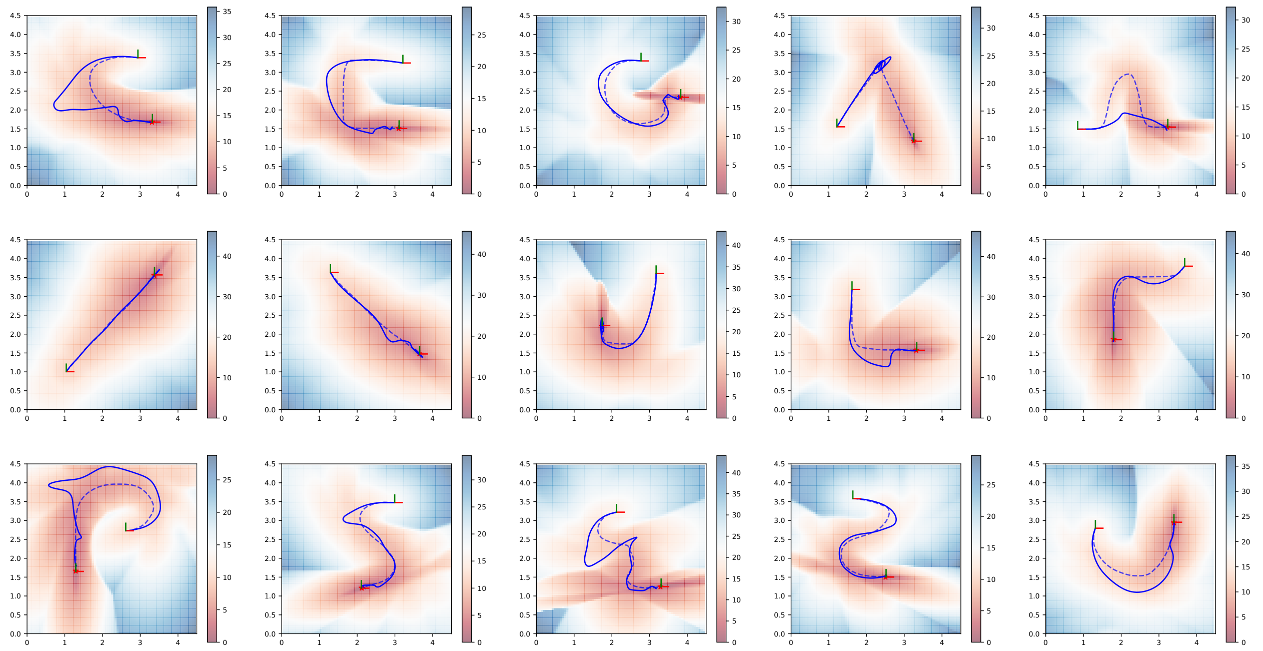


Figure 5.5: Reproduced trajectories of TP-RMPs models with $K = 9$ corresponding to 15 2D skills in Figure 5.1. The task parameters are the same as in demonstrations. Note that there is one fail reproduction in the first row and fourth column, perhaps because the demonstration has a discontinuity at the point $(2.2, 3.4)$.

5.3 Task adaptation experiment

After observing a set of demonstrations in demonstrated situations, we would like to generalize the skill to real-time new situations. Here, we benchmark the ability of TP-RMP to adapt the skill behavior to new task situations dynamically. As mentioned, the task parameter frames will influence the dynamic shape of GMM instances and their induced fields, thus adapting the skill behavior dynamically as the frames moving in real-time. In this section, we specifically focus on the goal convergence capability of TP-RMP realized by conditioning the goal at zero coordinate of the *end* frame. The reason is that there is no suitable nominal reference to track for new adapting situations or under disturbance schemes. Hence, the primary metric would be the **Goal error** in meters, representing the efficacy in solving the problem.

To benchmark the *reactive* and *adaptability* properties of TP-RMP, the setups for this experiment are:

- **Task adaptation**: We fix the *start* frame and condition the *end* frame moving in circle of radius $r = 0.2m$ and angular velocity $\omega = \pi$. For picking skill, we also fix the *end-effector* frame at the beginning and condition the object moving in a circle. The system starts at starting frame and will be guided by the dynamically updated RMP. When the system reaches close to the goal frame, the goal frame stops moving, and the system is expected to converge in close vicinity of the goal.
- **Task adaptation with increasing disturbances**: This benchmark has the same conditions as the **Task adaptation** benchmark, but with additional difficulty by applying an increasing disturbance force for each test run. The disturbance force is applied for over 100 timesteps $t \in [50, 150]$, which is orthogonal to the moving direction in reproduction (see Figure 5.6).
- **Task adaptation with increasing differences**: This benchmark has the same conditions as the **Task adaptation** benchmark. However, we increase the radius r of the moving goal frame for each test run and thus increase the degree of difference comparing to the demonstrated conditions.

Similar to tracking experiment 5.2, for the 2D setting, we evaluate the adaptation capability of 15 2D skills over the number of components K. In picking skill, for sufficient statistics, we test 10 runs with randomized angular velocities of moving object frame for each chosen K TP-RMP.

Figure 5.7 demonstrates the **Task adaptation** benchmark. In Figure 5.7a, it seems to be consistent with the suitable number of components K argument presented in the previous experiment. According to the data, there is a positive correlation between tracking performance and goal convergence. In this case, when reproducing skills with

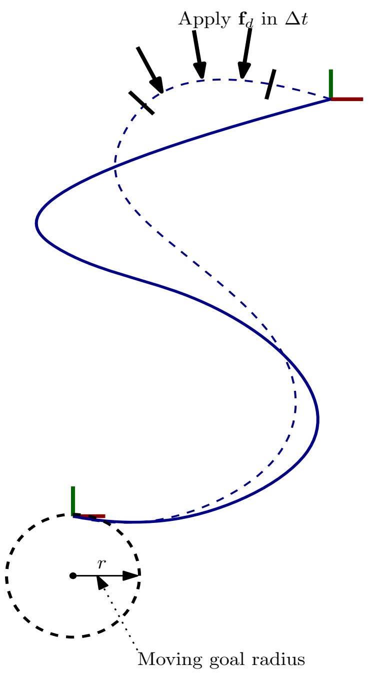
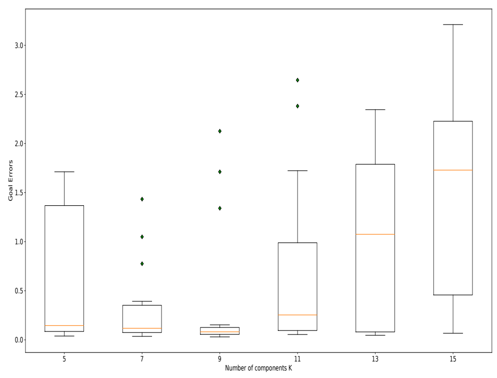
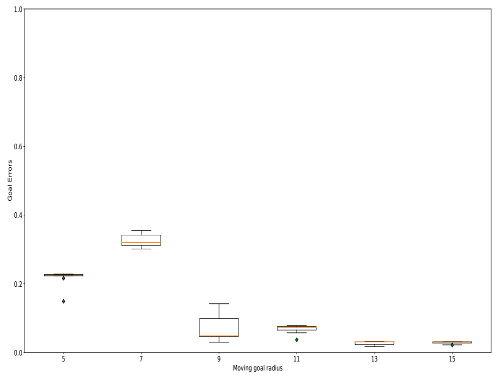


Figure 5.6: Illustration of the disturbance and real-time changing task situation schemes. The dashed blue line is the nominal demonstration. The blue line is the trajectory taken under disturbance. The dashed circle represents the path that the goal frame moves.

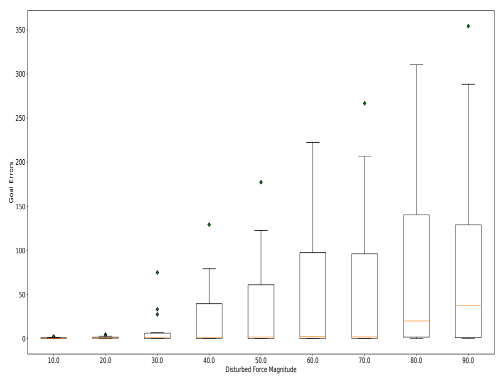


(a) 2D skills

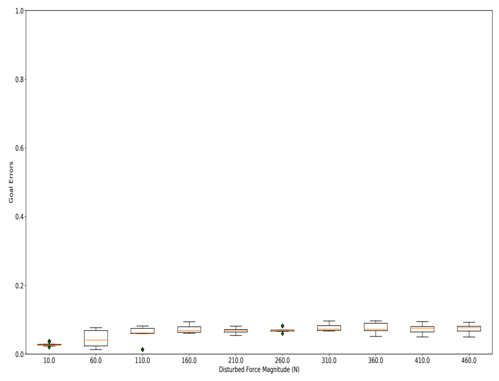


(b) picking skill

Figure 5.7: Graph of goal errors over number of Gaussian components K.

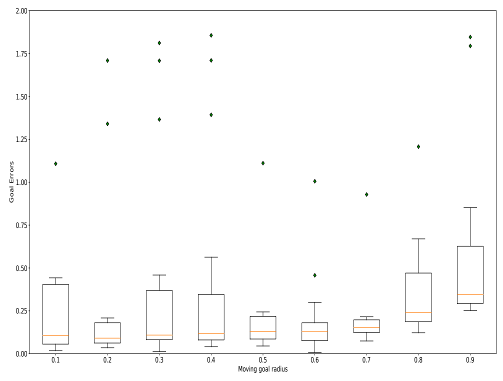


(a) 2D skills of $K = 9$

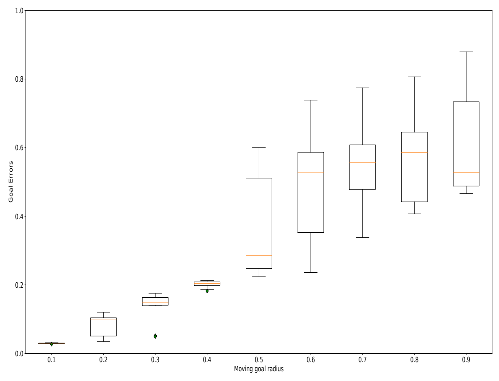


(b) picking skill of $K = 15$

Figure 5.8: Graph of goal errors over increasing disturbance force (N).



(a) 2D skills of $K = 9$



(b) picking skill of $K = 15$

Figure 5.9: Graph of goal errors over increasing moving goal radius (m).

high curvature variation, $K = 9$ seems to be the best model for this 2D dataset in the adaptability category, achieving the goal error median of nearly $0.1m$. However, for the case of picking skill in Figure 5.7b, low curvature skill is much more tolerated in model selection, with most of the large K achieving very low goal error. Figure 5.10 shows the reproduced trajectories of 2D models with $K = 9$ under the real-time adaptation scenario. Note that the wiggling shapes of trajectories reflect the models' reaction to the goal frame moving in a circle.

Figure 5.8 demonstrates the **Task adaptation with increasing disturbances** benchmark. We also evaluate 15 skills for 2D setting and 10 test runs of randomized angular velocity for picking skill over disturbance force values. We then select the best model $K = 9$ for 2D setting and $K = 15$ for picking skill to evaluate their robustness. This benchmark shows the TP-RMP robustness of goal convergence under the increasing disturbance force applied in a short duration of time at the beginning of the reproduction. The goal convergence of 2D TP-RMP models begins to worsen after applying $30N$ or more. The result is expected since the 2D dataset is designed to have a high variation of curvature. However, in Figure 5.8b, the learned TP-RMP model for low curvature variation picking skill is robust to much larger disturbance forces than 2D models. Figure 5.10 shows the reproduced trajectories of 2D models with $K = 9$ under the real-time adaptation with disturbance force of 10N. Note that the bottom left model fails to reach near the goal frame. These fail cases are the outliners plotted in benchmark graphs.

Figure 5.9 demonstrates the **Task adaptation with increasing differences** benchmark. We also evaluate 15 skills for 2D setting and 10 test runs of randomized angular velocity for picking skill over moving goal radius values. We also select the best model $K = 9$ for 2D setting and $K = 15$ for picking skill. The benchmark shows the TP-RMP robustness of goal convergence under the increasing differences comparing to the demonstrated situations. In this case, both 2D setting and picking skill have the same pattern of goal convergence performance worsen as moving goal radius increases because the TP-RMP adaptability is inherently dependent on the extrapolation capability of TP-GMM model by construction. Under larger difference of relative distance between the task parameters comparing to demonstrations, the computed GMM may lose the statistical coverage due to strong stretching (see Equation (3.6)), leading to worsening skill behavior. Note that this test benchmarks and tries to push the limit of the extrapolation capability of TP-RMP in real-time. It is not necessarily a limitation of TP-RMP since it operates outside of the demonstration region.

As seen from the data, 2D skills are designed as extreme scenarios where the shape of demonstration varies greatly, while in practice, actual manipulative tasks such as handover, insertion, pressing, etc. can have demonstrations with much lower curvature variations. These benchmarks imply that TP-RMP, in practice, can encode the skill behavior and adapt it to new situations in real-time. Furthermore, the skill can be robust to significant disturbances if the curvature variation of the skill demonstration is sufficiently low.

Regarding real-time constraint, Figure 5.12 shows the total time to execute a control loop in Algorithm 4.1. For each K model, we record and plot all the control iteration computation times in a reproduced trajectory. The computation time is low enough to adapt the skill in real-time to conduct this experiment in simulation. However, to ensure the strict real-time constraint in practice, which caps the control loop computation time under $\Delta t = \frac{1}{f}$, TP-RMP must be implemented in a highly optimized production code version in C++ or reducing control frequency f.

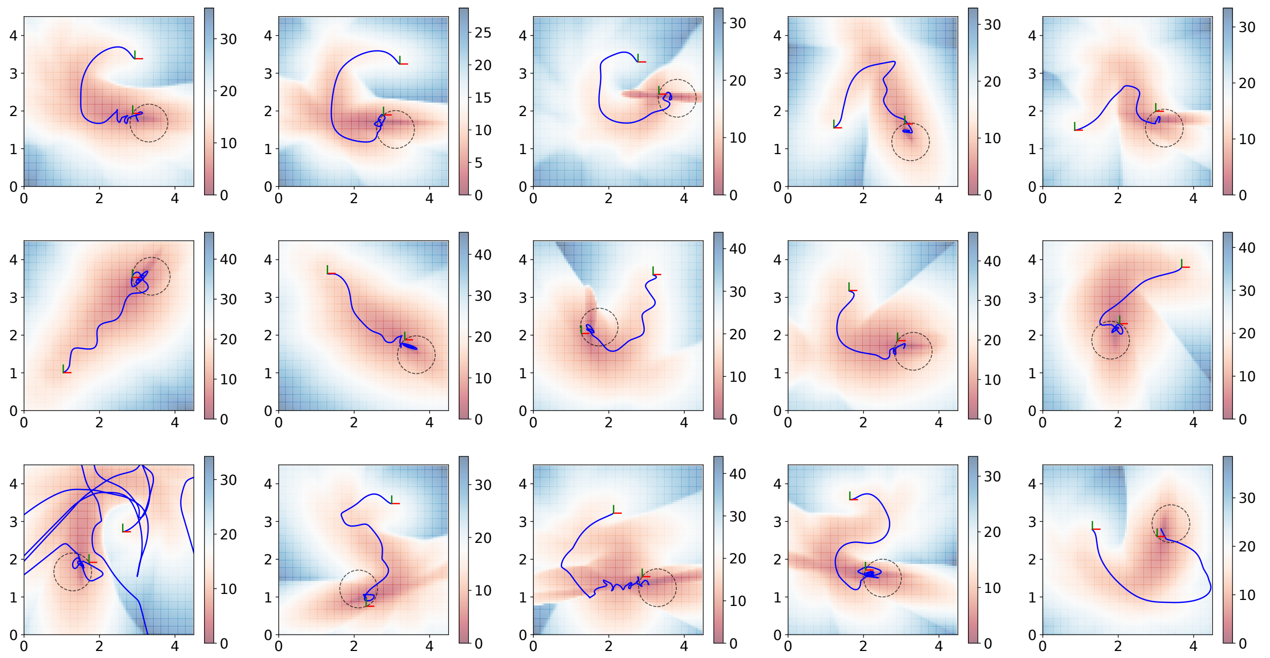


Figure 5.10: Reproduced trajectories of TP-RMPs models with $K = 9$ corresponding to 15 2D skills. In this case, we fix the *start* frame and move the *end* frame in circle of radius $r = 0.5m$ with angular velocity $\omega = \pi$ represented by the dashed circles.

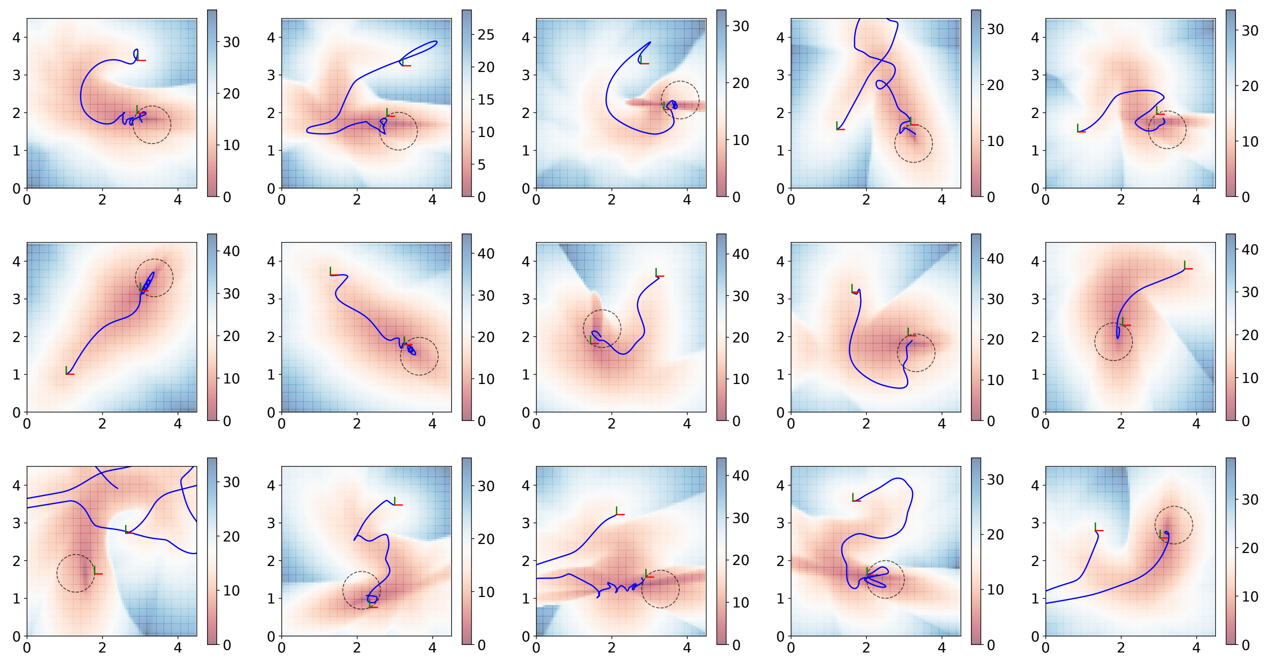


Figure 5.11: Reproduced trajectories of TP-RMPs models with $K = 9$ corresponding to 15 2D skills. We additionally apply a disturbance force of 10N for the duration $t \in [50, 100]$. Other conditions are the same as Figure 5.10.

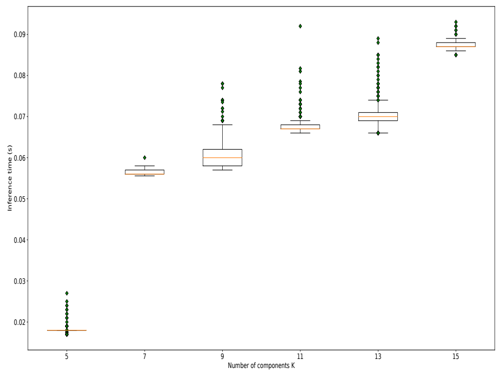


Figure 5.12: Total inference time of TP-RMP over number of components K. The inference time is the sum of new RMP computation time and configuration acceleration policy computation time at the root node in a control loop. Note that this time measurement is conducted on Python code and run on the CPU AMD Ryzen 7 5800X.

5.4 TP-RMP composability evaluation

Here, we benchmark the *composability* property of TP-RMP with other RMPs via RMPflow. In this evaluation setup, we keep the conditions of moving goal frame like the previous section. The system is expected to follow the learned behavior encoded in TP-RMP, while also avoiding the obstacle in the task space. We implement the collision avoidance RMP for a circle obstacle with radius R as in [CMI+19]. To realize the combined policy, for the robot arm, the RMP-tree is constructed in Figure 5.13. In 2D setting, the task space $\mathbb{R}^2$ is equivalence to the configuration space. The skill reproduction is executed using Algorithm 4.1.

Figure 5.14 shows the success rate of reproducing 15 2D skills while maintaining collision avoidance constraints. Successful reproduction is achieved when the system does not collide with the obstacle and converges in the goal vicinity $\epsilon \leq 0.1m$. Note that larger obstacle makes collision avoidance RMP overwhelmed the learned TP-RMP behavior, as the combined policy starts to fail the test at obstacle radius $R = 0.4m$. Some examples of the skill reproduction under moving goal radius $r = 0.5m$ are shown in Figure 5.15. The skill reproduces trajectories resembling the nominal demonstration while avoiding the obstacle and converging to the moving goal.

Figure 5.16 demonstrates the visual examples of real-time task adaptation of picking skill with and without the obstacle (i.e., white ball). As the obstacle also moves dynamically,

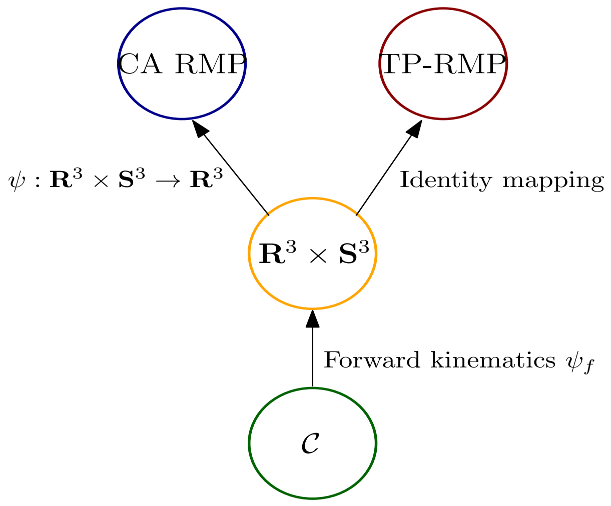

CA RMP
TP-RMP
$\psi : \mathbf{R}^3 \times \mathbf{S}^3 \rightarrow \mathbf{R}^3$
Identity mapping
$\mathbf{R}^3 \times \mathbf{S}^3$
Forward kinematics ψ_f
$\mathcal{C}$

Figure 5.13: RMP-tree for UR5 robot arm that combines learned TP-RMP skill and Collision Avoidance RMP (CA RMP).

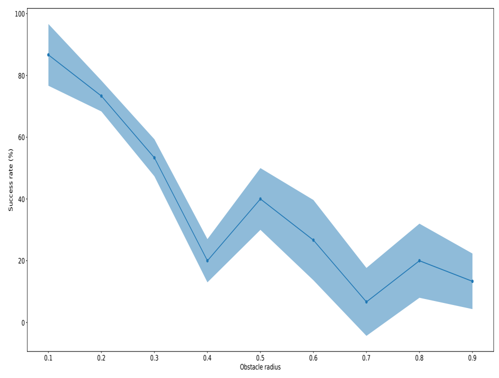


Figure 5.14: Success rate of goal convergence over increasing obstacle radius hindering skill execution. For each obstacle radius, we execute 15 2D skills ten times.

the collision avoidance RMP is also updated with the current obstacle position for each timestep.

These visual evaluations exhibit all three properties of *reactive*, *adaptability* and *composability* at the same time. As a remark, other RMPs can be view as weighted disturbance forces in the TP-RMP perspective. This view is significant because TP-RMP can be robust to external forces for goal convergence, as shown in the previous section. Hence, TP-RMP can solve the problem while compromising with other RMPs to satisfy additional task objectives, which renders the desired combined manipulative behavior.

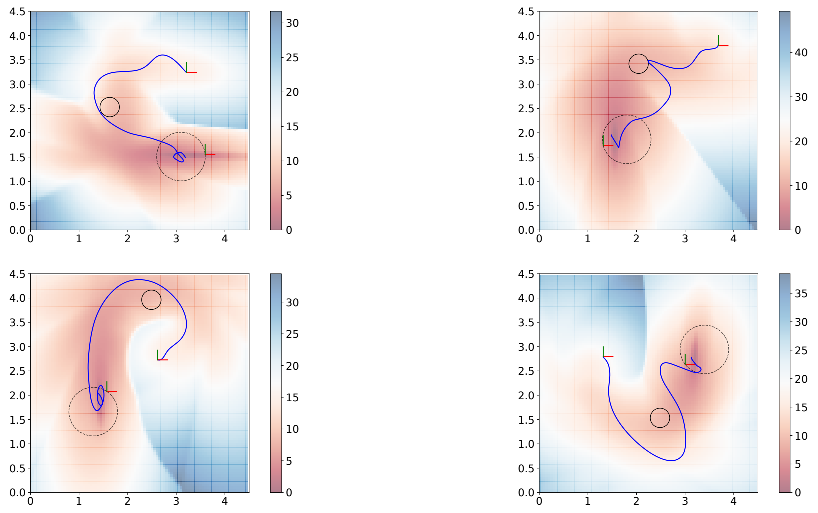


Figure 5.15: Reproduced trajectories of selected TP-RMPs models with $K = 9$. The black circle with radius $R = 0.2$ is the obstacle placed to hinder the skill execution.

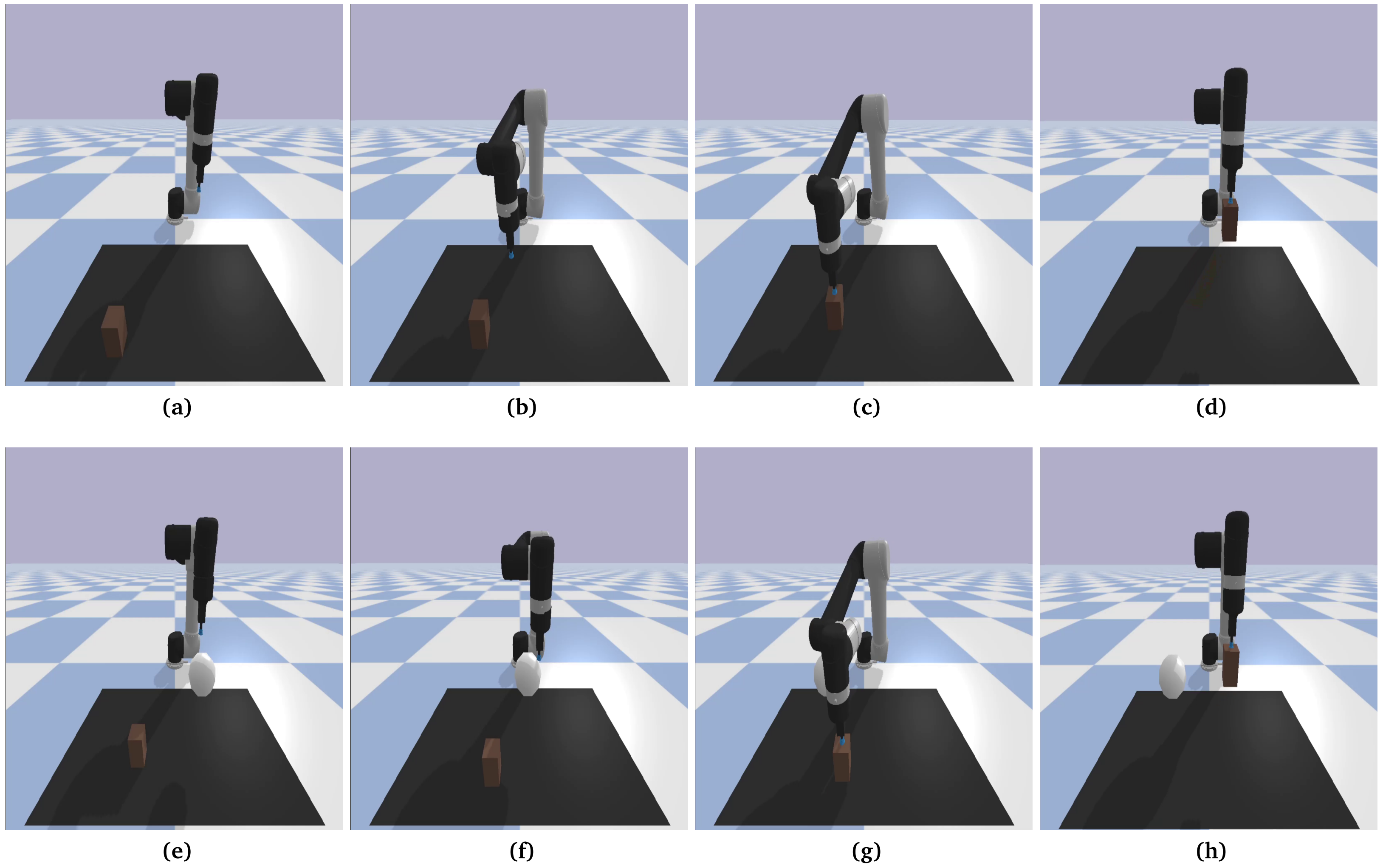


Figure 5.16: (a)-(d) Picking skill reproduction under moving object in circle $r = 0.15m$ and angular velocity $\omega = 0.3\pi$. **(e)-(h)** Picking skill reproduction in combination with collision avoidance RMP with the same moving object. The white ball with radius $R = 0.06$ is the obstacle.

6 Conclusion and Outlook

In this thesis, we presented a novel skill model TP-RMP that is robust to real-time changing task situations, external disturbances and is composable inside the RMPflow framework. By modeling with TP-GMM, TP-RMP creates a task adapting RMP, in which the potential field, dissipative field, and the Riemannian metric are also task conditioned. In addition, we provide concrete theorems to ensure the generated RMP is Lyapunov stable under any task parameters and initial system states.

We show a sanity test that the learned skill can track the nominal demonstration under the same demonstrated situation. We then provide a series of benchmarks using a 2D skill dataset and simulated UR5 robot arm to show the robustness of TP-RMP under increasing task difficulties in terms of external disturbance, situation differences, and combining with collision avoidance RMP. It is shown that TP-RMP achieves solid task solving capability in the case of practical tasks with low curvature variation demonstration trajectories while performs satisfactorily with more difficult hand-designed 2D skills. These experiment results exhibit the promised properties of TP-RMP, namely *reactive*, *adaptability* and *composable*.

Outlook

Our model also achieves low inference computational cost, which is suitable for real-time control. Hence, it would be interesting to implement this work on a real robot system to test its real-time adaptability while satisfying other task objectives. Furthermore, this work can be extended to Task and Motion Planning paradigms. A direct way to realize is to sequence and/or branch the TP-GMM structure into multiple paths [RGK+20]. The suitable tasks to evaluate are the handover task with two behaviors, such as grasp-top and grasp-side, and the pick-and-insert task with straight insertion (e.g., peg in the hole) and side insertion (e.g., AA battery).

Another consideration is that we can use the formulation of TP-RMP inside the Policy Search paradigm [DNP13]. It would be interesting that we can treat TP-RMP as a contextual policy representation for Policy Search methods. In Reinforcement Learning settings, this outlook is beneficial since the desired behavior can be represented by expert policies or implicit reward functions.

A Illustrations of the learned 2D skills

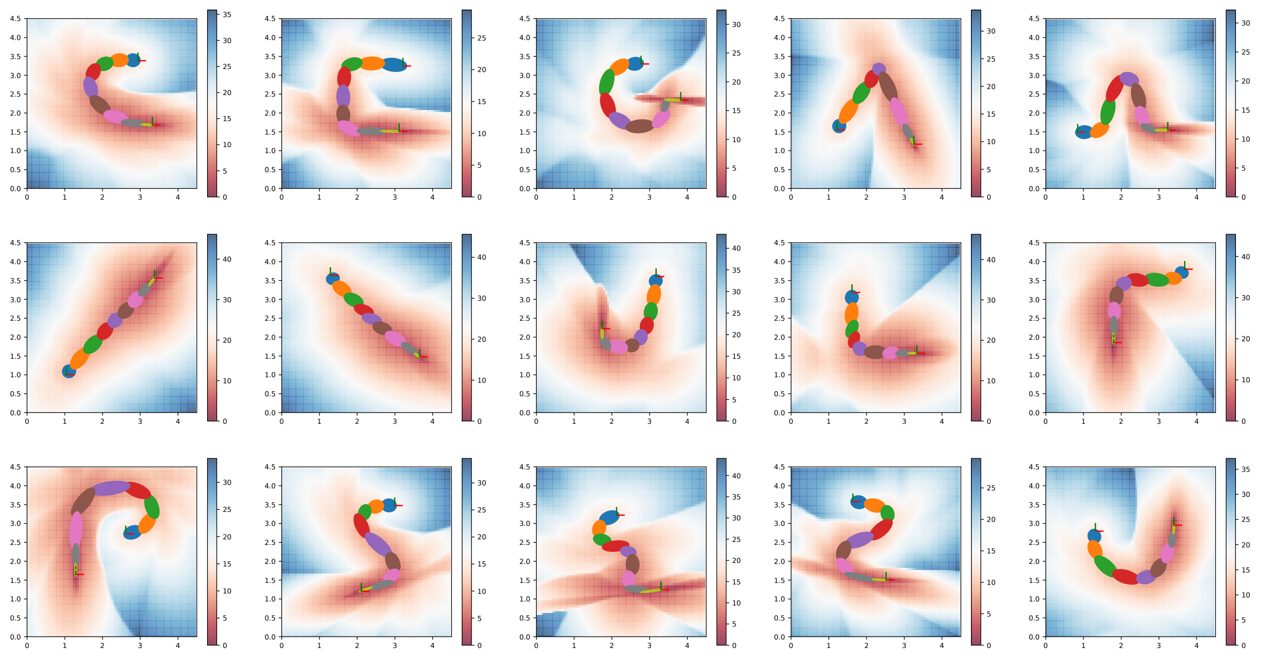


Figure A.1: The computed GMMs given the demonstration task parameters in Figure 5.1. The backgrounds are the induced potential fields of these GMMs. The ellipsoids represent the standard variation regions of the Gaussians.

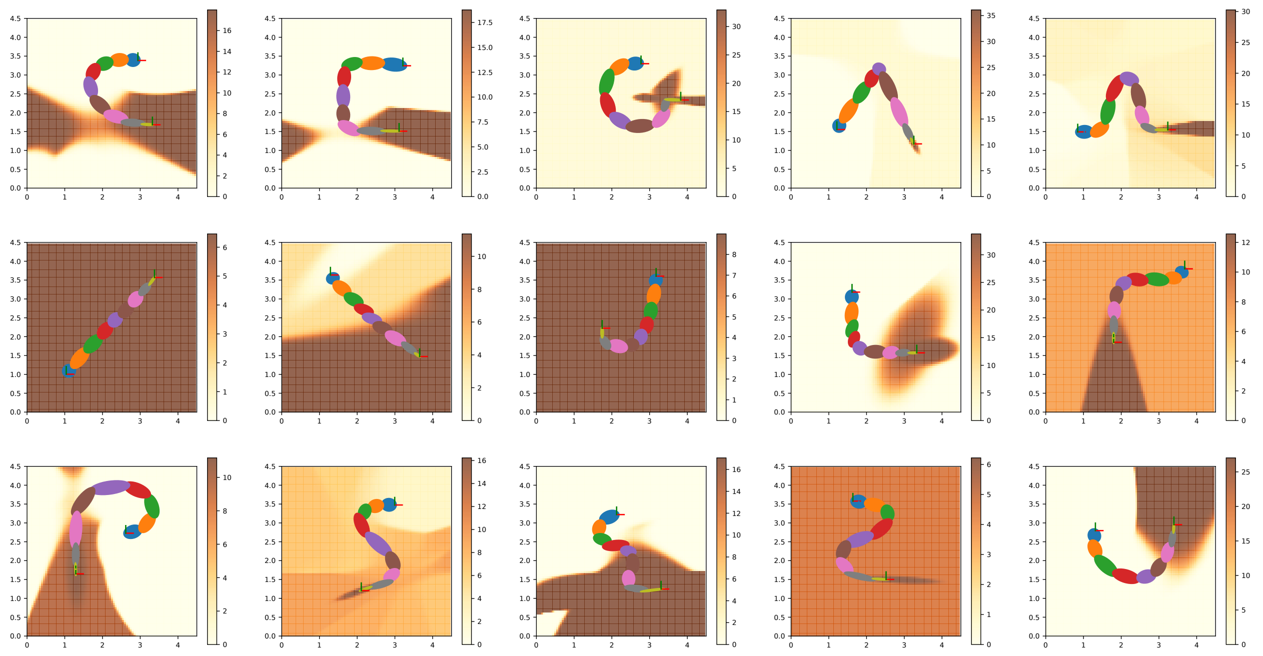


Figure A.2: The computed GMMs given the demonstration task parameters in Figure 5.1. The backgrounds are the induced dissipative fields of these GMMs. The ellipsoids represent the standard variation regions of the Gaussians.

Bibliography

[BCDS08] A. Billard, S. Calinon, R. Dillmann, S. Schaal. “Survey: Robot Programming by Demonstration.” In: *Springer Handbook of Robotics* (2008), pp. 1371–1394. DOI: 10.1007/978-3-540-30301-5_60. URL: http://infoscience.epfl.ch/record/114050 (cit. on pp. 15, 17).

[BL04] F. Bullo, A. D. Lewis. *Geometric Control of Mechanical Systems*. Vol. 49. Texts in Applied Mathematics. New York-Heidelberg-Berlin: Springer Verlag, 2004. ISBN: 0-387-22195-6 (cit. on pp. 20, 28, 30).

[Cal16] S. Calinon. “A Tutorial on Task-Parameterized Movement Learning and Retrieval.” In: 9.1 (2016). ISSN: 1861-2776. DOI: 10.1007/s11370-015-0187-9 (cit. on pp. 17, 23).

[CB21] E. Coumans, Y. Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2021 (cit. on p. 43).

[CMI+19] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, N. Ratliff. *RMPflow: A Computational Graph for Automatic Motion Policy Generation*. 2019. arXiv: 1811.07049 [cs.RO] (cit. on pp. 15, 18, 20, 22, 28, 39, 58).

[CSC10] S. Calinon, I. Sardellitti, D. G. Caldwell. “Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies.” In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 249–254. DOI: 10.1109/IROS.2010.5648931 (cit. on p. 17).

[DB16] S. Diamond, S. Boyd. “CVXPY: A Python-embedded modeling language for convex optimization.” In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5 (cit. on p. 34).

[DNP13] M. P. Deisenroth, G. Neumann, J. Peters. 2013 (cit. on p. 63).

[Hub64] P. J. Huber. “Robust Estimation of a Location Parameter.” In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. DOI: 10.1214/aoms/1177703732 (cit. on p. 32).

[INS02] A. Ijspeert, J. Nakanishi, S. Schaal. "Movement imitation with nonlinear dynamical systems in humanoid robots." In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, 1398–1403 vol.2. DOI: 10.1109/ROBOT.2002.1014739 (cit. on p. 17).

[KK17] S. M. Khansari-Zadeh, O. Khatib. "Learning Potential Functions from Human Demonstrations with Encapsulated Dynamic and Compliant Behaviors." In: 41.1 (2017), pp. 45–69. ISSN: 0929-5593. DOI: 10.1007/s10514-015-9528-y (cit. on pp. 18, 26, 32–34, 37).

[KTNS12] V. Krüger, V. Tikhanoff, L. Natale, G. Sandini. "Imitation learning of non-linear point-to-point robot motions using dirichlet processes." In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2029–2034. DOI: 10.1109/ICRA.2012.6224674 (cit. on p. 48).

[LaS60] J. LaSalle. "Some Extensions of Liapunov's Second Method." In: *IRE Transactions on Circuit Theory* 7.4 (1960), pp. 520–527. DOI: 10.1109/TCT.1960.1086720 (cit. on p. 39).

[LGD+21] A. T. Le, M. Guo, N. v. Duijkeren, L. Rozo, R. Krug, A. G. Kupcsik, M. Burger. "Learning Forceful Manipulation Skills from Multi-modal Human Demonstrations." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct. 2021) (cit. on pp. 17, 33).

[MKKP12] K. Muelling, J. Kober, O. Kroemer, J. Peters. "Learning to select and generalize striking movements in robot table tennis." In: *The International Journal of Robotics Research* 32 (2012), pp. 263–279 (cit. on p. 17).

[Mur12] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 0262018020 (cit. on p. 48).

[PL18] A. Pervez, D. Lee. "Learning task-parameterized dynamic movement primitives using mixture of GMMs." In: *Intelligent Service Robotics* 11 (2018), pp. 61–78 (cit. on p. 17).

[RGK+20] L. Rozo, M. Guo, A. G. Kupcsik, M. Todescato, P. Schillinger, M. Giftthaler, M. Ochs, M. Spies, N. Waniek, P. Kesper, et al. "Learning and Sequencing of Object-Centric Manipulation Skills for Industrial Tasks." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct. 2020). DOI: 10.1109/iros45743.2020.9341570. URL: http://dx.doi.org/10.1109/IROS45743.2020.9341570 (cit. on pp. 17, 33, 63).

[RIK+18] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, D. Fox. *Riemannian Motion Policies*. 2018. arXiv: 1801.02854 [cs.RO] (cit. on pp. 15, 17, 20).

[RLR+20] M. A. Rana, A. Li, H. Ravichandar, M. Mukadam, S. Chernova, D. Fox, B. Boots, N. Ratliff. “Learning Reactive Motion Policies in Multiple Task Spaces from Human Demonstrations.” In: *Proceedings of the Conference on Robot Learning*. Ed. by L. P. Kaelbling, D. Kragic, K. Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, Nov. 2020, pp. 1457–1468 (cit. on pp. 18, 29, 34, 40).

[RWX+21] N. D. Ratliff, K. V. Wyk, M. Xie, A. Li, M. A. Rana. *Optimization Fabrics for Behavioral Design*. 2021. arXiv: 2010.15676 [cs.RO] (cit. on p. 28).

[UGAM10] A. Ude, A. Gams, T. Asfour, J. Morimoto. “Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives.” In: *IEEE Transactions on Robotics* 26.5 (2010), pp. 800–815. DOI: 10.1109/TRO.2010.2065430 (cit. on p. 17).

[ULL+21] J. Urain, P. Liu, A. Li, C. D’Eramo, J. Peters. “Composable Energy Policies for Reactive Motion Generation and Reinforcement Learning.” In: *Robotics: Science and Systems XVII* (July 2021). DOI: 10.15607/rss.2021.xvii.052. URL: http://dx.doi.org/10.15607/RSS.2021.XVII.052 (cit. on p. 18).

[Zee18] M. Zeestraten. “Programming by demonstration on Riemannian manifolds.” PhD thesis. 2018 (cit. on p. 19).

All links were last followed on September 9, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Stuttgart 09.09.2021 [illegible]

place, date, signature