A Geometric Optimal Control Approach for Imitation and Generalization of Manipulation Skills

Boyang Ti[1,2], Amirreza Razmjoo[2], Yongsheng Gao[1], Jie Zhao[1] and Sylvain Calinon[2]

[1]*State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin 150001, China*
(email: 17b908043@stu.hit.edu.cn; gaoys@hit.edu.cn; jzhao@hit.edu.cn)

[2]*Idiap Research Institute, CH-1920 Martigny, Switzerland*
(email: amirreza.razmjoo@idiap.ch; sylvain.calinon@idiap.ch)

Abstract

Daily manipulation tasks are characterized by regular features associated with the task structure, which can be described by multiple geometric primitives related to actions and object shapes. Only using Cartesian coordinate systems cannot fully represent such geometric descriptors. In this article, we consider other candidate coordinate systems and propose a learning approach to extract the optimal representation of an observed movement/behavior from these coordinates. This is achieved by using an extension of Gaussian distributions on Riemannian manifolds, which is used to analyse a small set of user demonstrations statistically represented in different coordinate systems. We formulate the skill generalization as a general optimal control problem based on the (iterative) linear quadratic regulator ((i)LQR), where the Gaussian distribution in the proper coordinate systems is used to define the cost function. We apply our approach to object grasping and box-opening tasks in simulation and on a 7-axis Franka Emika robot using open-loop and feedback control. The results show that the robot can exploit several geometries to execute the manipulation task and generalize it to new situations. The results show high variation along the *do-not-matter* direction, while maintaining the invariant characteristics of the task in the coordinate system(s) of interest. We then tested the approach in a human-robot shared control task. Results show that the robot can modify its grasping strategy based on the geometry of the object that the user decides to grasp.

Keywords: Learning from Demonstration, Riemannian Geometry, Model-based Optimization, Optimal Control

1. Introduction

Skillful manipulation does not just mean how precisely a person can perform a task, but also how well one can cope with complex and changing scenarios and exploit the system redundancy to counteract perturbations. This fact arises from a variety of research areas, including biomechanics, neuroscience, sport science, control, and robotics, with related formulations including minimal intervention principle [1], uncontrolled manifold [2], and optimal feedback control [3]. In the uncontrolled manifold, for example, we assume that the human would not be very stiff in the directions that do not matter for the task, so we expect to see wide variations in the motion of those directions. The central nervous system will not focus on the *do-not-matter* directions. Instead, its control effort would be delegated among the variables crucial for the task (low variability) [4].

Synergies in the system can be defined by considering a more general definition of variation, namely the correlation. Synergistic systems can keep the system functionality not merely by controlling only one variable but also all the linked elements. Studies such as [5] show that this capability is the main principle that helps biological systems in nature deal with complex situations.

We can use different coordinate systems to describe the task. Sternad et al. [4] have shown that the choice of the coordinate system plays an important role in (co)variation modeling. Each coordinate system can be seen as a set of different features. Some features (coordinate systems) can exploit the structure of the task by forgiving the errors in different ways, so they are more advantageous over others. This advantage can be seen from both computational and geometrical aspects, see [6] for an overview. Moreover, studies in human motion planning [7, 8] suggest that humans

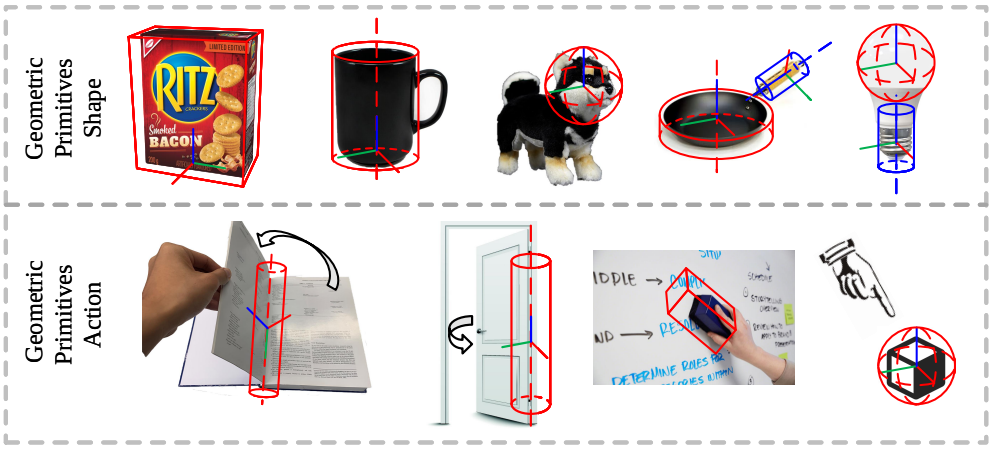


Figure 1: Geometric primitives in daily objects and tasks, with examples involving prismatic, cylindrical and spherical shapes.

can model the task in different or even "mixed" coordinate frames, such as body reference frame (the putamen) and gaze- or head-centered frame (parietal cortex), which may be well addressed by the concept of multiple candidate geometries formulated as an optimal control problem.

In this article, we introduce a motion planning approach that can benefit from different coordinate systems and present an approach to extract the most relevant one by statistical analysis. We use a small set of demonstrations and a set of Riemannian manifolds to estimate Gaussian distributions for each coordinate system. The skill generalization problem is formulated as a general optimal control problem (OCP). The cost function of the OCP is defined using the Gaussian distributions constructed in the optimal coordinate system with the reference reproduced by Gaussian mixture regression (GMR). The processing pipeline of the approach is shown in Fig. 2.

Most graspable shapes in our human-made environment, either the whole object or a local part of it, can be described by three main types of coordinate systems: Cartesian, cylindrical and spherical coordinate systems as shown in Fig. 1. In addition to object shapes, some tasks can be represented more efficiently in a specific coordinate system. As shown in the second row of Fig. 1, rotating an object (e.g., opening/closing a door, turning a page of a book) can be described in a cylindrical system, while for wiping a table and pointing to an object, it is better suited to use a prismatic and a spherical system, respectively. Defining the task in the proper manifold allows the task to be represented with more relevant geometric features, enabling the robot to extract and learn the skill more easily.

In our previous work [9], we used open-loop control in an optimal control framework to reproduce the task. However, the (i)LQR method can also provide optimal feedback gains, which can be used to cope with external disturbances. The values of the feedback gains determine how the system would react to the disturbances, which is highly related to the task. Learning task-dependent feedback gains is discussed in LfD by considering the data variations. We show in this article how modeling these variations in different types of coordinate systems can improve the system's autonomous behavior. Modeling the data in multiple types of coordinate systems reveals the more subtle correlation between the states of the system and provides a more systematic way to exploit the variations.

Feedback gains are also necessary for collaborative tasks to keep humans safe (by not applying excessive force) and provide more intuitive interactions. However, it is not enough. Feedback gains allow the system to react to spatial disturbances, but they do not consider temporal ones. When the robot interacts with a predictable environment, it may not be a big issue. However, a human is highly unpredictable and introduces many sources of spatiotemporal disturbances to the system. It would be more favorable to have a time-independent controller, usually called *policy* in the literature. To gain this goal, we implement the phase estimation technique by decoupling the time variable of the system, i.e., phase, from the real-world clock, and calculating the phase as a function of the robot states. This improvement allows us to get a time-independent policy from the iLQR method without changing anything on the optimization part.

We also represented invariant features of observed manipulation in the chosen coordinate system from Cartesian, Cylindrical with z axis and Spherical coordinates in [9]. The system was defined at the level of the robot kinematic, where a linear system reproduced the time-driven movement in an optimal control framework with a non-linear cost

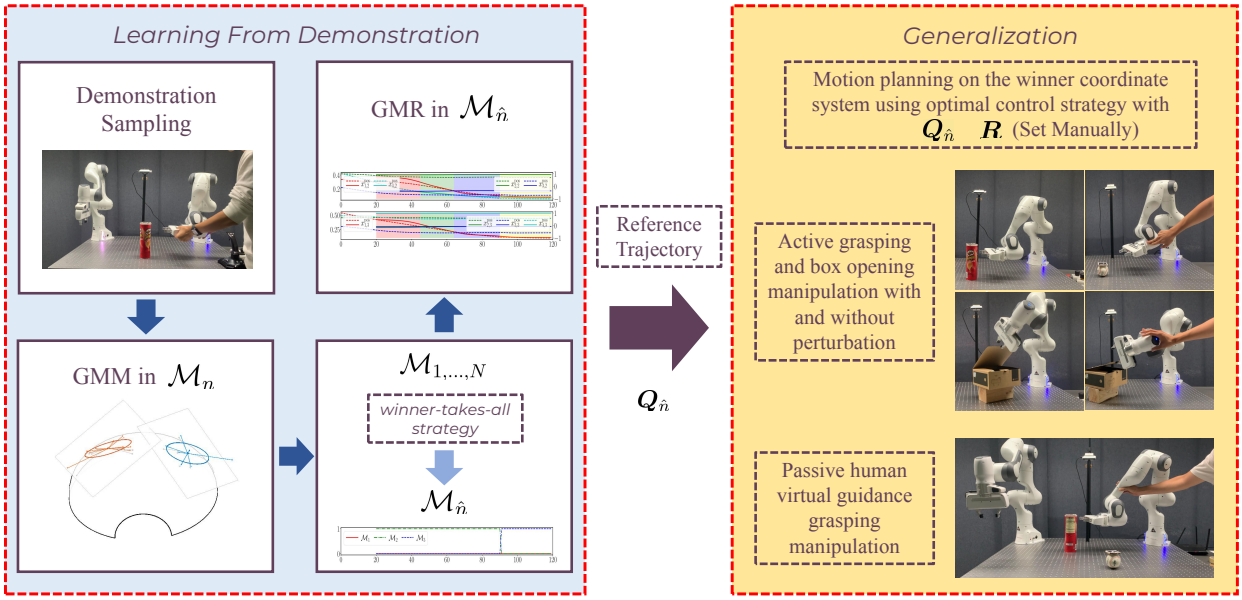


Figure 2: Processing pipeline of the approach. The left block is the learning part, where the Gaussian distribution is constructed in the Riemannian manifold based on human demonstrations (Grasping and Box-opening). By exploiting a *winner-takes-all* strategy and GMR, a reference trajectory is generated in the chosen coordinate system with the precision matrix information passed to the generalization block. The right block shows the generalization approach using an optimal control strategy for autonomous manipulation (with and without perturbation), and for a collaborative grasping task with human guidance.

function. Beyond the combination of our preliminary contributions on imitation of manipulation skills using multiple geometries, the contributions of this research are the following: (1) we propose an approach to improve the generalization capability of manipulation skills by fully considering different types of coordinate systems including Cartesian, Cylindrical with three distinct main axes, and Spherical coordinates; (2) we introduce a motion planning approach using OCP defined in different coordinate systems both at the level of the task and at the level of the robot kinematic structure; (3) we introduce a probabilistic approach to define the variations that are allowed in a task, which introduces the compliance along *do-not-matter* dimension of the movement; (4) we validate the application of our method for manipulation tasks in autonomous and shared-control modes.

The notations are summarized in Table 1. In the remainder of the article, we summarize the related works in Sec. 2, and give an overview of the background in Sec. 3. The method is explained in Sec. 3.3. Sec. 5 includes the experiments and we discuss the results in Sec. 6. We conclude the article in Sec. 7 by summarizing the contribution, limitations and future works.

2. Related Work

2.1. Application of Geometry Information

The geometry of objects has been considered in robotic manipulation for different reasons. One of its famous applications is force control [10]. The object or the task action constraint can have a complicated shape. However, we can often reconstruct each shape approximately using simple geometric primitives [11], or by combining it with point clouds [12, 13]. The manipulation features can be represented efficiently using three priority levels of safety, primary and auxiliary constraints extracted from geometric primitives [14]. The difference from our approach is that we consider multiple types of coordinate systems, which can represent curved geometry features better than the strategy that uses multiple Cartesian coordinate systems.

Different from the most of research [15, 16] on skill learning, which encoded the system only in a Cartesian coordinate system ignoring the task geometry information. The introduction of geometry can fill this gap in this field [17, 18]. C. Pérez-D'Arpino et al. [19] have used computer-aided designs (CAD) and the Euclidean group SE(3)

Table 1: List of the notation in this article.

Notation	Explanation
$\boldsymbol{A}$	state matrix
$\boldsymbol{B}$	input matrix
$\boldsymbol{K}$	feedback gain
$\boldsymbol{N}$	number of coordinate systems
$\boldsymbol{O}$	basis origin of coordinate systems
$\boldsymbol{Q}_{\hat{n}}$	precision matrix of selected coordinate system
$\boldsymbol{R}$	control weight matrix
$\mathbb{R}^d$	Euclidean space
$\boldsymbol{S}_{\boldsymbol{x}/\boldsymbol{u}}$	augmented state/input transformation matrices
$\mathcal{S}^d$	sphere manifold
$\mathcal{M}_{1/2/3}$	manifold of Cartesian/Cylindrical/Spherical coordinate system
$\mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$	tangent space of the point x_0 on manifold
$c(\hat{c})$	cost function(in the selected coordinate system)
$\hat{n}$	index of selected coordinate system
$\boldsymbol{x}_n$	state vector/in the n-th coordinate system
$\boldsymbol{x}_{d,n,t}^{\text{pos/ori}}$	d-th dimension of position/orientation represented in the n-th manifold at timestep t
$\boldsymbol{u}$	control command vector
$\text{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$	exponential map
$\text{Log}_{\boldsymbol{x}}(\boldsymbol{y})$	logarithmic map

to reason about the geometric constraints. These constraints, including axial rotation, fixed points, etc., can also be extracted from a list of kinematic constraints [20]. The demonstrations are also utilized to define soft constraints in cost functions. Vochten et al. [21] introduced the trajectory shape descriptors to represent the demonstration in a coordinate-free way, which eliminates the dependency on the coordinate reference of the demonstration during movement generalization. Calinon et al. [22] formulate an OCP with a quadratic cost whose precision matrix was proportional to the inverse of (co)variation of demonstration data expressed in multiple Cartesian coordinate systems (from the perspective of different objects or landmarks). This approach allows the robot to benefit from the variation observed in the data to determine the importance of the different Cartesian coordinate systems and provide an adaptive compliance behavior for the manipulation task based on this information.

2.2. Share Control

Shared control has a wide variety of applications, ranging from robot-assisted human surgery [23] to navigation control [24, 25]. Virtual fixtures is a technique to improve the efficiency of human-robot co-manipulation by constraining human motion in task-relevant trajectories [26]. These virtual fixtures can be defined efficiently using demonstration. Raiola et al. [27] introduced a framework of multiple probabilistic virtual guides learned from demonstrations. This framework allows the human to "escape" from the original constraint to a new task. Bodenstedt et al. [28] presented a semi-autonomous strategy, where the human control tool translation and robot is responsible for its orientation. To obtain more functions on virtual guidance, Nemec et al. [29] introduced the integration of Frenet–Serret frame into SS-DMP [30], where it can embed the compliance according to the learned variance and speed of motion. The commonality of the above research is that motion is planned in the Cartesian coordinate system, which limits its extension application in complex geometric operations such as polishing curved surfaces or grasping objects, where different objects may require different strategies to control the orientation of the robot end-effector while approaching the object. Our approach is advantageous for this purpose by providing different virtual guides according to the shape of the object.

2.3. Curvilinear Representation

Another direction in this topic is using a curvilinear coordinate system, which can be simpler to use than the Cartesian coordinate system for applications considering various geometries. Currently, the curvilinear coordinate system has been exploited in plane motion planning like autonomous driving [31, 32]. Ju et al.[33] developed the curvy axis Gaussian model in two-dimensional Cartesian space, which needs enough data points to fit the Gaussian. In the manipulation case, Zhang et al. [34] introduced an Adapted Curvilinear Gaussian Mixture Model to encode the curve distribution which bends the principal axis of Gaussian into a nonlinear shape. It happens in $\mathbb{R}^d$ space, which can roughly fit the curves, but it will lose the geometry information in the case of a curve with a large curvature.

2.4. Optimal Control

The optimal control approach can be used to generalize the motion to new task parameters by formulating the cost as a function of these parameters and searching for an optimal solution over joint/task spaces. Task parameters can include different viapoints at different time steps that the robot end-effector should reach (for position and/or orientation). Different approaches have been proposed to solve optimal control problems, see [35] for an overview. Here, we focus on the linear quadratic regulator (LQR) and on the iterative LQR (iLQR) extension [36, 37, 38]. These methods leverage Gauss–Newton optimization and dynamic programming to provide controllers either as open-loop commands or as a feedback controller with varying gains. M. M. Hassan et al. [39] have presented a general formulation and numerical scheme for the fractional optimal control problem of distributed systems in spherical and cylindrical coordinates. However, this approach has no relation to motion planning and divides the spherical into axial and complete symmetry to discuss. Kobilarov et al. [40] introduced the variational discretization of the dynamics to generate optimal trajectories on a given Lie group by providing its Lagrangian, group structure, and description of acting forces. Here, we will derive how the LQR/iLQR is used in motion planning problems in the cylindrical and spherical coordinate systems using Riemannian manifolds.

Methods discussed in reinforcement learning (RL) field mostly focus on finding a controller in the form of a policy $\boldsymbol{u} = f(\boldsymbol{x})$, with state $\boldsymbol{x}$ and control commands $\boldsymbol{u}$ that are only state-dependent. In optimal control and trajectory optimization problems, the formulation is simplified by finding a controller $\boldsymbol{u} = f(\boldsymbol{x}, t)$ that is dependent on time variable. This simplified version is often more practical when targeting high-dimensional real robot applications. In this article, we aopt an intermediary strategy by describing the evolution of the system in a path-dependent but time-independent fashion. To do this, a phase variable s is used as an auxiliary term to describe the evolution of the system as a function of the robot current state $s(\boldsymbol{x})$, yielding a time-independent controller $\boldsymbol{u} = f(\boldsymbol{x}, s(\boldsymbol{x}))$. The use of a phase variable has previously been motivated in physical human-robot interaction [41] and bimanual force-motion control [42].

Standard LQR/iLQR usually utilizes only one coordinate system for all time steps, which can be located either at the end-effector or the base of the robot, depending on the functionality. Previously, we have shown the advantage of using multiple coordinate systems located at different landmarks such as objects [43]. In that work, we defined the LQR problem in the task space and used an additional inverse kinematic to find the robot joint trajectory. Our work is an extension of that approach in which the coordinate systems are not only attached to different objects or landmarks but also consider different geometries. In this article, we use LQR with an impedance controller to counteract the perturbation and then extend this to iLQR for the open-loop control of non-linear systems. We also verify the approach in a shared human-robot interaction, where a user drags the robot to different objects, where the robot automatically selects an appropriate grasping strategy based on the shape characteristics of the target, and helps the user achieve the manipulation task (shared control with adaptive virtual fixtures).

Defining the target as a point limits the capability of manipulation skills. Sometimes we prefer the system to reach an area, such as any point around a circle instead of a single point. Specifying these shapes as a cost function in the Cartesian coordinate system with only one Gaussian distribution is not possible. We can use a mixture of Gaussians, but we typically need multiple demonstrations to estimate this mixture model and the modeling as a circle remains approximate. The approach we propose in this article can model these primitive geometric shapes more efficiently and more systematically by describing the observed demonstrations in other types of coordinate systems. Our method allows the system to automatically extract these shape primitives and use them inside the cost functions defined for the LQR/iLQR optimization problem. This approach is similar to the grasping notion mentioned in [44], which often describes an object using prismatic, cylindrical, and spherical descriptors. To learn the data distribution in different

manifolds, we need to consider probabilistic distributions that take into account the specificity of these manifolds. This is achieved by using an approach to estimate Gaussian distributions on Riemannian manifolds [45].

3. Background

3.1. (iterative)Linear Quadratic Regulator ((i)LQR) control

In an OCP, a cost function is minimized with respect to control commands over a time window, subject to a function describing the system evolution by starting from an initial state. The general discrete form of OCP consists of a cost

$$c(\boldsymbol{x}, \boldsymbol{u}) = \sum_{t=1}^{T-1} c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) + c_T(\boldsymbol{x}_T, \boldsymbol{u}_T), \tag{1}$$

subject to the dynamics

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{2}$$

where $\boldsymbol{x}$ and $\boldsymbol{u}$, where $\boldsymbol{x} = \left(\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top \cdots, \boldsymbol{x}_T^\top\right)^\top$ and $\boldsymbol{u} = \left(\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \cdots, \boldsymbol{u}_{T-1}^\top\right)^\top$.

An LQR problem is a subclass of OCPs, where the cost is quadratic and the dynamics is linear, namely

$$c(\boldsymbol{x}, \boldsymbol{u}) = \sum_{t=1}^{T} \|\boldsymbol{x}_t\|^2_{\boldsymbol{Q}_t} + \|\boldsymbol{u}_t\|^2_{\boldsymbol{R}_t}, \tag{3}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t. \tag{4}$$

This problem has an analytical solution, which can be derived either as a batch formulation or as a recursive one. The latter, which is also called dynamic programming (for linear systems) and differential dynamic programming (for non-linear system) [46], can provide feedback gains as well. We have describe the batch version in Appendix A and have shown how it can be modified to non-quadratic cost and/or non-linear dynamics. In Sec. 4.2, we will modify it to be used on manifolds. Without any loss of generality, the same modifications can be applied to the recursive version.

3.2. Riemannian manifold

A smooth d-dimensional manifold $\mathcal{M}$ is a topological space that locally behaves like the Euclidean space $\mathbb{R}^d$. Most common manifolds in robotics are homogeneous, providing simple analytic expressions for exponential/logarithmic mapping and parallel transport. A *Riemannian manifold* is a smooth and differentiable manifold equipped with a positive definite metric tensor. There are many examples of Riemannian manifolds that can be employed in robot manipulation [45]. In this article, we just introduce the sphere manifold $\mathcal{S}^d$ characterized by constant positive curvature. The $\mathcal{S}^d$ manifold can be used in robotics to encode directions or orientations. Unit quaternion $\mathcal{S}^3$ is used to represent end-effector (tooltip) orientation [47], and $\mathcal{S}^2$ is used to represent unit directional vector perpendicular to surfaces (e.g., for contact planning). Articulatory joints can be represented on the torus $\mathcal{S}^1 \times \mathcal{S}^1 \times \cdots \times \mathcal{S}^1$ [48, 49]. For the multiple types of coordinate systems, including Cartesian, Cylindrical, and Spherical coordinate systems, we can also use the sphere manifold to represent the position in different coordinates (Cartesian $\rightarrow \mathbb{R}^3$, Cylindrical $\rightarrow \mathcal{S}^1 \times \mathbb{R}^2$ and Spherical $\rightarrow \mathcal{S}^2 \times \mathbb{R}^1$). We use $\mathcal{M}_1$ for Cartesian, $\mathcal{M}_2$ for Cylindrical and $\mathcal{M}_3$ for Spherical manifolds in the remainder of the article to simplify the notation.

For each point $\boldsymbol{p} \in \mathcal{M}$ on the manifold, there exists a tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ that locally linearizes the manifold. The curve with the minimum length between two points on a Riemannian manifold is called geodesic. Similar to straight lines in Euclidean space, the second derivative is zero everywhere along a geodesic. The exponential map $\text{Exp}_{\boldsymbol{x}_0} : \mathcal{T}_{\boldsymbol{x}_0}\mathcal{M} \rightarrow \mathcal{M}$ maps a point $\boldsymbol{u}$ in the tangent space of $\boldsymbol{x}_0$ to a point $\boldsymbol{x}$ on the manifold, so that $\boldsymbol{x}$ lies on the geodesic starting at $\boldsymbol{x}_0$ in the direction $\boldsymbol{u}$. The norm of $\boldsymbol{u}$ is equal to the geodesic distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}$. The inverse map is called the logarithmic map $\text{Log}_{\boldsymbol{x}_0} : \mathcal{M} \rightarrow \mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$. The exponential and logarithm maps for $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}^d$ can be computed analytically as (see also [50])

$$\boldsymbol{y} = \text{Exp}^{\mathcal{S}^d}_{\boldsymbol{x}}(\boldsymbol{u}) = \boldsymbol{x}\cos(\|\boldsymbol{u}\|) + \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\sin(\|\boldsymbol{u}\|), \tag{5}$$

$$\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}^{\mathcal{S}^d}(\boldsymbol{y}) = \arccos(\boldsymbol{x}^\top \boldsymbol{y}) \frac{\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y}\boldsymbol{x}}{\|\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y}\boldsymbol{x}\|}. \tag{6}$$

There is no exponential map in the Cartesian space, and the logarithmic map for the Cartesian space $\mathcal{M} = \mathbb{R}^d$ of d dimensions in general is

$$\mathrm{Log}_{\boldsymbol{x}}^{\mathbb{R}^d}(\boldsymbol{y}) = \boldsymbol{y} - \boldsymbol{x}. \tag{7}$$

Note that there are two ways to express the point belonging to the $\mathcal{S}^d$, one is using the angle between each axis to locate the position and the other way is to use the unit vector to report its projection on different axes. For example in $\mathcal{S}^1$ ($\mathcal{S}^2$), we can use θ ($\{\theta, \phi\}$) to express the angle between a vector and the positive side of x ($\{x, z\}$) direction or we can use the projection of the vector in the x and y directions, such as $\mathcal{I} = \{I^x, I^y\}$ (the projection of the vector in the x, y and z directions $\mathcal{I} = \{I^x, I^y, I^z\}$), where the norm of $\mathcal{I}$ is 1. In our article, we choose the vector way to express the point on the manifold.

3.3. Gaussian Distribution on a Riemannian manifold

We construct the Gaussian distribution on the manifold by using an iterative strategy to estimate the mean of the Gaussian as a centroid on the manifold and use the covariance represented in its tangent space to express the variation of the data [51, 52, 47]. The distribution is denoted by

$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left((2\pi)^d |\boldsymbol{\Sigma}|\right)^{-\frac{1}{2}} \exp(-\frac{1}{2} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})\, \boldsymbol{\Sigma}^{-1}\, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})), \tag{8}$$

where a point on the manifold is denoted as $\boldsymbol{x} \in \mathcal{M}$, the mean of the distribution (origin of the tangent space) is denoted as $\boldsymbol{\mu} \in \mathcal{M}$, and the covariance matrix represented in its tangent space is $\boldsymbol{\Sigma} \in \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$.

The mean of a set of N data points on the manifold is obtained by an optimization problem

$$\min_{\boldsymbol{\mu}} \sum_{n=1}^{N} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)^\top\, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n), \tag{9}$$

whose solution can be found using an iterative approach, such as Gauss–Newton, as

$$\boldsymbol{u} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n), \qquad \boldsymbol{\mu} \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}}(\boldsymbol{u}), \tag{10}$$

which should be repeated until convergence. With the obtained mean, the covariance matrix in its tangent space can be expressed as $\boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)\ \mathrm{Log}_{\boldsymbol{\mu}}^\top(\boldsymbol{x}_n)$, see [45] for details.

4. Proposed Method

4.1. Manifold selection

We assume that we have gathered a set of data such as $\mathcal{D} = \{\boldsymbol{d}^1, \boldsymbol{d}^2, \ldots, \boldsymbol{d}^D\}$, where D is the total number of demonstrations. The choice of the manifold to express the demonstrations affects the learning and control procedure. So the first step is to find the most relevant manifold to represent the task. We assume that this manifold is time-dependent, and use the same manifold for all the time steps in a stage. In the following, we describe the procedure of selecting a manifold at time t, and in the experiment part (Sec. 5), we will explain how this time can be selected. So, we select the corresponding data from the demonstrations such as

$$\boldsymbol{d}_t = \{\boldsymbol{d}_t^1, \boldsymbol{d}_t^2, \ldots, \boldsymbol{d}_t^D\},$$

where $\boldsymbol{d}_t^i$ is the t-th element from the i-th demonstration. Using the method described in Sec. 3.3, we construct Gaussian distributions for each $\boldsymbol{d}_t$ on different candidate manifolds $n \in \{1, \ldots, N\}$, where N is the number of coordinate

systems. We then choose the coordinate system $\hat{n}$ that shows the most regularities between the different demonstrations (*winner-takes-all* strategy), namely

$$\hat{n}_t = \arg\min_n |\boldsymbol{\Sigma}_{n,t}|, \quad \forall n \in \{1, \dots, N\}, \tag{11}$$

where $|\boldsymbol{\Sigma}_{n,t}|$ is the determinant of the covariance matrix $\boldsymbol{\Sigma}_{n,t}$. For 2-D systems, $N = 2$ ($\mathcal{M}_1, \mathcal{M}_2$), and for 3-D systems, $N = 5$ ($\mathcal{M}_1, \mathcal{M}_{2-x,y,z}, \mathcal{M}_3$), where $\mathcal{M}_{2-x}$ is an $\mathcal{M}_2$ manifold whose main axis is in the x direction (see Fig. 3(a)).

4.2. OCP in different types of coordinate systems

Both (1) and (2) are defined in Euclidean space in an standard OCP. Either the cost function or the dynamic equation can be modified to use the notion of manifold in this formulation. We can define state residual terms in (1) at time t on the desired manifold $\hat{n}$ as

$$\boldsymbol{e}^x_{\hat{n},t} = \mathrm{Log}^{\mathcal{M}_{\hat{n},t}}_{\boldsymbol{\mu}_{\hat{n},t}}(\boldsymbol{x}_{\hat{n},t}), \tag{12}$$

where $\boldsymbol{x}_{\hat{n},t}$ is a vector composed of elements $x_{d,\hat{n},t}$, d represents each dimension of the manifold, and $\boldsymbol{\mu}_{n,t}$ is the desired value of $\boldsymbol{x}_{\hat{n},t}$ learned from demonstrations. The cost function for reaching task on Euclidean space, for example, can be defined with quadratic residual error $c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = \|\boldsymbol{x}_t - \boldsymbol{\mu}_t\|^2_{\boldsymbol{Q}_t} + \|u\|^2_{\boldsymbol{R}_t}$, where $\boldsymbol{\mu}_t$ is the tracking point at time t, $\boldsymbol{Q}_t$ is the precision matrix at time t, and $\boldsymbol{R}_t$ is the effort weight matrix. The modified version of this on other manifolds can be described as

$$\hat{c}_t = \|\boldsymbol{e}^x_{\hat{n},t}\|_{\boldsymbol{Q}_{\hat{n},t}} + \|u_t\|_{\boldsymbol{R}_t}, \tag{13}$$

where, $\boldsymbol{e}^x_{\hat{n},t}$ corresponds to the state error defined on the manifold $\hat{n}$ at time t. $\boldsymbol{Q}_{\hat{n},t}$ is the precision matrix of the data at time t calculated on the $\hat{n}$ manifold, which can be defined from the demonstrations with different approaches. In Sec. 5, we will describe two methods to determine this matrix inspired by the minimal intervention method in [22]. This problem can be solved with the iLQR method described in Appendix A.

Other than the cost function, the dynamic equation of the system can be expressed on the manifold. This idea makes sense only when the state variables are defined in the desired manifold, unless for other cases (e.g., states are defined as the robot's joint angles), we keep the dynamic equation the same as before and only modify the cost function. However, the robot end-effector can be modeled as a point mass system on different manifolds. The system can be represented on $\mathcal{M}_2$ and $\mathcal{M}_3$ in similar ways as on $\mathcal{M}_1$ by projecting the state from the manifold to the tangent space defined on the current state or target state. The evolution of the system then can be described linearly as

$$\hat{\boldsymbol{u}}_{\hat{n},t} = \mathrm{Log}^{\mathcal{M}_{\hat{n},t}}_{\boldsymbol{x}_{\hat{n},t-1}} \boldsymbol{x}_{\hat{n},t}, \tag{14}$$

$$\hat{\boldsymbol{x}}_{\hat{n},t+1} = \hat{\boldsymbol{x}}_{\hat{n},t} + \hat{\boldsymbol{u}}_{\hat{n},t} dt, \tag{15}$$

$$\boldsymbol{x}_{\hat{n},t+1} = \mathrm{Exp}^{\mathcal{M}_{\hat{n},t}}_{\boldsymbol{x}_{\hat{n},t}} \hat{\boldsymbol{x}}_{\hat{n},t+1}, \tag{16}$$

where $\boldsymbol{x}_{\hat{n},t}$ and $\hat{\boldsymbol{x}}_{\hat{n},t}$ represent the state on the manifold and the tangent space of the local origin, respectively ($\boldsymbol{x}_{2,t} = \{r_t, I^x_t, I^y_t, z_t\}$, $\boldsymbol{x}_{3,t} = \{r_t, I^x_t, I^y_t, I^z_t\}$, r_t represents the radius of circle and sphere, z_t represents the height of cylinder). The control $\hat{\boldsymbol{u}}_{n,t}$, precision matrix $\boldsymbol{Q}$ and control weight $\boldsymbol{R}$ are defined on the tangent space. Due to the involvement of the transformations (14)–(16), it is hard to construct the batch version of the state function, where there is a coupling relationship between each state. Therefore, we select the recursive version and update the state using the control command as

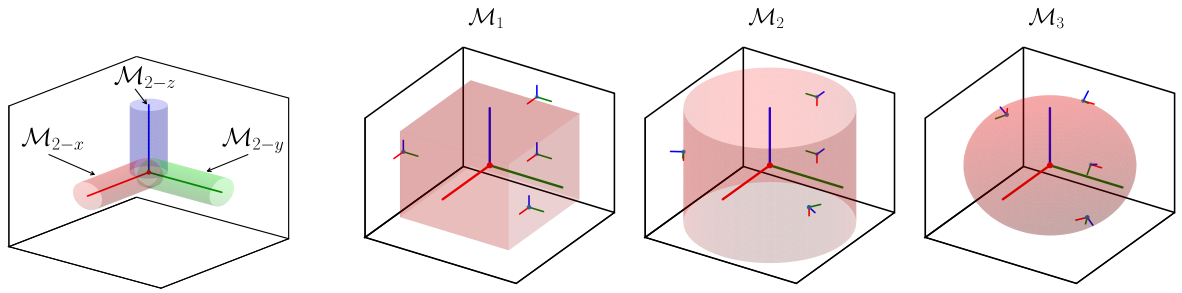
$$\hat{\boldsymbol{u}}_{n,t} = -\boldsymbol{K}_{n,t} \hat{\boldsymbol{x}}_{n,t}, \tag{17}$$

where $\boldsymbol{K}_{n,t}$ is calculated using a Riccati equation.

In our experiments, the full pose in these manifolds consists of the position and orientation part like $\boldsymbol{x}_{\hat{n},t} = [\boldsymbol{x}^{\mathrm{pos}\,\top}_{\hat{n},t}, \boldsymbol{x}^{\mathrm{ori}\,\top}_{\hat{n},t}]^\top$ and $\boldsymbol{\mu}_{\hat{n},t} = [\boldsymbol{\mu}^{\mathrm{pos}\,\top}_{\hat{n},t}, \boldsymbol{\mu}^{\mathrm{ori}\,\top}_{\hat{n},t}]^\top$. We list the notation of manifold represented in 2D and 3D space in Table 2, where $\mathcal{S}^d$ and $\mathbb{R}^d$ represent sphere and Cartesian manifolds of dimension d, respectively. ($\mathcal{S}^1 \times \mathbb{R}^1$ consists of polar angle and radius; $\mathcal{S}^1 \times \mathbb{R}^2$ consists of polar angle, radius and height; $\mathcal{S}^2 \times \mathbb{R}^1$ consists of polar angle, azimuth angle and radius.)

Table 2: The manifold of different coordinate systems in 2D and 3D spaces.

	2D Space ($N = 2$)		3D Space ($N = 5$)		
	Cartesian	Polar	Cartesian	Cylindrical	Spherical
Position	$\mathcal{M}_1^{\text{pos}} \in \mathbb{R}^2$	$\mathcal{M}_2^{\text{pos}} \in \mathcal{S}^1 \times \mathbb{R}^1$	$\mathcal{M}_1^{\text{pos}} \in \mathbb{R}^3$	$\mathcal{M}_{2-x,y,z}^{\text{pos}} \in \mathcal{S}^1 \times \mathbb{R}^2$	$\mathcal{M}_3^{\text{pos}} \in \mathcal{S}^2 \times \mathbb{R}^1$
Orientation	$\mathcal{M}_1^{\text{ori}} \in \mathcal{S}^1$	$\mathcal{M}_2^{\text{ori}} \in \mathcal{S}^1$	$\mathcal{M}_1^{\text{ori}} \in \mathcal{S}^3$	$\mathcal{M}_2^{\text{ori}} \in \mathcal{S}^3$	$\mathcal{M}_3^{\text{ori}} \in \mathcal{S}^3$



(a) Different manifolds illustration

(b) Base frames of different manifolds

Figure 3: (a) shows the manifold considered in this article including $\mathcal{M}_1$ (i.e., $\mathbb{R}^3$), $\mathcal{M}_{2-x,y,z}$ (i.e., $\mathbb{R}^2 \times \mathcal{S}^1$) and $\mathcal{M}_3$ (i.e., $\mathbb{R}^1 \times \mathcal{S}^2$). (b) shows the base frames defined in different manifolds.

The Cartesian product property of Riemannian manifolds ($\mathcal{M}_{\hat{n}} = \mathcal{M}_{\hat{n}}^{\text{pos}} \times \mathcal{M}_{\hat{n}}^{\text{ori}}$) can simplify the distance calculation including position and orientation of the cost function, where it follows the composition rule

$$\text{Log}_{\boldsymbol{\mu}}^{\mathcal{M}_a \times \mathcal{M}_b}(\boldsymbol{x}) = \begin{bmatrix} \text{Log}_{\boldsymbol{\mu}_a}^{\mathcal{M}_a}(\boldsymbol{x}_a) \\ \text{Log}_{\boldsymbol{\mu}_b}^{\mathcal{M}_b}(\boldsymbol{x}_b) \end{bmatrix}. \tag{18}$$

We use a local frame to express the orientation of the end-effector, where the local base frame is constructed according to its position on the manifold by a parallel transport of the basis $\boldsymbol{O}$ defined at the origin of the manifold, see Fig. 3.

4.3. Phase Estimation

The optimal control method described above is time-dependent. Although it may be beneficial for some tasks, it is not suitable for all scenarios. For example, the robot will not consider temporal disturbances, which may have undesirable consequences. In the shared control collaboration between the robot and the human, the robot's motion is semi-autonomous controlled by itself as well as guided by humans. We consider a shared control collaboration between the robot and the human in this article. The robot will try to grasp some objects in its workspace, while the human will guide the robot toward different objects. Having time dependent controller may cause some difficulties such as the tracking point can move far away while the human keeps the robot, or the time may reach the end, and the robot may stop while the human is still holding the robot. We use a phase estimation method to make the controller time-independent, without having to consider a full policy estimation problem.

We decouple the time variable of the system, a.k.a. phase, from the real-world clock. Using an additional phase variable is not new in robotics, and it has been investigated in other works, such as dynamical movement primitives (DMP) [53], where the phase variable is defined as a function of the world clock. Here, we define it as a function of the robot's current state, and the desired trajectory learned from demonstrations $\boldsymbol{x}_d$. This trajectory is a time-series of variables which starts at $t = 0$ and finishes at $t = T$. Without any loss of generality, it can be assumed to start at phase $s = 0$ and end at $s = 1$. We can define this trajectory as a function of the phase as $\boldsymbol{x}_d(s)$. The phase estimation then can be mathematically formulated as

$$s^* = \underset{s}{\text{argmin}}\ \text{Log}_{\boldsymbol{x}_t^r}^{\mathcal{M}}(\boldsymbol{x}_d(s)), \tag{19}$$

where $\boldsymbol{x}_t^r$ is the robot's current (full or partial) state. This variable does not need to be the full state as some states do not matter in the task (*do-not-matter* directions).

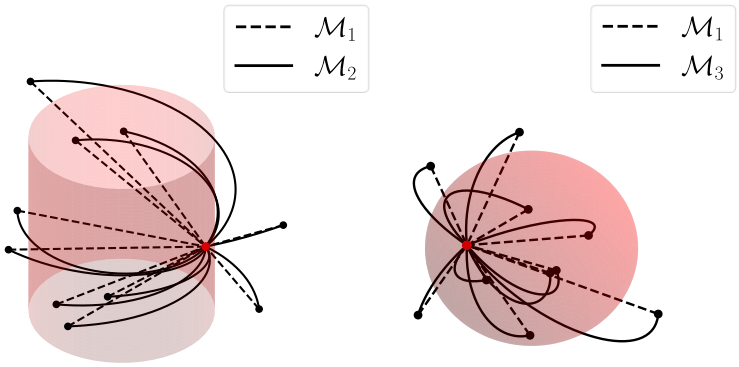


Figure 4: Point reaching trajectory in $\mathcal{M}_1$ (i.e., $\mathbb{R}^3$), $\mathcal{M}_2$ (i.e., $\mathbb{R}^2 \times \mathcal{S}^1$) and $\mathcal{M}_3$ (i.e., $\mathbb{R}^1 \times \mathcal{S}^2$).

There are different approaches to finding the phase, such as [41]. We found that the nearest neighbor method can perform well for the tasks we have in this article. At each time, we update the system time with s^*. If the user holds the robot, s^* will remain constant as $\boldsymbol{x}_t^r$ will stay the same. So, the robot will not try to increase the force or stop the task, as it can be the cases for the time-dependent controller mentioned above.

5. Experiments

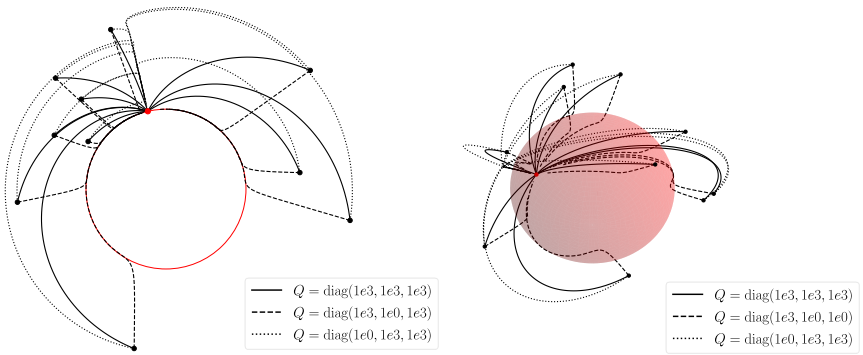
We first analyze the controller proposed in different coordinate systems by introducing a point-mass reaching task. For the whole pipeline, we took grasping and box-opening as two typical manipulation tasks in our daily life to validate our approach in the presence and absence of perturbation, both in a simulation and on a real robot. As a last experiment, we verify our method on a shared control human-robot collaboration using a virtual guidance system and policy extraction utilizing the phase variable. We used a 3-DoF planar robot for the simulation and a 7-DoF Franka Emika manipulator for the real robot experiment.

5.1. LQR reaching task in different coordinate systems

This experiment evaluates the effect of defining the optimal control problem on different manifolds. The system has to reach a target starting from ten random points in the space. We compare the results in multiple coordinate systems with equal precision Q and control weight R in each dimension. The results are shown in Fig. 4. We can see how the choice of the manifold for the optimization problem can help the system respect the object geometry. We observe that the point-mass system approaches the target along a sphere manifold with a uniformly decreasing radius in $\mathcal{M}_2$ and $\mathcal{M}_3$, while the resulting trajectories in $\mathcal{M}_1$ are straight lines. Fig. 5 shows how adjusting the precision weights for each dimension can affect the resulting trajectories. Although the motion of the point-mass system will reach the dimension with higher precision primarily, it will respect the shape of the manifold.

5.2. Grasping simulation

This experiment is a simulated planar grasping task. We validate our approach with open-loop (velocity controller) and feedback control (impedance controller), showing that the grasping movements can maintain the demonstration characteristics in the presence and absence of perturbations. We generated six sets of demonstrations manually and each demonstration consists of three viapoints representing three stages of the motion. We then construct Gaussian distributions for each stage of the task and the extracted distribution is used to define the stepwise reference with the precision matrix in $\mathcal{M}_1$ and $\mathcal{M}_2$, as shown in Fig. 6. In the left plot of Fig. 6(a), we observe that the means of positions for different stages of the task represented in $\mathcal{M}_1$ almost overlap, with significant covariances losing the feature of the circular distribution of the data points. Similarly, in the left plot of Fig. 6(b), there are large variations in orientations when expressed in $\mathcal{M}_1$. In contrast, for the representation in $\mathcal{M}_2$, as shown in the middle plot of Fig. 6(a), the distribution of positions has a very low variance along $\boldsymbol{x}_{1,2}^{\text{pos}}$ while being well separated, reflecting the decrease of radius while approaching the object. The high variance along $\boldsymbol{x}_{2,2}^{\text{pos}}$ means that the robot can grasp the object from any direction along the geometric manifold of the chips can. Similarly, in the right plot of Fig. 6(b),



(a) (b)

Figure 5: Point reaching trajectories in (a) $\mathcal{M}_2$ (i.e., $\mathbb{R}^2 \times \mathcal{S}^1$) (view along the axis) and (b) $\mathcal{M}_3$ (i.e., $\mathbb{R}^1 \times \mathcal{S}^2$) with different precision matrices.

Table 3: Determinants of covariance matrices at different stages of the motion for the grasping and box-opening simulations.

	Grasping / Box-opening		
	P_1	P_2	P_3
$\mathcal{M}_1$	3.2e1 / 4.3e-6	2.2e0 / 3.0e-3	3.5e-4 / 6.8e-6
$\mathcal{M}_2$	**3.9e-5 / 1.1e-6**	**2.0e-5 / 8.2e-8**	**1.2e-6 / 1.9e-6**

low variance in orientation reflects that the gripper kept orienting toward the object while approaching it, similarly to the demonstration. We use the determinants of the covariance matrices in the two coordinate systems to select a coordinate system at each stage of the motion, as shown in Table 3. $\mathcal{M}_2$ is here correctly selected as best manifold to express the grasping motion.

To reproduce similar motions with the robot, we set four random initial robot states and substitute the Gaussian distributions for each stage of the task into the cost function of the iLQR framework. Here, the robot should approach the object while pointing to it. Fig. 7 compares the results of this experiment on two manifolds and for two strategies. We observe in Fig. 7(a) that the reproduced motion becomes irregular with $\mathcal{M}_1$. In contrast, on $\mathcal{M}_2$ (the selected manifold), the robot generates the desired behavior (pointing toward the object while approaching it), correctly exploiting the variations in the *do-not-matter* directions.

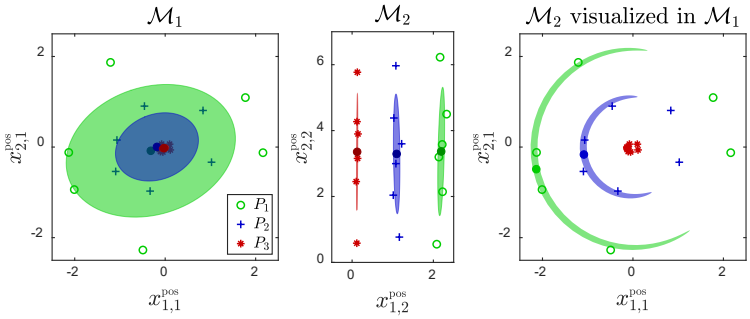
To further demonstrate the advantage of our approach, we introduce the feedback controller with variance learned from the demonstration to endow the robot with compliance behavior, with a rejection of external perturbations leveraging the optimal coordinate system selected by the algorithm. From Fig. 7(b), we observe that the motion (solid black line) regenerated from the perturbation (cyan dots) correctly keeps a pointing direction toward the target while approaching the object rather than returning to the initially planned trajectory (dashed black line).

5.3. Box-opening simulation

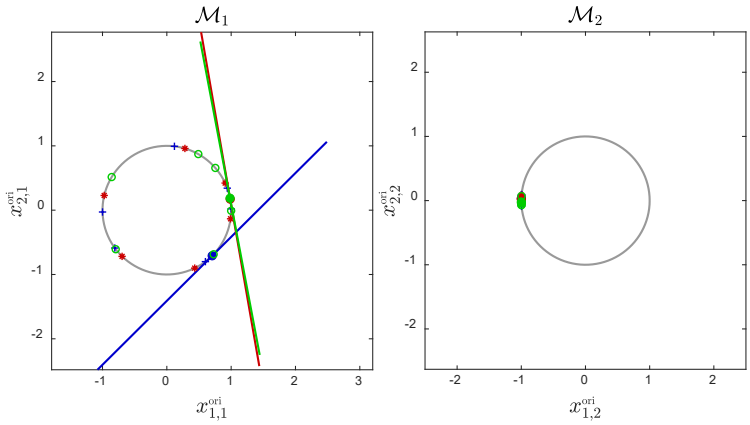
We want to teach the robot to open a box: a task that should be efficiently described in the $\mathcal{M}_2$ manifold. We generated one demonstration of the box opening movement. To simplify the process, we segmented the demonstration into these different stages manually and use the data at each stage to construct Gaussian distributions as stepwise references with its precision matrix (via-points) in $\mathcal{M}_1$ and $\mathcal{M}_2$. Fig. 8(a) shows the distribution represented in $\mathcal{M}_1$ and $\mathcal{M}_2$ and the distribution of $\mathcal{M}_2$ visualized in $\mathcal{M}_1$. The latter clearly shows that it can cover the circular spread of the data points. $\mathcal{M}_2$ is selected by the algorithm as the most representative manifold, which keeps the radius constant during the opening stage (small variance of $x_{1,2}^{\text{pos}}$). The variation values for the viapoints on the two manifolds are represented in Table 3. The motions reproduced by the robot using $\mathcal{M}_2$ are shown in the bottom-left plot of Fig. 8(a).

5.4. Autonomous grasping with the Franka Emika robot

We then implemented the grasping experiment on the real robot. We used six objects with different shapes and placed them randomly within the robot's workspace. We tracked their location using Aruco markers [54]. The task



(a) Positions of the end-effector w.r.t. the object to grasp



(b) Orientations of the end-effector w.r.t. the object to grasp

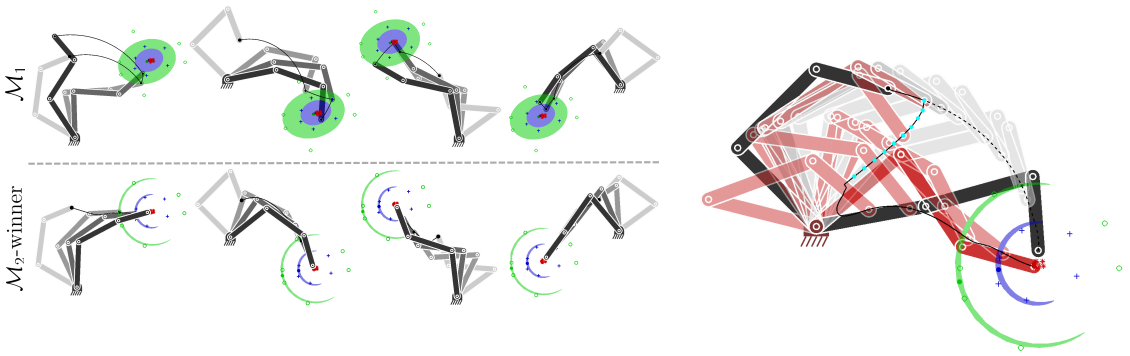
Figure 6: Distributions of the three sets of stepwise reference ($\circ, +, \star$) in $\mathcal{M}_1$ and $\mathcal{M}_2$. The indices in the graphs correspond to $x_{d,n}$. (a) The position is represented in two coordinate systems. The ellipses represent the Gaussian distributions for the three different stages of the motion (contours of one standard deviation). The contours of the Gaussian in $\mathcal{M}_2$ is also visualized in $\mathcal{M}_1$ to show that this distribution is a better fit (compare the left and right graphs). (b) The orientation is represented in the base frame of the two coordinate systems. The solid lines represent the variance in the tangent space of the $\mathcal{S}_1$ manifold, and the points represent the mean on the $\mathcal{S}_1$ manifold.

is deemed successful if the robot can reach the object from a home position, grasp it firmly, and return to the home position. We used kinesthetic teaching to gather six demonstrations for each object. Each demonstration consists of spatio-temporal data for the robot end-effector presented in the object frame, see Fig. 9. We use a GMM on the spatio-temporal data on $\mathcal{M}_1$ manifold to divide the task into four stages (number of stages chosen manually). We call this distribution a staging GMM to distinguish it from the data distribution (distribution on the manifold) represented on different manifolds. We then used GMR with the staging GMM to reproduce a set of references for each manifold $\mathcal{M}_1$, $\mathcal{M}_{2-x}$, $\mathcal{M}_{2-y}$, $\mathcal{M}_{2-z}$ and $\mathcal{M}_3$. The mean values of the Gaussians in the staging GMM also contain the time variable. We also used the variation modeled with data distribution on each desired manifold to construct the cost function of the OCP. The precision matrix $\boldsymbol{Q}_{\hat{n},t}$ at each time step is obtained by superposition of the precision matrices evaluated at each desired manifold as

$$\begin{aligned} \boldsymbol{Q}_{\hat{n},t} &= \frac{1}{z_t} \sum_{i=1}^{i=r} w_{i,t} \boldsymbol{Q}_{\hat{n}_i}, \\ w_{i,t} &= (t - \mu_t^i)^\top \Sigma_t (t - \mu_t^i), \quad z_t = \sum_{i=1}^{i=r} w_{i,t}, \end{aligned} \tag{20}$$

where μ_t^i and Σ_t are the mean value and the covariance matrix at time step t, obtained from the i-*th* Gaussian in the staging GMM, respectively, and $\boldsymbol{Q}_{\hat{n}_i}$ is the precision matrix obtained on the desired manifold of the i-*th* stage.

Determinants of the covariance matrices decide the type of coordinate systems at each stage. These values are presented in Fig. 10, whose apparent differences can be seen according to the object shapes. Our approach selects



(a) Open-loop control (b) Feedback control

Figure 7: (a) The grasping movement generated by open-loop control is simulated in $\mathcal{M}_1$ and $\mathcal{M}_2$ (selected coordinate system) from four different initial states of the robot generated randomly. The movements of the robot are depicted with different shades of gray, with the lightest depicting the initial state of the robot. (b) The grasping movement is generated by feedback control, where the perturbation happens for a short duration as labeled by the cyan points. The movements with and without perturbation are depicted with red and grey shades. The initial and final states are depicted in non-transparent.

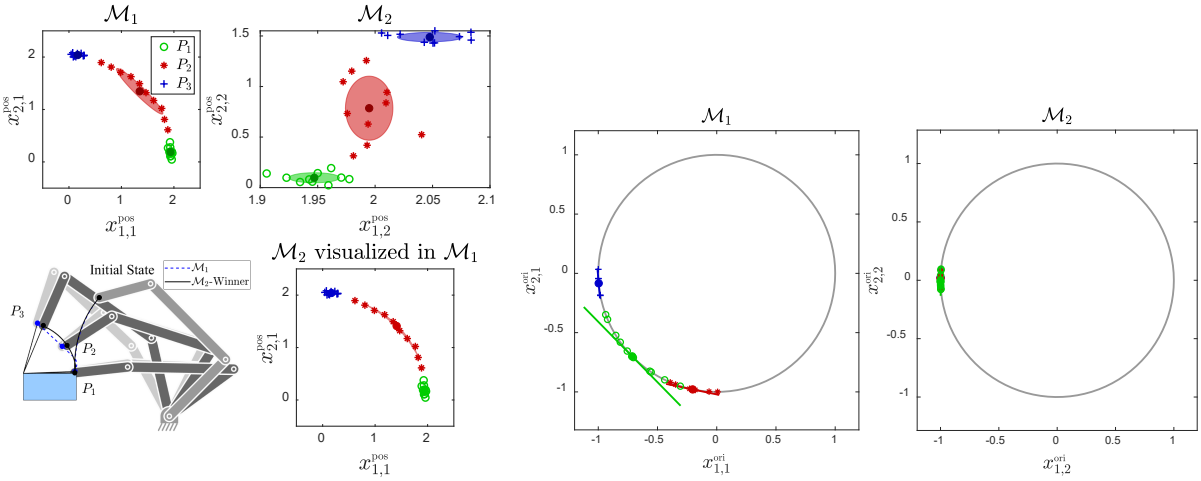
Table 4: Success rates for grasping objects with different shapes.

	$\mathcal{M}_1$	$\mathcal{M}_{2-x}$	$\mathcal{M}_{2-y}$	$\mathcal{M}_{2-z}$	$\mathcal{M}_3$	Optimal $\mathcal{M}_{\hat{n}}$
Chips Can	0/50	0/50	0/50	**45/50**	30/50	**42/50** ($\mathcal{M}_{2-z}$)
Baseball	2/50	0/50	0/50	**41/50**	**45/50**	**44/50** ($\mathcal{M}_{2-z}$+$\mathcal{M}_3$)
Bowl	0/50	0/50	0/50	**42/50**	2/50	**45/50** ($\mathcal{M}_{2-z}$)
Rubik's Cube	**47/50**	0/50	28/50	34/50	39/50	**46/50** ($\mathcal{M}_1$)
Cracker Box	**45/50**	0/50	23/50	**43/50**	26/50	**47/50** ($\mathcal{M}_1$)
Pitcher	**45/50**	38/50	0/50	**40/50**	33/50	**46/50** ($\mathcal{M}_1$)

$\mathcal{M}_{2-z}$ and $\mathcal{M}_1$ to represent the motion in the case of cylindrical shapes (chips can and bowl) and prismatic objects (the Rubik's cube, the cracker box, and pitcher handle), respectively. It can also be seen in Fig. 11(d) that the selected manifold can change during the motion, as for the baseball object. We observe that during the first stages of the motion (P_1, P_2, and P_3), although $\mathcal{M}_{2-z}$ is selected as the proper manifold, the determinants of $\mathcal{M}_{2-z}$ and $\mathcal{M}_3$ are close to each other. The choice of the manifold changes to $\mathcal{M}_3$ for the last stage P_4. One reason for this switching can be described by external or environmental geometric factors, such as the task shape, robot gripper, and robot geometry. The robot's motion in the first stage is influenced by other constraints, such as avoiding the table or other external obstacles or preferring some directions over others because of the robot's kinematic shape and configuration. These constraints can influence the motion to favor a cylindrical manifold over a spherical one. We can also observe that the reference (dashed line) is not precisely tracked by the robot (solid line), which means that the robot generalizes the motion to new situations guaranteeing key features of the task while reducing the amount of effort, thus ignoring the tracking of less essential features.

We generated 50 random poses for each of the object from the distribution (modeled with a Gaussian) observed in the demonstrations to validate the generalization ability of our approach to new poses in the simulation. The objects are always located at the origin, as the demonstrations are gathered in the object frame.

The proposed method is compared to the benchmark solution of using only one of the manifolds. The reproduced motions are shown in Fig. 12, where we showcase the typical baseball grasping results. The object orientation makes the motion on $\mathcal{M}_1$ to be undesirable, as the robot tries to grasp the object from only a specific orientation. The performance of the system on $\mathcal{M}_{2-z}$ and $\mathcal{M}_3$ is close to each other. However, the system performs slightly better on $\mathcal{M}_{2-z}$ in the first stages of the task and on $\mathcal{M}_3$ in the last stage. We summarized the success rate of this experiment for



(a) Positions of the end-effector w.r.t. the box to open

(b) Orientations of the end-effector w.r.t. the box to open

Figure 8: Box-opening simulation. The indices in the graphs correspond to $x_{d,n}$. (a) Gaussian distributions for positions in the three stages of the motion, expressed in $\mathcal{M}_1$ and $\mathcal{M}_2$. The trajectories generated by using these two options are shown in the bottom-left plot, where P_1, P_2, and P_3 represent the three stages. (b) Gaussian distributions for orientations in the three stages of the motion, expressed in $\mathcal{M}_1$ and $\mathcal{M}_2$.

all objects on all manifolds in Table 4. These results gathered simulations from 1800 trials (50 (random object pose)$\times$ 6 (objects)$\times$6 (5 manifolds plus the optimal one)), where bold numbers are for success rate more than 80%. Although our proposed method chose one of the five manifolds, note that there may be a slight difference in the final results as we placed the target pose randomly (for each manifold, we had different target points).

We use a Cartesian impedance controller to track the planned end-effector motion defined as a virtual point-mass to evaluate the system performance under external disturbances. We set the stiffness and damping parameters of the end-effector to $K_p = \text{diag}(1200, 1200, 1200, 100, 100, 100)$, $K_v = \sqrt{2K_p}/2$. We use this impedance controller as the low-level controller and update the path using the feedback gains calculated with the OCP in the selected manifold. The perturbed motions are shown in Fig. 14. The robot can regenerate the grasping movement while keeping the motion characteristics of the demonstration. For example, in the case of the chips can and baseball objects, the robot always points to the object along the object manifold with high compliance, which allows the robot to move along that manifold, even in case of external perturbations. For bowl grasping, in addition to the above feature, the robot can also find the nearest edge to grasp based on the current state during the perturbation. For prismatic objects, the robot has low compliance outside the planned linear grasping trajectory. Additional reproduction attempts on the real robot are presented in Fig. 13 and in the accompanying video.

5.5. Autonomous box opening with the Franka Emika robot

We repeated the same protocol in a box-opening task to test our approach. We attached Aruco markers to the demonstrator's hand to track the hand motion and gathered six demonstrations. The task is considered successful if the box is opened with a constant opening radius with the gripper not applying unnecessary excessive force on the lid. We used a GMM to extract four stages of the task and employed the OCP approach described in Sec. 5.4 to track the trajectory generated by GMR. Snapshots of this task are shown in Fig. 18(a) (see the complete process in the accompanying video). The experiment shows that the skill can be generalized to different robots and box configurations. The robot can keep a constant distance from the axis of rotation, which is an essential property of the opening stage (the third stage). This property cannot be fulfilled in $\mathcal{M}_1$ or $\mathcal{M}_3$, where the robot either slides along the radial direction or fails to reach the edge of the object, which will cause potential damage to the robot or failure to open the box. Other tasks, such as opening most of the doors, steering wheels, or turning a book page, can be described with similar skills.

We use an impedance controller to perform the opening movement under perturbations in radius and axis direction. We keep the same protocol and parameters as for the grasping experiment. We show the reproduced motions in Fig. 17

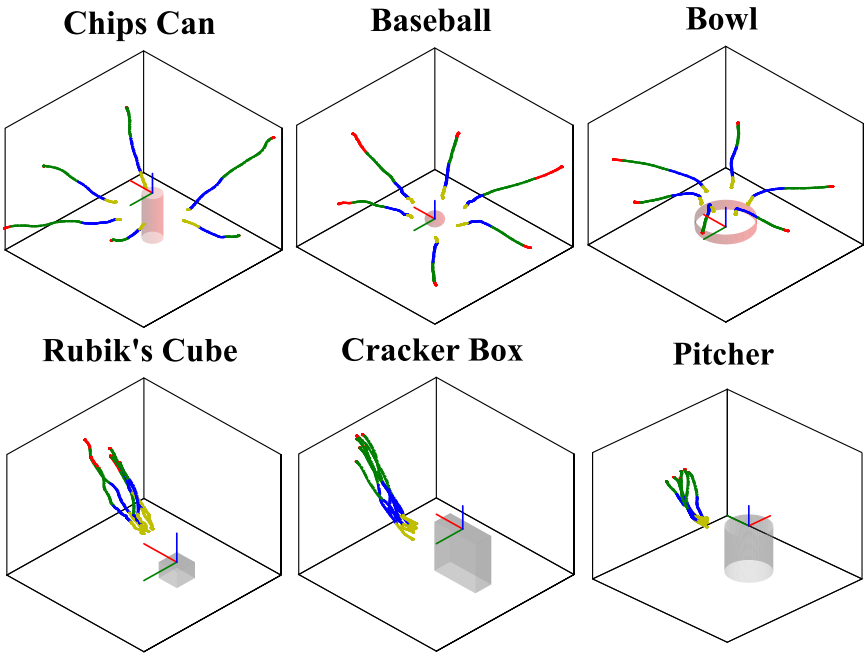


Figure 9: Demonstrations for the grasping task with six different objects. The demonstrations are shown in the object frame and the trajectories with different colors are clustered with a GMM.

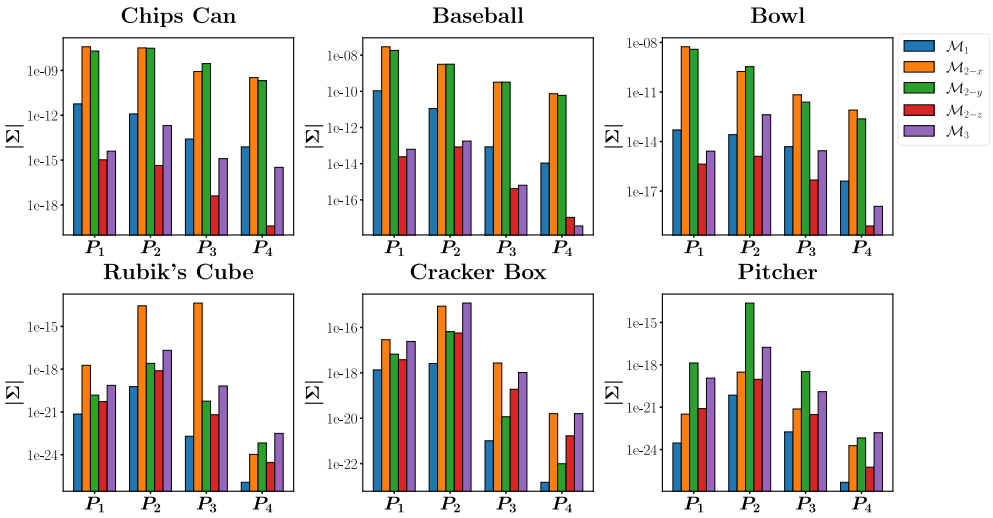


Figure 10: Determinants of the covariance matrices in the four stages (in logarithm scale). Blue, orange, green, red and purple bars represent the values of $\mathcal{M}_1$, $\mathcal{M}_{2-x}$, $\mathcal{M}_{2-y}$,$\mathcal{M}_{2-z}$ and $\mathcal{M}_3$. The manifold with the smallest determinant is the selected coordinate system for each stage of the movement.

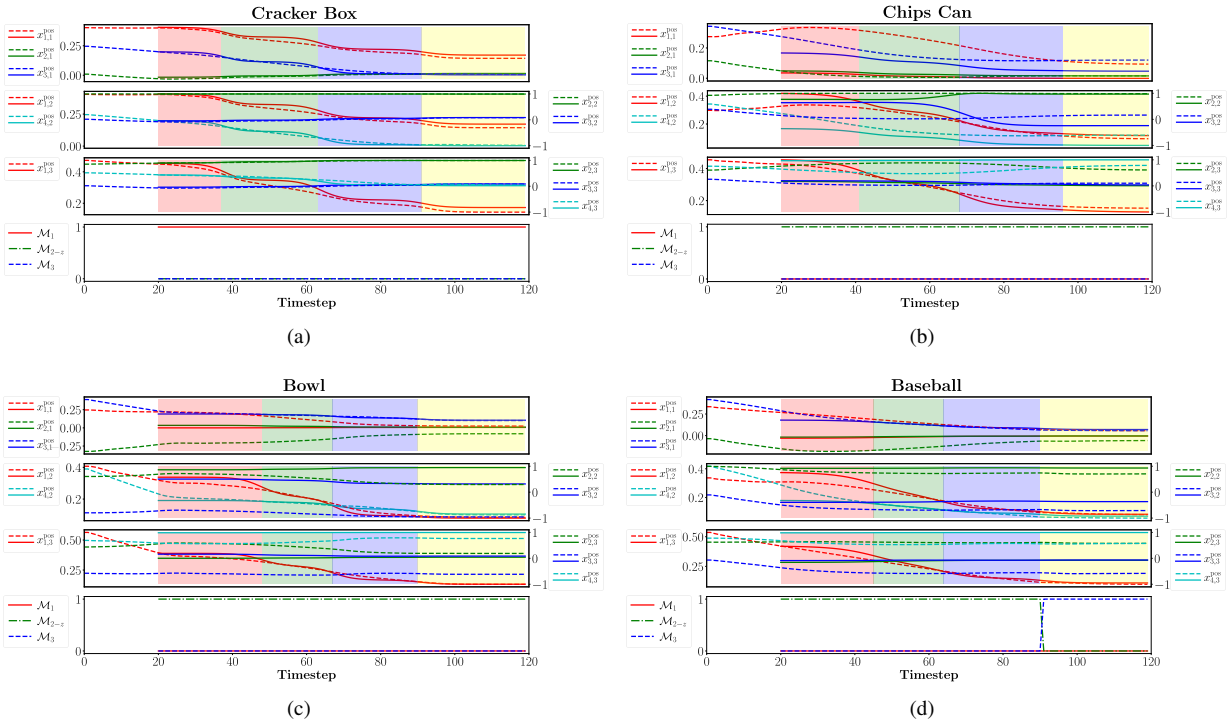


Figure 11: Timeline graphs of four typical objects. The solid lines represent the references in $\mathcal{M}_1$, $\mathcal{M}_{2-z}$ and $\mathcal{M}_3$ generated from a GMM with four components (four consecutive stages of the motion depicted in red, green, blue and yellow). The dashed lines represent a grasping movement generated from a different initial pose, by using the selected manifold at each time step. The bottom plot shows the selected manifold for the different time steps (here, $\mathcal{M}_{2-z}$ then $\mathcal{M}_3$ are selected by the algorithm).

and Fig. 18(b) (also in accompanying video). The robot can open the box successfully with perturbations in different directions, except for the opening stage, where the robot is pulled away from the lid. In particular, we can observe that the user can drag the robot along the edge where it is endowed with high compliance, but has difficulty pushing it in the radius direction. With the different compliance learned from the demonstrations, the user can use this property to correct the robot's movement, or the robot can avoid the obstacle while maintaining the principal characteristics of the movement.

5.6. User-guided objects grasping with the Franka Emika robot

Finally, we tested our algorithm in a shared control scenario in which adaptive virtual guides are used to assist the user in grasping tasks. In this application, the user guides the robot end-effector while the robot is automatically adapting its position and orientation according to the task and to the object (adaptive virtual guides). This adaptive shared control behavior can be helpful in diverse tasks, including remote teleoperation, shared control by kinesthetic guidance, human-robot collaborative transportation, or daily living assistance with exoskeletons and prosthetics. In human-robot co-manipulation, virtual guides are an essential tool to assist the human worker by reducing physical effort and cognitive overload during task accomplishment [55]. There are potential applications in this field using our approach, where the robot grasps a target object under human guidance and applies the corresponding grasping strategy learned from the demonstration for objects with different shape characteristics. In this experiment, we use the phase estimation technique described in Sec. 4.3 to find the current phase of the system and extract the desired values of active variables from the demonstrations.

We first get the desired trajectories of the robot to reach each of the objects as well as proper feedback gains using the method mentioned in the previous sections. We use the Euclidean distance between the current position of the robot end-effector and the desired path created for each object to estimate the current phase of the system, i.e., we replaced $\boldsymbol{x}_t^r = \boldsymbol{x}_{\mathcal{M}_1,t}$ and $\mathcal{M} = \mathcal{M}_1$ in (19). With this approach, not only can we find the current phase of the system but also which object we are approaching.

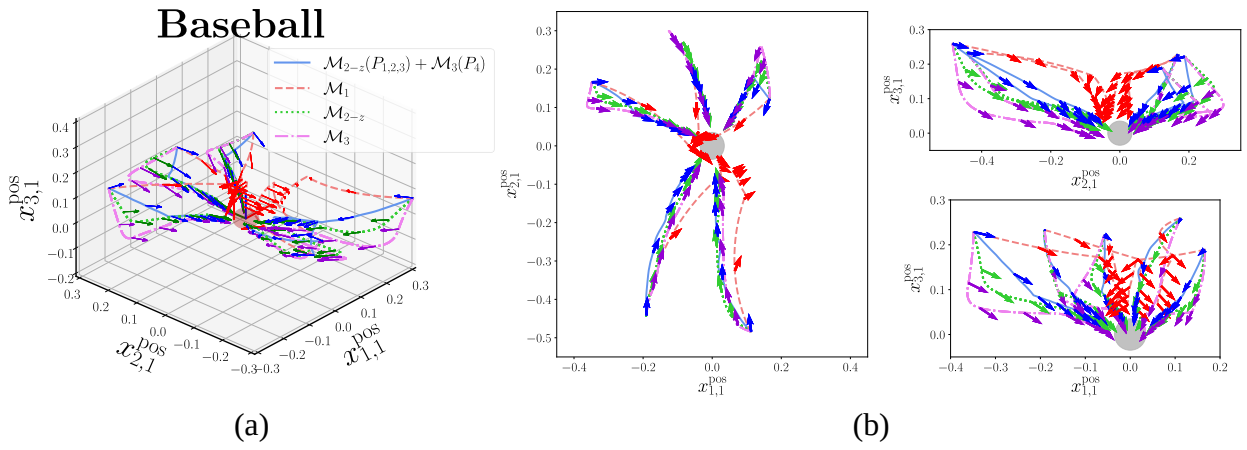
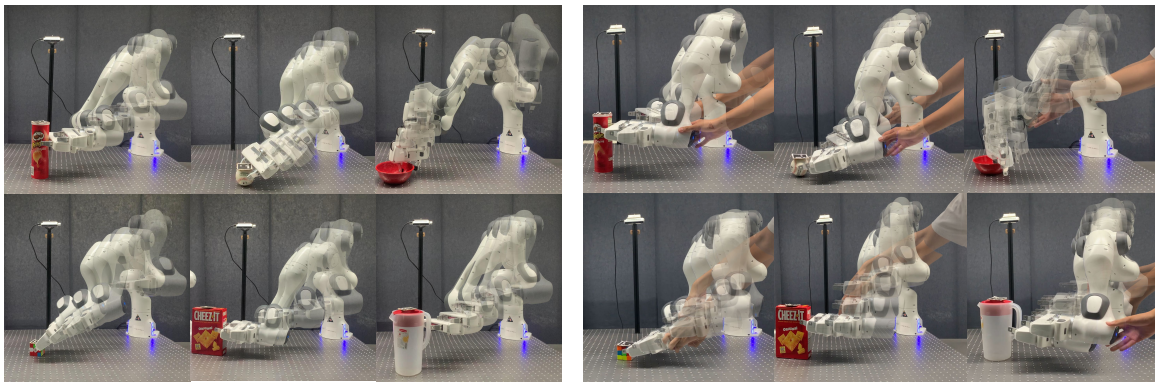


Figure 12: (a) Reproduced trajectories by using the optimal manifold method (blue solid lines) and by using only one manifold ($\mathcal{M}_1$ (red dashed lines), $\mathcal{M}_{2-z}$ (green dot lines) and $\mathcal{M}_3$ (green dashed dot lines)) in 3D space. The arrows represent the directions of the end-effector. (b) The projected trajectories from 3D space to three orthogonal planes, with the arrows representing the orientation.



(a) Open-loop Control (b) Feedback Control

Figure 13: The learned grasping manipulation trials with random placements of the objects (with and without human perturbation).

When the robot is far from all objects, the controller enters into a fully compliant behavior. When it is brought near one of the objects (with a predetermined threshold), it actively controls the positions and orientation of the end-effector according to the *learned* feedback gains. For both position and orientation, if one direction has a low variance in the demonstrations, the robot will apply control effort to reject perturbations along this direction, effectively correcting the guidance so that the user does not need to care about being precise in this direction (adaptive virtual barrier guiding the robot end-effector). These virtual guides typically define a spatial region instead of a single point, which still allows the user to remain in charge of the object being selected for grasping, of the approaching part of the motion and of the dynamics used to approach and manipulate the objects. Note that the feedback gains in each direction are computed based on the variations in the demonstrations at some δs ahead of the current phase determined by the phase estimation algorithm. This anticipative behavior is adjusted manually to provide smoother shared control by compensating for the frictional effects of the robot.

We choose two shape primitive objects, the chips can and the baseball, which we place randomly within the robot workspace. We first drag the robot near the baseball and then near the chips can. We observe that the robot can be moved in free motion when it is far from both objects. When it is close to the baseball, the user can still easily rotate the end-effector along the spherical manifold while the gripper points to the baseball all the time. In contrast, when it is close to the chips can, the gripper points horizontally toward the object. It only allows the user to rotate the end-effector around the central axis of the chips can. The resulting shared control behavior provides a grasping assistance

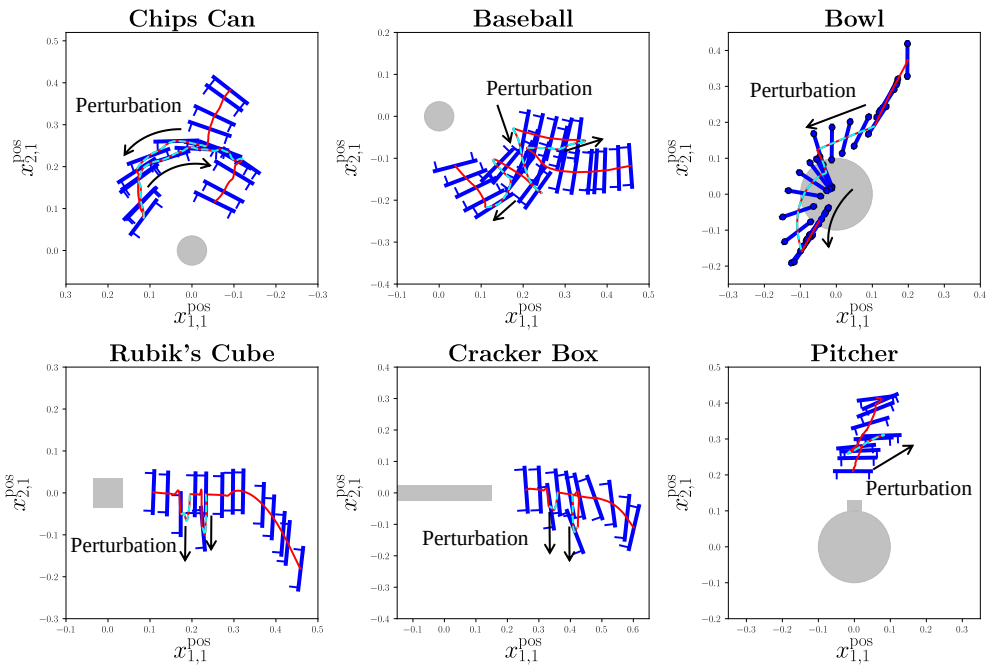


Figure 14: Reproduced trajectories generated by feedback control with the optimal manifold method (solid red lines), including perturbation (dashed cyan line). The blue symbols represent the gripper of the end-effector and the grey shades represent the grasped objects.

that automatically adapts to the shapes and affordances of objects, which are learned from only few demonstrations. The whole process is shown in Fig. 19 and in the accompanying video.

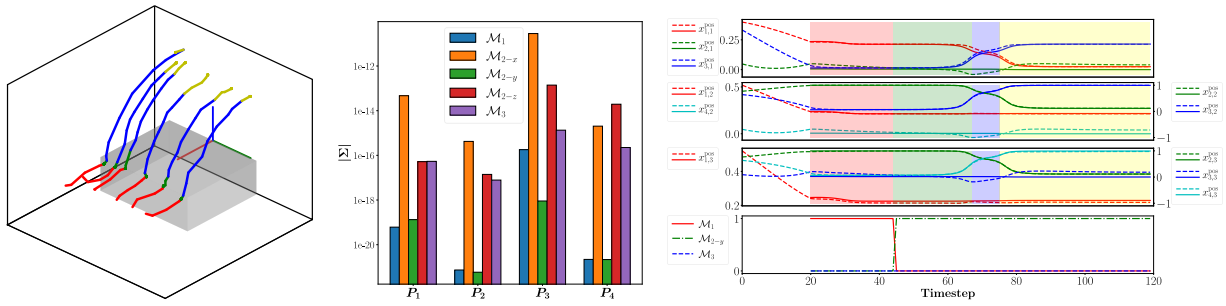
6. Discussion

One of the biggest challenges in LfD is that we are limited to few demonstrations. Typically, the recorded demonstrations cannot describe all the aspects of the task if we try to learn those in a black-box manner. These challenges motivated researchers to gather information from other sources, such as providing structures and representations that can generalize to a large range of manipulation skills.

In this article, we proposed a *winner-takes-all* strategy by estimating which manifold (coordinate system) represents the task in the most consistent manner. We showed how this method provides better compliance behavior by exploiting the property of *do-not-matter* directions. Compared to standard task-parameterized models [43], our approach explores the use of multiple types of coordinate systems that can be attached to multiple objects, rather than using a single type of coordinate systems attached to multiple objects.

We showed that to obtain curved shape distributions like circles and spheres, a single Gaussian is enough by considering such a variety of manifolds. By considering only a Cartesian coordinate system, one Gaussian is not enough to represent the task, and more elaborated distribution models such as mixtures of Gaussians would not generalize as well as with the proposed strategy, as they would require many demonstrations that would also need to be carefully planned to cover such shapes. In contrast, we showed that our approach can rely on very few demonstrations, by exploiting the shape structure of the object and/or task (Fig.6(a) and Fig.8(a)).

For the implementation of the proposed approach, we chose to keep a single manifold to represent the task at each stage of the motion (*winner-takes-all* strategy) and to ignore the information modeled in other manifolds. Future work could extend this approach by fusing the information from the different manifolds, as it is done in task-parameterized probabilistic models [43]. This could be achieved by considering a Gaussian product on Riemannian manifold. Such a fusion strategy needs further investigations. On the one hand, the *winner-takes-all* approach that we currently use might lose some intrinsic information, but on the other hand, it acts as some form of regularization, which might be important when only few demonstrations are considered.



(a) Demonstrations (b) Determinants of covariance matrices (c) Timeline evolution

Figure 15: (a) Demonstrations of the box-opening movement, with different colors representing different stages of the movement. (b) Determinants of covariance matrices in the four stages (in logarithmic scale). The manifold with the smallest determinant is the selected coordinate system for each stage of the movement. (c) In the first three plots, the solid lines represent the references for $\mathcal{M}_1$, $\mathcal{M}_{2-y}$ and $\mathcal{M}_3$. The dashed lines represent a grasping movement generated from a different initial pose, by using the selected manifold at each time step. The bottom plot shows the selected manifold for the different time steps (here, $\mathcal{M}_1$ then $\mathcal{M}_{2-y}$ are selected).

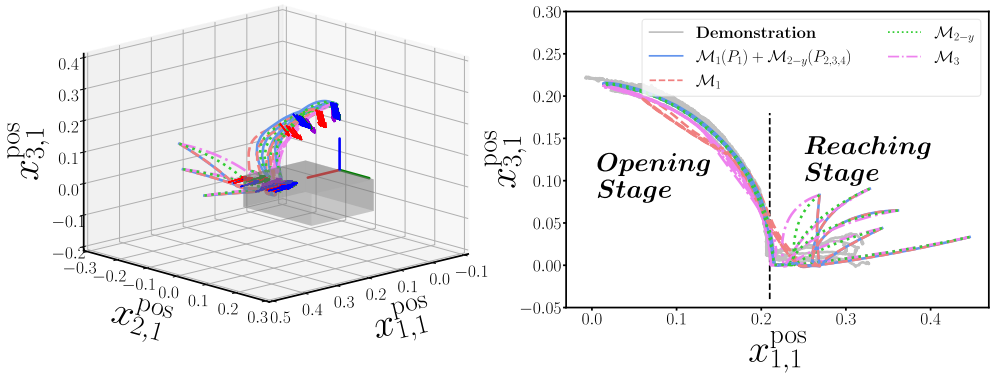


Figure 16: Reproduced box-opening movements for different initial states.

In the proposed experiment, each shape primitive provides a different grasping strategy by prioritizing directions non-identically, which improves the compliance behavior of the system. This fact is shown in the virtual guidance experiment. The phase-estimation module plays an important role in this task by providing better human-robot interaction. This technique allows us to implement the solution of OCP as a time-independent policy that does not require expensive computation. The phase estimation is done after the optimization without impacting it. It also does not add much computational load. For this task, the phase estimation time was in the order of $O(0.01ms)$. In the proposed shared control experiment, there were only two simple shape objects in the workspace. Further work could be extended to more complex scenarios consisting of objects composed of multiple local shape primitives. Obstacle avoidance will also be considered in future work, where the robot can adjust the avoidance movement according to the geometric property of the obstacle. The phase estimation technique can also be modified to consider additional features, such as forces and orientations.

7. Conclusion

In this article, we proposed a learning from demonstration approach considering different types of coordinate systems. The goal is to reduce the number of demonstrations required by a robot to acquire manipulation skills. The data distribution is extracted using a model of Gaussian distributions on Riemannian manifolds. We showed how standard optimal control formulation could be easily extended to other manifolds, by learning the structure of the cost and the precision matrices used in the cost from a very small set of demonstrations. We showed that our approach could

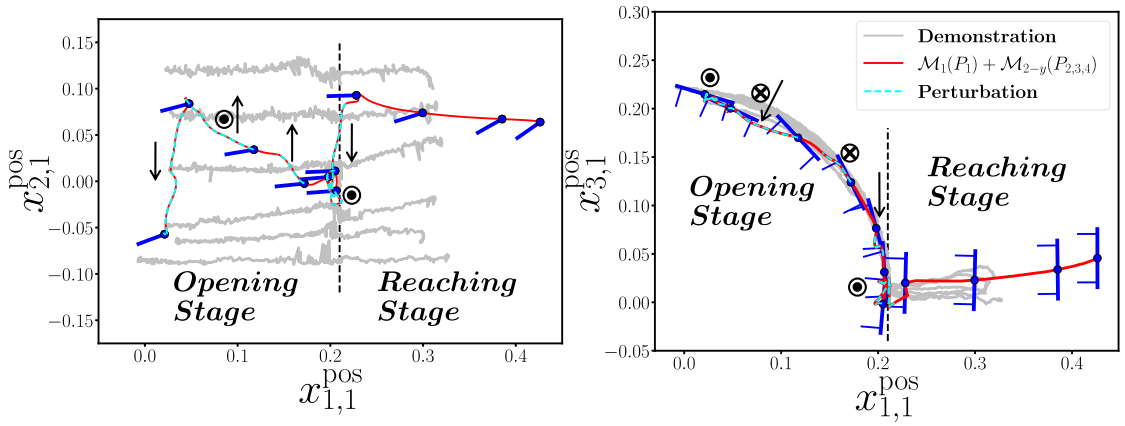
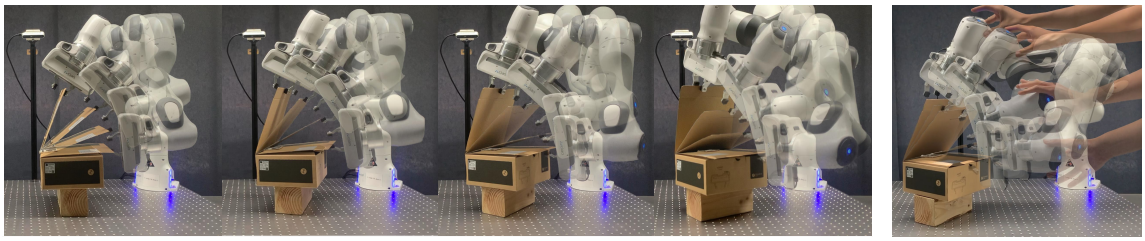


Figure 17: Reproduced box-opening movements using impedance feedback control in the optimal manifold combination under the perturbation. Blue symbols represent the gripper of robot and black arrows (→), dot circles (⊙) and cross circles (⊗) represent the perturbation direction.



(a) Open-loop Control (b) Feedback Control

Figure 18: The learned box-opening manipulation trials with random placement of the box (with and without human perturbation).

be applied in both open-loop and feedback control frameworks. The feedback version can reject external perturbations by considering the symmetries and affordances of the object, by efficiently exploiting the variations allowed by the task. We demonstrated this approach in both autonomous and shared control tasks. In the latter, the robot can be guided by the user while automatically determining the grasping strategy to assist the user in the different steps of the task, according to his/her grasping intention. We compared our approach to the case of using merely one type of coordinate system, as used in previous work. We showed that our proposed adaptive assistance could be learned from very few demonstrations, with experimental results demonstrating the high generalization capability of the approach. The method provides a formal and intuitive method to regulate compliance behaviors in manipulation tasks, with a geometric structure to describe the variations allowed by the task.

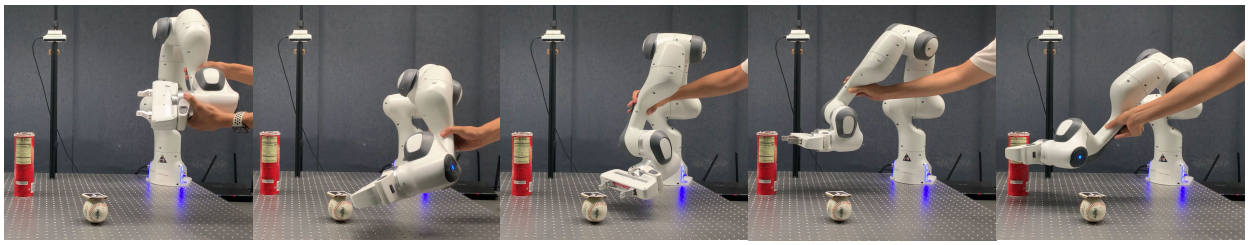


Figure 19: Shared control with virtual guides for the collaborative grasping of different objects.

Acknowledgement

This work was supported by the China Scholarship Council (CSC, No.202006120159) funded by the Major Research Plan of the National Natural Science Foundation of China (No. 92048301), and by the SWITCH project (https://switch-project.github.io/) funded by the Swiss National Science Foundation.

References

[1] E. Todorov, M. I. Jordan, A minimal intervention principle for coordinated movement, in: Advances in Neural Information Processing Systems (NeurIPS), 2002, pp. 27–34.
[2] J. P. Scholz, G. Schoener, The uncontrolled manifold concept: identifying control variables for a functional task, Experimental Brain Research 126 (3) (1999) 289–306.
[3] D. M. Wolpert, J. Diedrichsen, J. R. Flanagan, Principles of sensorimotor learning, Nature Reviews 12 (2011) 739–751.
[4] D. Sternad, S.-W. Park, H. Mueller, N. Hogan, Coordinate dependence of variability analysis, PLoS Comput. Biol. 6 (4) (2010) 1–16.
[5] J. A. S. Kelso, Synergies: Atoms of brain and behavior, in: D. Sternad (Ed.), Progress in Motor Control, Vol. 629 of Advances in Experimental Medicine and Biology, Springer US, 2009, pp. 83–91.
[6] J.-P. Laumond, N. Mansard, J.-B. Lasserre, Geometric and Numerical Foundations of Movements, Springer, 2018.
[7] D. Bennequin, R. Fuchs, A. Berthoz, T. Flash, Movement timing and invariance arise from several geometries, PLoS Comput. Biol. 5 (7) (2009) 1–27.
[8] G. Ganesh, E. Burdet, Motor planning explains human behaviour in tasks with multiple solutions, Robotics and Autonomous Systems 61 (4) (2013) 362–368.
[9] B. Ti, Y. Gao, J. Zhao, S. Calinon, Imitation of manipulation skills using multiple geometries, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 7391–7398.
[10] M. T. Mason, Compliance and force control for computer controlled manipulators, IEEE Trans. on Systems, Man, and Cybernetics 11 (6) (1981) 418–432.
[11] A. Kaiser, J. A. Ybanez Zepeda, T. Boubekeur, A survey of simple geometric primitives detection methods for captured 3d data, in: Computer Graphics Forum, Vol. 38, Wiley Online Library, 2019, pp. 167–196.
[12] C. Romanengo, A. Raffo, Y. Qie, N. Anwer, B. Falcidieno, Fit4cad: A point cloud benchmark for fitting simple geometric primitives in cad objects, Computers & Graphics.
[13] L. Li, M. Sung, A. Dubrovina, L. Yi, L. J. Guibas, Supervised fitting of geometric primitives to 3d point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2652–2660.
[14] G. Borghesan, E. Scioni, A. Kheddar, H. Bruyninckx, Introducing geometric constraint expressions into robot constrained motion specification and control, IEEE Robotics and Automation Letters 1 (2) (2015) 1140–1147.
[15] C. Zeng, X. Chen, N. Wang, C. Yang, Learning compliant robotic movements based on biomimetic motor adaptation, Robotics and Autonomous Systems 135 (2021) 103668.
[16] B. Ti, Y. Gao, M. Shi, J. Zhao, Generalization of orientation trajectories and force–torque profiles for learning human assembly skill, Robotics and Computer-Integrated Manufacturing 76 (2022) 102325.
[17] A. G. Billard, S. Calinon, R. Dillmann, Learning from humans, in: B. Siciliano, O. Khatib (Eds.), Handbook of Robotics, Springer, Secaucus, NJ, USA, 2016, Ch. 74, pp. 1995–2014, 2nd Edition.
[18] S. Calinon, Learning from demonstration (programming by demonstration), in: M. H. Ang, O. Khatib, B. Siciliano (Eds.), Encyclopedia of Robotics, Springer, 2019.
[19] C. Pérez-D'Arpino, J. A. Shah, C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy, in: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), 2017, pp. 4058–4065.
[20] G. Subramani, M. Zinn, M. Gleicher, Inferring geometric constraints in human demonstrations, in: Proc. Conference on Robot Learning (CoRL), 2018, pp. 223–236.
[21] M. Vochten, T. De Laet, J. De Schutter, Generalizing demonstrated motion trajectories using coordinate-free shape descriptors, Robotics and Autonomous Systems 122 (2019) 103291.
[22] S. Calinon, D. Bruno, D. G. Caldwell, A task-parameterized probabilistic model with minimal intervention control, in: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 3339–3344.
[23] D. Zhang, W. Si, W. Fan, Y. Guan, C. Yang, From teleoperation to autonomous robot-assisted microsurgery: A survey, Machine Intelligence Research (2022) 1–19.
[24] Z. Li, S. Zhao, J. Duan, C.-Y. Su, C. Yang, X. Zhao, Human cooperative wheelchair with brain–machine interaction based on shared control strategy, IEEE/ASME Transactions on Mechatronics 22 (1) (2016) 185–195.
[25] J. Luo, Z. Lin, Y. Li, C. Yang, A teleoperation framework for mobile robots based on shared control, IEEE Robotics and Automation Letters 5 (2) (2019) 377–384.
[26] L. Rosenberg, Virtual fixtures: Perceptual tools for telerobotic manipulation, in: Proceedings of IEEE Virtual Reality Annual International Symposium, 1993, pp. 76–82. doi:10.1109/VRAIS.1993.380795.
[27] G. Raiola, X. Lamy, F. Stulp, Co-manipulation with multiple probabilistic virtual guides, in: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, 2015, pp. 7–13.
[28] S. Bodenstedt, N. Padoy, G. D. Hager, Learned partial automation for shared control in tele-robotic manipulation., in: AAAI Fall Symposium: Robots Learning Interactively from Human Teachers, 2012.
[29] B. Nemec, N. Likar, A. Gams, A. Ude, Human robot cooperation with compliance adaptation along the motion trajectory, Autonomous robots 42 (5) (2018) 1023–1035.

[30] B. Nemec, A. Gams, A. Ude, Velocity adaptation for self-improvement of skills learned from user demonstrations, in: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), IEEE, 2013, pp. 423–428.
[31] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, J. Lygeros, Optimization-based hierarchical motion planning for autonomous racing, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 2397–2403.
[32] H. Ren, S. Chen, L. Yang, Y. Zhao, Optimal path planning and speed control integration strategy for ugvs in static and dynamic environments, IEEE Transactions on Vehicular Technology 69 (10) (2020) 10619–10629.
[33] Z. Ju, H. Liu, Fuzzy gaussian mixture models, Pattern Recognition 45 (3) (2012) 1146–1158.
[34] H. Zhang, Y. Leng, Motor skills learning and generalization with adapted curvilinear gaussian mixture model, Journal of Intelligent & Robotic Systems 96 (3) (2019) 457–475.
[35] M. Diehl, H. Bock, H. Diedam, P.-B. Wieber, Fast direct multiple shooting algorithms for optimal robot control, in: M. Diehl, K. Mombaur (Eds.), Fast Motions in Biomechanics and Robotics, Vol. 340, Springer Berlin Heidelberg, pp. 65–93. doi:10.1007/978-3-540-36119-0_4.
[36] A. Bemporad, M. Morari, V. Dua, E. N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, Automatica 38 (1) (2002) 3–20.
[37] D. Mayne, A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems, International Journal of Control 3 (1) (1966) 85–95.
[38] W. Li, E. Todorov, Iterative linear quadratic regulator design for nonlinear biological movement systems, in: Proc. Intl Conf. on Informatics in Control, Automation and Robotics (ICINCO), 2004, pp. 222–229.
[39] M. M. Hasan, X. W. Tangpong, O. P. Agrawal, Fractional optimal control of distributed systems in spherical and cylindrical coordinates, Journal of Vibration and Control 18 (10) (2012) 1506–1525.
[40] M. B. Kobilarov, J. E. Marsden, Discrete geometric optimal control on lie groups, IEEE Transactions on Robotics 27 (4) (2011) 641–655. doi:10.1109/TRO.2011.2139130.
[41] J. Campbell, H. B. Amor, Bayesian interaction primitives: A SLAM approach to human-robot interaction, in: Proceedings of the 1st Annual Conference on Robot Learning (CoRL), PMLR, pp. 379–387.
[42] S. Stepputtis, M. Bandari, S. Schaal, H. B. Amor, A system for imitation learning of contact-rich bimanual manipulation policies. arXiv:2208.00596[cs].
[43] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intelligent Service Robotics 9 (1) (2016) 1–29. doi:10.1007/s11370-015-0187-9.
[44] C. Eppner, O. Brock, Grasping unknown objects by exploiting shape adaptability and environmental constraints, in: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), 2013, pp. 4000–4006.
[45] S. Calinon, Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control, IEEE Robotics and Automation Magazine (RAM) 27 (2) (2020) 33–45. doi:10.1109/MRA.2020.2980548.
[46] Y. Tassa, T. Erez, E. Todorov, Synthesis and stabilization of complex behaviors through online trajectory optimization, Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS) (2012) 4906–4913.
[47] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, D. G. Caldwell, An approach for imitation learning on Riemannian manifolds, IEEE Robotics and Automation Letters (RA-L) 2 (3) (2017) 1240–1247.
[48] S. Arimoto, M. Yoshida, M. Sekimoto, K. Tahara, A riemannian-geometry approach for modeling and control of dynamics of object manipulation under constraints, Journal of Robotics 2009.
[49] A. Biess, T. Flash, D. G. Liebermann, Riemannian geometric approach to human arm dynamics, movement optimization, and invariance, Physical Review E 83 (3) (2011) 031927.
[50] P. A. Absil, R. Mahony, R. Sepulchre, Optimization Algorithms on Matrix Manifolds, Princeton University Press, 2007.
[51] X. Pennec, Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements, Journal of Mathematical Imaging and Vision 25 (1) (2006) 127–154.
[52] E. Simo-Serra, C. Torras, F. Moreno-Noguer, 3D human pose tracking priors using geodesic mixture models, International Journal of Computer Vision 122 (2) (2017) 388–408.
[53] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: Learning attractor models formotor behaviors, Neural Computation 25 (2) (2013) 328–373.
[54] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, M. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition 47 (6) (2014) 2280–2292.
[55] S. S. Restrepo, G. Raiola, P. Chevalier, X. Lamy, D. Sidobre, Iterative virtual guides programming for human-robot comanipulation, in: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), IEEE, 2017, pp. 219–226.

Appendix A. (iterative) Linear Quadratic Regulator ((i)LQR) method

We can describe all states $\boldsymbol{x}$ as a linear function of the initial state $\boldsymbol{x}_1$ and the applied control commands $\boldsymbol{u}$ as

$$\boldsymbol{x} = \boldsymbol{S}_x \boldsymbol{x}_1 + \boldsymbol{S}_u \boldsymbol{u}, \tag{A.1}$$

where

$$\boldsymbol{S_x} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{T-1} \end{bmatrix}, \boldsymbol{S_u} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{T-2}\boldsymbol{B} & \boldsymbol{A}^{T-3}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \tag{A.2}$$

The cost function can also be modified as

$$c(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}, \tag{A.3}$$

where $\boldsymbol{Q} = \text{blockdiag}\,(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_T)$ and $\boldsymbol{R} = \text{blockdiag}\,(\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_{T-1})$. Substituting (A.1) and (A.2) to (A.3), we obtain

$$c(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{u}^\top (\boldsymbol{S}_u^\top \boldsymbol{Q}\boldsymbol{S}_u + \boldsymbol{R})\boldsymbol{u} + 2\boldsymbol{u}^\top \boldsymbol{S}_u^\top \boldsymbol{Q}\boldsymbol{S}_x \boldsymbol{x}_1 + \boldsymbol{x}_1^\top \boldsymbol{S}_x^\top \boldsymbol{Q}\boldsymbol{S}_x \boldsymbol{x}_1. \tag{A.4}$$

The optimal control command $\boldsymbol{u}$ is computed by differentiating (A.4) with respect to $\boldsymbol{u}$ and equating to zero, providing the batch form solution

$$\boldsymbol{u} = -(\boldsymbol{S}_u^\top \boldsymbol{Q}\boldsymbol{S}_u + \boldsymbol{R})^{-1} \boldsymbol{S}_u^\top \boldsymbol{Q}\boldsymbol{S}_x \boldsymbol{x}_1. \tag{A.5}$$

For non-quadratic costs and/or non-linear dynamics, the problem can be solved iteratively by starting from an initial estimate and by computing a Taylor expansion of the cost and dynamics $\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ around the point $(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)$, namely

$$\boldsymbol{x}_{t+1} \approx \boldsymbol{f}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t}(\boldsymbol{u}_t - \hat{\boldsymbol{u}}_t) \iff \Delta\boldsymbol{x}_{t+1} \approx \boldsymbol{A}_t \Delta\boldsymbol{x}_t + \boldsymbol{B}_t \Delta\boldsymbol{u}_t,$$

with error terms $\{\Delta\boldsymbol{x}_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t, \Delta\boldsymbol{u}_t = \boldsymbol{u}_t - \hat{\boldsymbol{u}}_t\}$, and Jacobian matrices $\{\boldsymbol{A}_t = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t}, \boldsymbol{B}_t = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_t}\}$.

The cost function can also be quadratized as

$$c(\boldsymbol{x}_t, \boldsymbol{u}_t) \approx c(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) + \Delta\boldsymbol{x}_t^\top \frac{\partial c}{\partial \boldsymbol{x}_t} + \Delta\boldsymbol{u}_t^\top \frac{\partial c}{\partial \boldsymbol{u}_t} + \frac{1}{2}\Delta\boldsymbol{x}_t^\top \frac{\partial^2 c}{\partial \boldsymbol{x}_t^2}\Delta\boldsymbol{x}_t + \Delta\boldsymbol{x}_t^\top \frac{\partial^2 c}{\partial \boldsymbol{x}_t \boldsymbol{u}_t}\Delta\boldsymbol{u}_t + \frac{1}{2}\Delta\boldsymbol{u}_t^\top \frac{\partial^2 c}{\partial \boldsymbol{u}_t^2}\Delta\boldsymbol{u}_t. \tag{A.6}$$

The resulting approach is called iLQR [37, 38], where an LQR problem is solved in each optimization step. with gradients $\{\frac{\partial c}{\partial \boldsymbol{x}_t}, \frac{\partial c}{\partial \boldsymbol{u}_t}\}$, and Hessian matrices $\{\frac{\partial^2 c}{\partial \boldsymbol{x}_t^2}, \frac{\partial^2 c}{\partial \boldsymbol{x}_t \boldsymbol{u}_t}, \frac{\partial^2 c}{\partial \boldsymbol{u}_t^2}\}$. It means that at each iteration, we should update $\hat{\boldsymbol{u}}$ with

$$\Delta\boldsymbol{u} = \left(\boldsymbol{S}_u^\top \boldsymbol{H}_x \boldsymbol{S}_u + 2\boldsymbol{S}_u^\top \boldsymbol{H}_{xu} + \boldsymbol{H}_u\right)^{-1} \left(-\boldsymbol{S}_u^\top \boldsymbol{g}_x - \boldsymbol{g}_u\right), \tag{A.7}$$

and repeat this step until the system converges to a local optimum.

Appendix B. Detail of the experiment setting

Appendix B.1. Demonstation Sampling

In the simulation part, we simplify the process by generating the demonstration manually to validate our approach. In the planar grasping task, the demonstration consists of the key points representing each motion stage marked by different colors in Fig. 6(a). We generate the motion from different directions, and in each motion, we attempt to simulate human grasping habits by adding random noise to the orientation, based on the pose of orienting toward the target, with the amplitude of the noise decreasing as it approaches the object. In the planar box-opening task, we generated one demonstration, which is enough to include the manipulation feature. We add a short pause at the beginning and at the end of the motion to clearly distinguish each stage of the task (preparing stage, opening stage and finishing stage). In the Fig. 8(a), we can observe that the green points and blue data points gather together, respectively, which means there is a stop at two stages. These red points widely spread reflecting the process of opening motion.

In the real experiment part, for the grasping task, we use the kinesthetic teaching method to demonstrate a robot grasping each object six times. The grasping direction for the chips can, baseball and bowl is distributed along the geometric shape, and the human's grasping habit decides the grasping point's height. For the prismatic objects, the grasping point is along the height of the object. For the box-opening task, we attached Aruco markers to the demonstrator's hand to record the hand motion six times, where the starting point of the opening movements was distributed evenly along the edge of the box.

Appendix B.2. Demonstation Learning and Generalization

Due to the manually designed demonstration, we already have data points for each stage in the simulation. So, we construct the Gaussian distribution of each stage using one Gaussian. We take the mean of each stage as the stepwise reference and the precision matrix of each stage as the weight $\boldsymbol{Q}$ in the cost function. The value of $\boldsymbol{R}$ is set manually. We use the method in Sec. 4.3 to change from time-driven to phase-driven. In the open-loop control case, the dynamics system is at the level of the robot kinematic, where we take the joint velocity as the state. In the feedback control, the state is defined on the end-effector pose expressed on the selected manifold and we use Cartesian impedance control to track the planned trajectory.

In the real experiment, we first use a GMM with four components to segment the demonstrations considering the spatial-temporal information into four stages (the experimenter defines the number of components). Then, we got the demonstration points for each stage of the motion used to construct their Gaussian. We use the same control protocol on the real robot.