Chapter 3

Programming industrial robots from few demonstrations

Sylvain Calinon[1]

3.1 INTRODUCTION

A wide range of industrial applications can benefit from robots acquiring manipulation skills by interaction with humans. This chapter discusses the challenges that such learning process encompasses, including the development of intuitive interfaces to acquire meaningful demonstrations, the development of representations for manipulation skills exploiting the structure and geometry of the acquired data in an efficient way, and the development of optimization and optimal control techniques that can adapt to new situations and that can exploit coordination, task variations and perception uncertainties. The chapter presents an overview of the research lines of the *Robot learning & Interaction* group at the *Idiap Research Institute*. It analyses the barriers that still need to be overcome to move these techniques out of the labs, by complying with real industrial application constraints. It also proposes a variety of research directions for which further investigations would be required to overcome these barriers.

Industrial robotics can leverage the development of robots that can acquire new skills from only few demonstrations and interactions. This challenge is multifaceted and can be addressed from varied perspectives. It often requires a wide range of research fields to be considered, as diverse as human–robot collaboration, interface design, movement primitives, motor skills, imitation learning, shared control, optimization and planning. This chapter focuses on the *learning from demonstration* (LfD) aspects [1]. It takes a practical stance by narrowing down the range of possible approaches by targeting the use of only *few* human demonstrations to program a robot, so that it can acquire new manipulation tasks fast and adapt the learned knowledge to changing situations.

[1]Idiap Research Institute, Martigny, Switzerland.
This work was supported by the COLLABORATE project (https://collaborate-project.eu), funded by the EU within H2020-DTFOF-02-2018 under grant agreement 820767.

3.2 LEARNING IN A HANDFUL OF TRIALS

The field of machine learning has evolved toward approaches relying on increasingly large datasets. In several application domains, these datasets are already available, or are relatively inexpensive to collect/produce, which motivated such an evolution. Robotics in industrial applications is characterized by a different problem setting, which contrasts to this race of using bigger and bigger amount of data in machine learning. It can be viewed as a *small data learning problem*, or as a wide-ranging learning data problem [2], with models that can start learning from small datasets, and that can still exploit additional data if such data become available during the robot's lifespan.

In contrast to other fields, the data formats in robotics vary significantly between tasks, environments, users and platforms (different sensors and actuators, not only in formats but also in modalities and organizations). The learned models often need to be interpretable to provide guarantees and to be linked to other techniques. This chapter motivates that research in industrial robot applications should target the development of learning approaches that can rely on only few demonstrations or trials.

To achieve this research agenda, the main challenges to target boil down to finding structures that can be used in a wide range of tasks, which include (from high level to low level):

- Learning structures (§3.3)
- Combination structures (§3.4)
- Geometric structures (§3.5)
- Data structures (§3.6)

which are described in the next sections of this chapter (see also [3] for a survey concentrating on the learning aspects). Examples will be used throughout the chapter to show that efficient manipulation and assembly skills acquisition requires a suitable trade-off between learning and exploitation of these different forms of structures (at model and algorithm levels).

Section 3.3 highlights the importance of considering bidirectional interaction in the learning process and the importance of providing user-friendly interfaces to facilitate learning. An example is given with a smartphone interface using augmented reality for a collaborative riveting task. Section 3.4 presents extensions of movement primitive representations. It includes task-parameterized models, which are object-centric representations of movements, and a probabilistic optimal control formulation of dynamical movement primitives. An example is given for the teleoperation of a distant robot opening valves. Section 3.5 discusses the importance of geometric representations. An example is given for the transfer of manipulation skill in a screwing task, by leveraging the use of manipulability ellipsoids as geometric descriptors. Section 3.6 finally highlights the roles of tensor factorization to extract essential information from high-dimensional data. An example is given with a peg-in-hole insertion task learned from demonstration by leveraging the use of ergodic control to efficiently search for the insertion point.

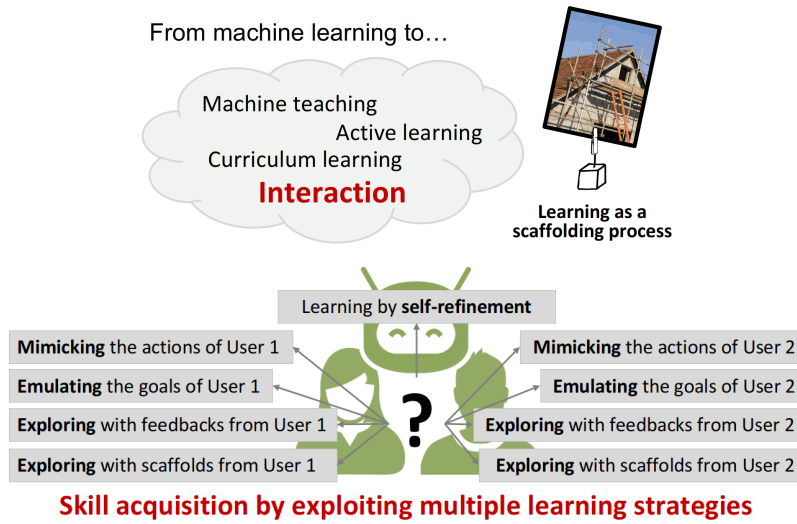


Figure 3.1: An orchestration of varied learning strategies is required to efficiently acquire new skills.

3.3 LEARNING STRUCTURES AND INTERFACES

To reduce the amount of required data, a first opportunity to seize is that machine learning in robotics goes beyond the standard training-set and testing-set paradigm. Skills acquisition in robotics is a *scaffolding process* rather than a bland learning process. The robot needs a lot of structures at the beginning and the structures can be progressively dismantled when the robot progresses: there is a continuous evolution from full assistance to full autonomy. We can thus exploit a number of interactive learning mechanisms to acquire/generate better data on-the-spot, including active learning, machine teaching (by generating data to train the robots), curriculum learning (by providing data of increased complexity adapted to the robot learner), and bilateral interactions that rely on several social mechanisms to transfer skills more efficiently, see Figure 3.1.

To acquire manipulation skills, humans greatly benefit from the combination of several learning modalities. Similarly, multiple learning strategies can jointly be exploited to transfer skills to industrial robots, while facilitating the evaluation of the robot's current capability at executing the task by the operator [4, 5]. The research direction of orchestrating different learning modalities is still in its infancy in robotics research. Currently, it essentially targets experiments in research labs. However, for industrial robot applications, this research direction has the potential to improve the robustness of the acquired manipulation and assembly skills, by allowing diverse forms of constraints to be considered (at the level of the users and the environments), as well as an improved monitoring of the robot knowledge (by testing the acquired skills in diverse situations).

The other important advantage of combining learning strategies is that it allows each individual learning modality to be simplified. Indeed, skills acquisition with a single learning strategy can be unnecessarily complex. As an illustration, we cannot learn to play a sport efficiently without practice, by only observing others play. Similarly, in a manufacturing environment, we cannot acquire efficiently a fabrication skill by limiting the available information to only end-goals (or final products). We instead need to observe an expert or to be guided throughout the process to acquire the underlying fabrication strategies. Similarly to us, robots might not be able to acquire skills efficiently by using a single learning modality. Thus, instead of focusing on the improvement of algorithms and models for a specific learning strategy, industrial robotics could highly benefit from further research to investigate the meta-learning problem of combining efficiently existing learning modalities, without defining the sequence and/or organization in advance [6].

3.3.1 User interfaces for skill transfer

Robots acquiring skills by user guidance require the development of user-friendly interfaces for the operators to teach and monitor the acquisition of manipulation skills. First, this is important to let the operators provide useful demonstrations of the task, by allowing them to concentrate on the skill to transfer instead of nonessential hindering factors. This is also important to let them assess the current capability of the robot, in order to provide demonstrations or corrections that are the most appropriate with respect to the current capability of the robot [4, 5]. Thus, interfaces enabling bidirectional interactions need to be developed within the context of industrial applications.

Currently, the most commonly used interfaces to teach robots new tasks in industrial applications are the *teaching pendants*. Recent torque-controlled or collaborative robot manipulators brought a new programming interaction capability, by letting operators drive the robots manually, with backdrivable motors and controllers that compensate for the effect of gravity. The resulting interfaces allow users to move robots freely in space as if there was no motorization in the articulations, and as if the robots had no weight. This *kinesthetic teaching* capability can be extended to more elaborated forms of virtual fixtures by exploiting impedance through the torque control capability of the robot, which provides a physical guidance capability to facilitate manipulation skill transfer [7].

Research in *virtual reality* (VR), *augmented reality* (AR) and *mixed reality* (MR), gathered under the name of *extended reality* (XR), offer promising alternative approaches to interact with robots, for both teaching and monitoring aspects. A subset of these approaches rely on the AR capability of recent smartphones or tablets, by displaying the video stream from the cameras on the back of the devices, and by superposing to these images virtual robots and/or virtual guides (typically, paths, labels or coordinate systems), see [8] for a review. The smartphone/tablet looks like a transparent window that the operator can freely move to observe a virtual scene superposed to the real scene. By changing the angle of view and by moving closer/further away, the user can intuitively analyze existing movements and program

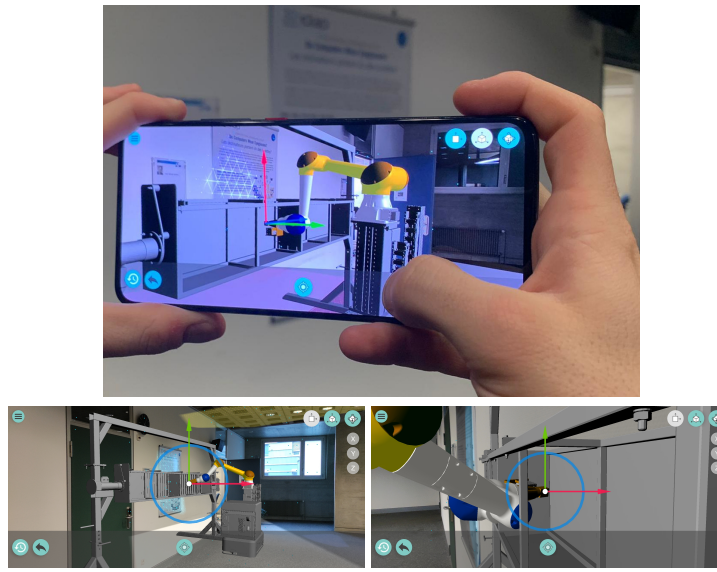


Figure 3.2: Smartphone interface using augmented reality (AR) capability, exploited in an industrial application requiring the operator to program a series of riveting poses to a 6-axis HC10 Yaskawa manipulator fixed to an elevating platform mounted on an AGV.

new robot movements. It facilitates the inspection of a taught path by replaying consecutively a robot motion on the virtual robot before executing it on the real robot.

In [9], this approach was tested in the context of industrial (de)insertion tasks with a NIST Task Board [10], a benchmark platform used to evaluate assembly performance. Figure 3.2 presents the use of this AR interface within an industrial application. This case study is part of the COLLABORATE European project[2] targeting the use of cobots in industrial applications. This case study involves the float of an aircraft to be riveted in collaboration with the operator. The compartments of the aircraft float require a careful plan of the robot movements in order to reach robot configurations that are convenient for collaborative riveting and that do not collide with the aircraft structure during the motion between any two consecutive rivets. Here, the developed AR interface is used to let the operator program and monitor the robot motion plans, either from a remote location (i.e., without access to the robot and aircraft float, as depicted in Figure 3.2), or onsite the industry without the robot (i.e., virtual robot superposed to the real aircraft float).

[2]https://collaborate-project.eu/

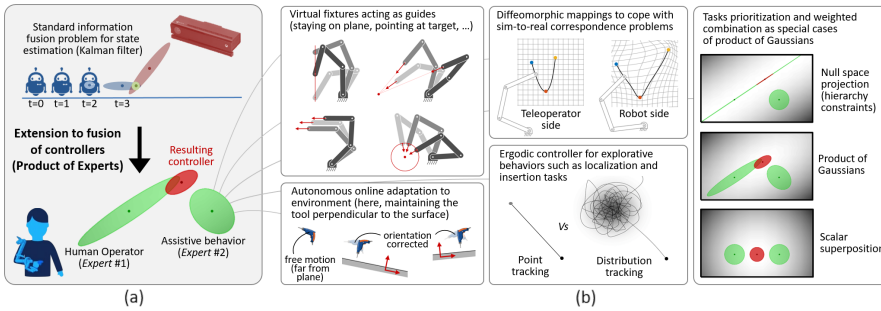


Figure 3.3: Examples of combination structures (see main text for details).

3.4 COMBINATION STRUCTURES

The term *movement primitives* refers to an organization of continuous movements in the form of a superposition in parallel and in series of simpler signals, which can be viewed as "building blocks" to create more complex movements. This principle of superposing high-level "bricks of motion" from a dictionary, coined in the context of motor control [11], remains valid for a wide range of continuous time signals (for both analysis and synthesis). In particular, this notion can be extended to *behavior primitives*, which form a richer set of assistive behaviors. These behavior primitives correspond to controllers that are either myopic or anticipative, with either time-independent or time-dependent formulations, see Figure 3.3 for an overview.

The combination of behavior primitives can be formalized as an information fusion problem in which several sources of information can be modeled as probability distributions [12]. This results in a *product of experts* (PoE) modeling a probability distribution by combining the output from several simpler distributions [13]. The core idea is to combine several distributions (called experts) by multiplying their density functions. This allows each expert to make decisions on the basis of a few dimensions, without having to cover the full dimensionality of a problem. Illustratively, a PoE corresponds to an "and" operation, which contrasts with a mixture model that corresponds to an "or" operation (by combining several probability distributions as a weighted sum of their density functions). Thus, each component in a PoE represents a soft constraint. For an event to be likely under a product model, all constraints must be (approximately) satisfied. In contrast, in a mixture model, an event is likely if it matches (approximately) with any single expert.

With Gaussian distributions, the fusion problem simplifies to a *product of Gaussians* (PoG), which can be solved analytically, where the distributions can either represent robot commands at the current time step (myopic control system), or trajectory distributions in the control space (anticipative planning system) [14].

State estimation is an example of problems classically solved as an information fusion problem, resulting in a product of Gaussians that takes into account uncertainty in motion and sensor(s) models (a well-known example is the Kalman filter), see Figure 3.3-*(a)*. A control problem in a collaborative industrial task can be treated in

a similar mathematical framework at the control level. This approach allow multiple behavior primitives to be combined in parallel (including prioritization with nullspace projection structures, see [12] for details). In this framework, each operator (human or artificial) contributes to the task by taking into account the (co)variations allowed in a task and the uncertainty of the expertise. In essence, this shared control strategy has links with products of Gaussians and Riemannian motion approaches, but the product of experts provides a more general formalism (incl. non-Gaussian distributions), and allows the same strategy to be considered for both shared perception and shared control paradigms.

For industrial applications, this approach allows the combination of different controllers, which can be learned separately or altogether (by variational inference, see [12] for details). With this formulation, a robot can counteract perturbations that have an impact on the fulfillment of the task, while ignoring other perturbations. This approach relates to research in biomechanics and motor control, with formulations including the principle of synergies [15], *minimal intervention principles* [16], *uncontrolled manifolds* [17] or *optimal feedback control* [18]. This formulation also opens the road to a rich set of challenges that can be investigated in manipulation and assembly with either autonomous robots or cobots, including bimanual manipulation skills, the handling of both force and position information, as well as the simultaneous handling of multiple objectives (including physical compliance, manipulability, safety, and task hierarchies). Figure 3.3-*(b)* presents examples of assistive behaviors in the context of manipulation and assembly skills.

Next, the PoE/PoG principle is illustrated with the problem of fusing information from multiple coordinate systems.

3.4.1 Task-parameterized movements

To facilitate the acquisition of manipulation skills, *task-parameterized models* can be exploited to take into account that motions typically relate to objects, tools or landmarks in the robot's workspace, see [19] for a review and [20] for an industrial case study.

The approach consists of encoding a movement in multiple coordinate systems (namely, from the perspectives of different objects), in the form of trajectory distributions. In a new situation (i.e., for new object locations), the reproduction problem corresponds to a fusion problem, where the variations in the different coordinate systems are exploited to generate a movement reference tracked with variable gains, providing the robot with a variable impedance behavior that automatically adapts to the precision required in the different phases of the task. For example, in a pick-and-place task, the robot will be stiff if the object needs to be reached/dropped in a precise way, and will remain compliant in the other parts of the task.

For user assistance in industrial operations, this approach can be used for local adaptation to objects/landmarks in the robot's environment, such as automatically correcting the orientation of a tool held by the operator to ensure that it remains perpendicular to a surface, see Fig. 3.3-*(c)*.

Figure 3.4 presents an example of application for remote valve turning operations, developed in the context of the DexROV European project [22]. In this example, a

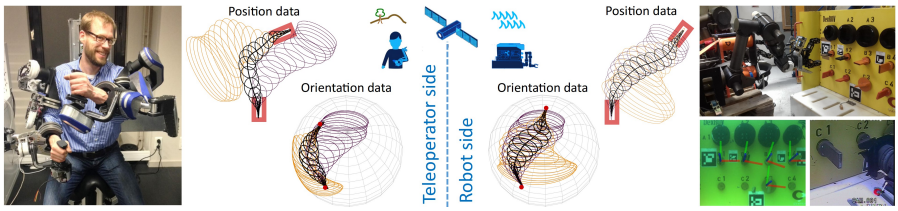


Figure 3.4: Task-parameterized movement models used within an offshore oil field application involving valves to be reached and turned remotely by a teleoperator. The purple and orange ellipsoids depict two GMMs, representing uncertain trajectories with respect to two different frames of reference (red U shapes for position data, and red points on the spheres for orientation data). The black ellipsoids represent the final trajectory and its uncertainty, obtained by fusing the trajectories of the two different frames of reference through products of Gaussians. Although the red U shapes and red points are not the same on the teleoperator side and on the robot side, the retrieved paths on the two sides can very quickly adapt to these different situations (see main text for details). Figure adapted from [21].

task-parameterized Gaussian mixture model (TP-GMM) approach is used within an optimal control strategy to teleoperate the arms of a bimanual underwater robot from distance, with a teleoperator wearing an exoskeleton and visualizing a copy of the robot workspace in a virtual environment. Because of the long communication delays between the teleoperator and the robot, the locations of the objects or tools of interest are not the same on the teleoperator side and on the robot side. With the proposed parameterization that leverages the locations of objects and tools, this discrepancy can be handled by automatic adaptation of the movement representation to the position and orientation of the objects/tools, represented as coordinate systems.

The central part of Figure 3.4 depicts two coordinate systems. The two components in red correspond to the robot and to the valve to be turned, for both position and orientation data. The encoding of the movements with respect to the robot and to the valve are depicted in orange and in purple, respectively.

The movements relative to the valve and to the robot are encoded as two sets of Gaussians in the two coordinate systems, for both position and orientation. During teleoperation, the distributions are rotated and translated according to the current situation on the teleoperator side and on the robot side. Products of Gaussians are computed at each side to fuse these representations (depicted in black in the figure).

Movement are encoded in this way for both position $\mathbb{R}^3$ and orientation $\mathcal{S}^3$ data (in Fig.3.4, a representation with $\mathbb{R}^2$ and $\mathcal{S}^2$ is shown as an illustration). For position, the retrieved path in black (and the associated covariances) corresponds to a movement going from the robot to the valve (represented as red U shapes), by taking into account how these different coordinate systems are oriented. We can see that the approaching phase is perpendicular to the coordinate system to properly reach the valve. For orientation, the retrieved path shows how the orientation of the endeffector changes with time. At the beginning of the motion, this orientation relates to the

orientation of the robot, while at the end, the orientation of the endeffector matches the orientation of the valve.

This task-parameterized modeling approach was successfully tested in field trials in the Mediterranean Sea offshore of Marseille, where 7 extended dives in 4 different sites (8m, 30m, 48m and 100m water depths) were performed with the underwater robot while being connected via satellite to the teleoperation center in Brussels, see [22] for an overview of the experiment.

The task-parameterized principle can be used as a first starting point to study the fusion of more elaborated coordinate systems. In Figure 3.4, simple coordinate systems centered on several points of interest were considered, but the approach could be extended to a richer set of behaviors, including a wider range of coordinate systems that take into account symmetries such as cylindrical and spherical coordinate systems [23] and nullspace projection structures [24].

In the next section, I will show that the description of movements as trajectory distributions can be extended to control problems by adopting an optimal control formulation with costs expressed as simple quadratic error terms.

3.4.2 Probabilistic optimal control

Skill acquisition encompasses a broad spectrum of learning approaches, from action-level *imitation* to goal-level *emulation*, where the mutual benefits of these two learning strategies have been highlighted in animals and humans studies [25].

While *emulation* requires higher cognitive skills to infer the intended goals behind a set of actions, *imitation* allows to acquire skills in a simple but limited way, by directly mimicking the movements and actions shown to the learner. Because of its higher level nature, *emulation* can easily be misinterpreted as being the only useful mechanism. Studies such as [25] show that the two are complementary, in the sense that *emulation* allows greater generalization (i.e., the learner understands the purpose of the task and can reproduce it in possibly different ways), while imitation allows to acquire tasks fast, even if all underlying goals are not fully understood (*"copy all, refine later"* strategy).

Similarly, in the engineering world, research in optimal control and reinforcement learning (RL) is tightly connected to the *emulation* approach, while research in movement primitives is tightly connected to the *imitation* approach. The mutual benefit can be illustrated by emulation making the system more generalizable, while imitation/mimicking can improve the learning rate, see e.g. [5, 26, 27].

I will show next that optimal control provides a formulation based on a cost that implements the goal-level nature of *emulation*, while control primitives provide a formulation based on basis functions that implements the action-level nature of *imitation*.

3.4.2.1 Linear quadratic tracking (LQT)

Linear quadratic tracking (LQT) and *linear quadratic regulation* (LQR) are the simplest forms of optimal control that trade off tracking and control costs expressed as quadratic terms over a time horizon, with the evolution of the state described in a

linear form. The LQT problem is formulated as the minimization of the cost

$$\begin{aligned} c &= (\boldsymbol{\mu}_T - \boldsymbol{x}_T)^\top \boldsymbol{Q}_T (\boldsymbol{\mu}_T - \boldsymbol{x}_T) + \sum_{t=1}^{T-1} (\boldsymbol{\mu}_t - \boldsymbol{x}_t)^\top \boldsymbol{Q}_t (\boldsymbol{\mu}_t - \boldsymbol{x}_t) + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t \\ &= (\boldsymbol{\mu} - \boldsymbol{x})^\top \boldsymbol{Q} (\boldsymbol{\mu} - \boldsymbol{x}) + \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u}, \end{aligned} \tag{3.1}$$

with $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_T^\top\right]^\top$ the evolution of the state variables, $\boldsymbol{u} = \left[\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \ldots, \boldsymbol{u}_{T-1}^\top\right]^\top$ the evolution of the control commands, and $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^\top, \boldsymbol{\mu}_2^\top, \ldots, \boldsymbol{\mu}_T^\top\right]^\top$ the evolution of the tracking targets. $\boldsymbol{Q} = \text{blockdiag}(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_T)$ represents the evolution of the precision matrices $\boldsymbol{Q}_t$, and $\boldsymbol{R} = \text{blockdiag}(\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_{T-1})$ represents the evolution of the control weight matrices $\boldsymbol{R}_t$.

The evolution of the system is linear, described by $\boldsymbol{x}_{t+1} = \boldsymbol{A}_t \boldsymbol{x}_t + \boldsymbol{B}_t \boldsymbol{u}_t$, yielding $\boldsymbol{x} = \boldsymbol{S}_x \boldsymbol{x}_1 + \boldsymbol{S}_u \boldsymbol{u}$ at trajectory level, with

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A}_1 \\ \boldsymbol{A}_2\boldsymbol{A}_1 \\ \vdots \\ \prod_{t=1}^{T-1} \boldsymbol{A}_{T-t} \end{bmatrix}}_{\boldsymbol{S}_x} \boldsymbol{x}_1 + \underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_2\boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\prod_{t=1}^{T-2} \boldsymbol{A}_{T-t}\right)\boldsymbol{B}_1 & \left(\prod_{t=1}^{T-3} \boldsymbol{A}_{T-t}\right)\boldsymbol{B}_2 & \cdots & \boldsymbol{B}_{T-1} \end{bmatrix}}_{\boldsymbol{S}_u} \underbrace{\begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{T-1} \end{bmatrix}}_{\boldsymbol{u}}. \tag{3.2}$$

The above problem can be solved either with a batch formulation or a recursive formulation. The batch formulation computes open loop control commands, which can be used to solve a planning problem or to simulate the resulting trajectories in control space and in state space. The recursive formulation computes a controller with adaptive feedback gains that can, to a certain extent, handle external perturbations.

Batch formulation

With open loop control commands organized as a vector $\boldsymbol{u}$, the minimization of (3.1) subject to $\boldsymbol{x} = \boldsymbol{S}_x \boldsymbol{x}_1 + \boldsymbol{S}_u \boldsymbol{u}$ provides an analytic result given by

$$\hat{\boldsymbol{u}} = \left(\boldsymbol{S}_u^\top \boldsymbol{Q} \boldsymbol{S}_u + \boldsymbol{R}\right)^{-1} \boldsymbol{S}_u^\top \boldsymbol{Q} \left(\boldsymbol{\mu} - \boldsymbol{S}_x \boldsymbol{x}_1\right). \tag{3.3}$$

The residuals of this least squares solution provides information about the uncertainty of this estimate in the form of a full covariance matrix (at control trajectory level)

$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{u}} = \left(\boldsymbol{S}_u^\top \boldsymbol{Q} \boldsymbol{S}_u + \boldsymbol{R}\right)^{-1}, \tag{3.4}$$

which provides a probabilistic interpretation of LQT. The resulting Gaussian distribution can then be projected back to a state trajectory space by exploiting the linear relationship between $\boldsymbol{u}$ and $\boldsymbol{x}$ in (3.2), so that a trajectory distribution in control space can be projected back to a trajectory distribution in state space. All the operations are analytic and only exploit basic linear algebra, see [14, 28] for details and [29] for examples of codes.

Figure 3.5-*left* shows that a LQT cost is composed of quadratic terms, whose solution corresponds to a product of Gaussians in control trajectory space, where

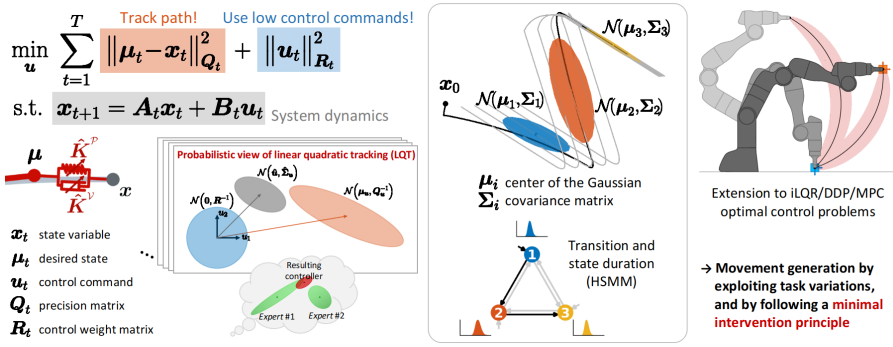


Figure 3.5: Probabilistic optimal control (see main text for details).

one Gaussian is centered on zero for all time steps (namely, staying still minimizes the control cost), and the other Gaussian corresponds to the swift tracking of a given reference. The product of Gaussians fuses the two constraints in control trajectory space. With this probabilistic interpretation, the batch formulation can be used to create bridges between learning and control. Figure 3.5-*center* shows an implementation example in learning from demonstration, where a set of demonstrations is encoded in a *hidden semi-Markov model* (HSMM) [30], providing an LQT cost function with the reference given by a stepwise reference built from the sequence of Gaussian centers and a full precision matrix(built from the sequence of Gaussian covariances. This approach allows the observed (co)variations of a task to generate a controller with anticipation capability. Here, 5 demonstrations are encoded as 3 Gaussians. Without anticipation, the reproduced movement would move from one center to the other, without reproducing the Z-shape of the demonstrations. Instead, LQT correctly reproduces the Z shape from any initial point $\boldsymbol{x}_0$ (see the trajectory produced by the controller in black), by exploiting the coordination patterns extracted from the demonstrations.

Figure 3.5-*right* highlights that the approach can be extended to *model predictive control* (MPC), *iterative LQR* (iLQR) and *differential dynamic programming* (DDP), whose solutions need this time either to be interpreted within an iterative process or updated at consecutive time steps, see [31] for details.

Recursive formulation

The problem of tracking a reference signal $\{\boldsymbol{\mu}_t\}_{t=1}^{T}$ can be recast as a regulation problem by considering a dynamical system $\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{B}}_t\boldsymbol{u}_t$ with an augmented state

$$\underbrace{\begin{bmatrix}\boldsymbol{x}_{t+1}\\ 1\end{bmatrix}}_{\tilde{\boldsymbol{x}}_{t+1}} = \underbrace{\begin{bmatrix}\boldsymbol{A}_t & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{bmatrix}}_{\tilde{\boldsymbol{A}}_t}\underbrace{\begin{bmatrix}\boldsymbol{x}_t\\ 1\end{bmatrix}}_{\tilde{\boldsymbol{x}}_t} + \underbrace{\begin{bmatrix}\boldsymbol{B}_t\\ 0\end{bmatrix}}_{\tilde{\boldsymbol{B}}_t}\boldsymbol{u}_t,$$

and augmented tracking precision matrices

$$\tilde{\boldsymbol{Q}}_t = \begin{bmatrix} \boldsymbol{Q}_t^{-1} + \boldsymbol{\mu}_t \boldsymbol{\mu}_t^\top & \boldsymbol{\mu}_t \\ \boldsymbol{\mu}_t^\top & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{\mu}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_t & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{\mu}_t \\ \boldsymbol{0} & 1 \end{bmatrix},$$

where the tilde notation $\tilde{\cdot}$ is used to refer to an augmented variable.

This augmented form is used to express the cost in (3.1) as

$$c = \tilde{\boldsymbol{x}}_T^\top \tilde{\boldsymbol{Q}}_T \tilde{\boldsymbol{x}}_T + \sum_{t=1}^{T-1} \tilde{\boldsymbol{x}}_t^\top \tilde{\boldsymbol{Q}}_t \tilde{\boldsymbol{x}}_t + \boldsymbol{u}_t^\top \boldsymbol{R}_t \boldsymbol{u}_t, \tag{3.5}$$

which has the same form as a standard LQR problem. For a tracking problem, the resulting optimal control policy takes the form

$$\hat{\boldsymbol{u}}_t = -\tilde{\boldsymbol{K}}_t \, \tilde{\boldsymbol{x}}_t = \boldsymbol{K}_t \, (\boldsymbol{\mu}_t - \boldsymbol{x}_t) + \boldsymbol{u}_t^{\mathrm{ff}}, \tag{3.6}$$

characterized by a feedback gain matrix $\boldsymbol{K}_t$ extracted from $\tilde{\boldsymbol{K}}_t = \begin{bmatrix} \boldsymbol{K}_t, \boldsymbol{k}_t \end{bmatrix}$, and a feedforward term $\boldsymbol{u}_t^{\mathrm{ff}} = -\boldsymbol{k}_t - \boldsymbol{K}_t \boldsymbol{\mu}_t$ depending on $\boldsymbol{\mu}_t$.

The augmented feedback gain matrices in (3.6) are computed with

$$\tilde{\boldsymbol{K}}_t = (\tilde{\boldsymbol{B}}_t^\top \boldsymbol{P}_{t+1} \tilde{\boldsymbol{B}}_t + \boldsymbol{R}_t)^{-1} \tilde{\boldsymbol{B}}_t^\top \boldsymbol{P}_{t+1} \tilde{\boldsymbol{A}}_t, \tag{3.7}$$

where $\boldsymbol{P}_t$ is evaluated recursively from $t = T-1$ to $t = 1$ as

$$\boldsymbol{P}_t = \tilde{\boldsymbol{A}}_t^\top \boldsymbol{P}_{t+1} \tilde{\boldsymbol{A}}_t + \tilde{\boldsymbol{Q}}_t - \tilde{\boldsymbol{A}}_t^\top \boldsymbol{P}_{t+1} \tilde{\boldsymbol{B}}_t (\tilde{\boldsymbol{B}}_t^\top \boldsymbol{P}_{t+1} \tilde{\boldsymbol{B}}_t + \boldsymbol{R}_t)^{-1} \tilde{\boldsymbol{B}}_t^\top \boldsymbol{P}_{t+1} \tilde{\boldsymbol{A}}_t,$$

starting from $\boldsymbol{P}_T = \tilde{\boldsymbol{Q}}_T$, see [29, 28] for details.

After all feedback gain matrices $\tilde{\boldsymbol{K}}_t$ have been computed by backward recursion, a forward recursion is used to compute the evolution of the state with the linear system $\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{A}}_t \tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{B}}_t \boldsymbol{u}_t$, starting from $\tilde{\boldsymbol{x}}_1$, by using the feedback control policy $\boldsymbol{u}_t = -\tilde{\boldsymbol{K}}_t \tilde{\boldsymbol{x}}_t$. Compared to a batch formulation, this recursive formulation computes a controller that can, to a certain extent, handle external perturbations.

Least squares computation of the recursive formulation (LQT-LS)

We will now show that an intermediary result exists between a batch and a recursive formulation, and that this formulation is helpful to link learning and control algorithms. First, by using augmented vectors to describe trajectories and writing $\boldsymbol{u} = -\boldsymbol{F} \tilde{\boldsymbol{x}}_1$, we can redefine the costw in (3.1) and (3.5) as the optimization problem

$$\min_{\boldsymbol{F}} \; \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{x}} + (-\boldsymbol{F} \tilde{\boldsymbol{x}}_1)^\top \boldsymbol{R} \, (-\boldsymbol{F} \tilde{\boldsymbol{x}}_1), \quad \text{s.t.} \quad \tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{S}}_{\boldsymbol{x}} - \tilde{\boldsymbol{S}}_{\boldsymbol{u}} \boldsymbol{F}) \tilde{\boldsymbol{x}}_1,$$

whose least squares solution is

$$\boldsymbol{F} = \left(\tilde{\boldsymbol{S}}_{\boldsymbol{u}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{S}}_{\boldsymbol{u}} + \boldsymbol{R} \right)^{-1} \tilde{\boldsymbol{S}}_{\boldsymbol{u}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{S}}_{\boldsymbol{x}}, \tag{3.8}$$

which can be used to iteratively reconstruct regulation gains $\tilde{\boldsymbol{K}}_t$, by starting from $\tilde{\boldsymbol{K}}_1 = \boldsymbol{F}_1$, $\boldsymbol{P}_1 = \boldsymbol{I}$. This is done by computing recursively

$$\boldsymbol{P}_t = \boldsymbol{P}_{t-1} (\tilde{\boldsymbol{A}}_{t-1} - \tilde{\boldsymbol{B}}_{t-1} \tilde{\boldsymbol{K}}_{t-1})^{-1}, \qquad \tilde{\boldsymbol{K}}_t = \boldsymbol{F}_t \, \boldsymbol{P}_t, \tag{3.9}$$

which can then be used in a feedback controller on an augmented state as in (3.6).

This least squares formulation of recursive LQT, that we will call LQT-LS, yields the same controller (3.6) as with the standard recursive formulation. However, the linear form in (3.8) used by LQT-LS has several advantages. First, it allows the use of full precision matrices $\tilde{\boldsymbol{Q}}$ that can encode correlation constraints within a given state but also between time steps, see [32] for an example. It also allows LQT-LS to be extended to the use of *control primitives*, which we discuss next.

LQT-LS with control primitives

A univariate control trajectory $\boldsymbol{u}^{\text{1D}} \in \mathbb{R}^T$ of T control commands can be represented as a weighted sum of K basis functions with

$$\boldsymbol{u}^{\text{1D}} = \sum_{k=1}^{K} \boldsymbol{\phi}_k \, w_k^{\text{1D}} = \boldsymbol{\phi} \, \boldsymbol{w}^{\text{1D}}, \tag{3.10}$$

where the matrix $\boldsymbol{\phi}$ is a horizontal concatenation of univariate basis functions.

A multivariate control trajectory $\boldsymbol{u} \in \mathbb{R}^{DT}$ of T control commands of dimension D can similarly be computed as

$$\boldsymbol{u} = \sum_{k=1}^{K} \boldsymbol{\Psi}_k w_k = \boldsymbol{\Psi}\boldsymbol{w}, \text{ with } \boldsymbol{\Psi} = \boldsymbol{\phi} \otimes \boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}\phi_{1,1} & \boldsymbol{C}\phi_{2,1} & \cdots & \boldsymbol{C}\phi_{K,1} \\ \boldsymbol{C}\phi_{1,2} & \boldsymbol{C}\phi_{2,2} & \cdots & \boldsymbol{C}\phi_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}\phi_{1,T} & \boldsymbol{C}\phi_{2,T} & \cdots & \boldsymbol{C}\phi_{K,T} \end{bmatrix}, \tag{3.11}$$

where $\otimes$ is the Kronecker product operator and $\boldsymbol{C}$ is a coordination matrix that can be set to an identity matrix $\boldsymbol{I}_D$ for the generic case of control variables with independent basis functions for each dimension $d \in \{1, \dots, D\}$.

Equation (3.11) aims at encoding the control commands as a weighted superposition of simpler commands. In an optimal control problem, the encoding of control commands as a set of weights has two advantages: 1) it reduces computation time and storage by working in a subspace of reduced dimensionality; and 2) it acts as a regularization approach by denoising the signal while capturing the essential aspects of a controlled movement. Such a representation can be illustrated as the use of basic building blocks that can be assembled in different manners to form more elaborated movements, often referred to as *movement primitives* (MP).

If we assume that the control commands profile $\boldsymbol{u}$ is composed of *control primitives* (CP) with $\boldsymbol{u} = \boldsymbol{\Psi}\boldsymbol{w}$, the LQT objective in (3.1) becomes

$$\min_{\boldsymbol{w}} (\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{Q} (\boldsymbol{x} - \boldsymbol{\mu}) \; + \; \boldsymbol{w}^\top \boldsymbol{\Psi}^\top \boldsymbol{R} \boldsymbol{\Psi} \boldsymbol{w}, \quad \text{s.t.} \quad \boldsymbol{x} = \boldsymbol{S}_x \boldsymbol{x}_1 + \boldsymbol{S}_u \boldsymbol{\Psi} \boldsymbol{w},$$

with a solution given by

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{\Psi}^\top \boldsymbol{S}_u^\top \boldsymbol{Q} \boldsymbol{S}_u \boldsymbol{\Psi} + \boldsymbol{\Psi}^\top \boldsymbol{R} \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{S}_u^\top \boldsymbol{Q} \left(\boldsymbol{\mu} - \boldsymbol{S}_x \boldsymbol{x}_1\right),$$

which is used to compute the trajectory $\hat{\boldsymbol{u}} = \boldsymbol{\Psi}\hat{\boldsymbol{w}}$ in control space, corresponding to the list of control commands organized in vector form. Note that in practice, $\boldsymbol{R}$ can be set to zero because the regularization role of $\boldsymbol{R}$ in the original problem formulation can be considered as redundant to the use of sparse basis functions.

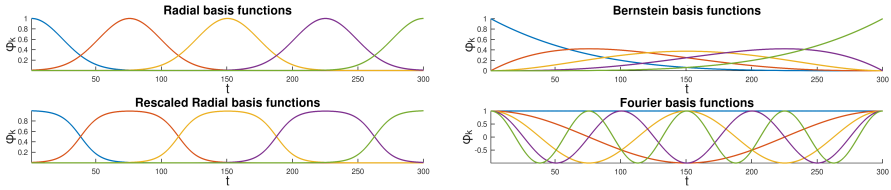


Figure 3.6: Examples of basis functions for $K = 5$ and $T = 300$ (see main text for details). Figure adapted from [35].

Similarly to (3.4), the residuals of the least squares solution provides a probabilistic representation of LQT with control primitives. Note that as seen before, the trajectory in control space can be converted to a trajectory in state space with $\hat{\boldsymbol{x}} = \boldsymbol{S}_{\boldsymbol{x}}\boldsymbol{x}_1 + \boldsymbol{S}_{\boldsymbol{u}}\hat{\boldsymbol{u}}$. Thus, since all operations are linear, a covariance matrix on $\boldsymbol{w}$ can be converted to a covariance on $\boldsymbol{u}$ and on $\boldsymbol{x}$. A similar linear transformation is used in the context of *probabilistic movement primitives* (ProMP) [33] to map a covariance matrix on $\boldsymbol{w}$ to a covariance on $\boldsymbol{x}$. LQT with control primitives provides a more general approach by considering both control and state spaces. It also has the advantage of reframing the method to the more general context of optimal control, thus providing various extension opportunities.

While radial basis functions (RBFs) are the most commonly used representation in movement primitives, other forms of basis functions can be considered, including stepwise control commands, Bernstein polynomials, B-splines or Fourier series, which can all be organized as a dictionary matrix $\boldsymbol{\Psi}$. These basis functions can also be optimized instead of predefined [34].

Figure 3.6 presents some examples, see [35] for details. The use of an exponential in radial basis functions (RBFs) make them infinitely differentiable. However, they typically require a bandwidth parameter to be set heuristically. Bernstein basis functions and Fourier basis functions do not require this type of hyperparameter to be set. Bernstein basis functions instead require the degree of the polynomial to be set, which can be selected based on the number of derivatives required. Their advantages are that they can be interpreted as control points (corresponding to Bézier curves) and that they sum to a regular constant signal. Fourier basis functions are infinitely differentiable and provide a spectral decomposition of the signal instead of a spatial decomposition, with an ordering of the basis functions corresponding to the resolution required for the reconstruction of the signal. They can be used for both periodic and discrete motion, by encoding a mirrored version of the original signal.

Next, we present an example of the proposed LQT-LS formulation with basis functions to implement the dynamical movement primitives (DMP) principle [36].

3.4.2.2 Dynamical movement primitives (DMP) formulated as LQT-LS with control primitives

The *dynamical movement primitives* (DMP) [36] approach proposes the reproduction of an observed movement by considering a controller composed of two parts: a closed-loop spring-damper system reaching the final point of the observed movement,

and an open-loop forcing term composed of the acceleration, velocity and position profiles of the observed movement. The evolution of this system is either driven by time or by a decay term. These two controllers are weighted so that the influence of the spring-damper system is progressively increased until it becomes the only active controller. In DMP, the forcing term is encoded with *radial basis functions* (RBFs) [37]. The spring-damper system parameters of the closed-loop system are defined heuristically, usually as a critically damped system, see [36] for details.

The LQT-LS approach presented in the previous section can be employed to implement the DMP principle, with a linear system defined as an integrator (e.g., second order integrator). The LQT cost can similarly request a target to be reached at the end of the movement, while following the observed acceleration/velocity/position profiles. The controller can be estimated either as open-loop control commands (3.3) (providing a probabilistic interpretation), or as a closed-loop controller (3.6), by encoding $\boldsymbol{F}$ in (3.8) as radial basis functions.

In the latter case, the matrix $\boldsymbol{F}$ in (3.8) is estimated by using control primitives and an augmented state space formulation, namely

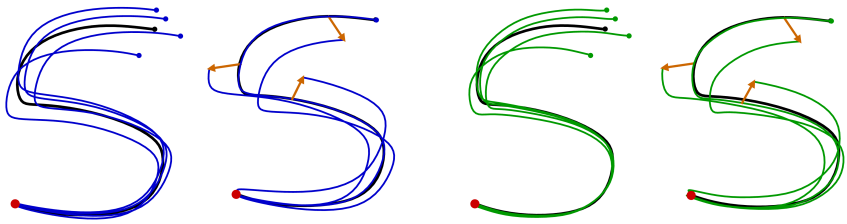
$$\hat{\boldsymbol{W}} = \left(\boldsymbol{\Psi}^\top \tilde{\boldsymbol{S}}_{\boldsymbol{u}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{S}}_{\boldsymbol{u}} \boldsymbol{\Psi} + \boldsymbol{\Psi}^\top \boldsymbol{R} \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\top \tilde{\boldsymbol{S}}_{\boldsymbol{u}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{S}}_{\boldsymbol{x}}, \qquad \boldsymbol{F} = \boldsymbol{\Psi} \hat{\boldsymbol{W}},$$

which is used to compute feedback gains $\tilde{\boldsymbol{K}}_t$ on the augmented state with (3.9).

The resulting feedback controller $\hat{\boldsymbol{u}}_t = -\tilde{\boldsymbol{K}}_t \tilde{\boldsymbol{x}}_t$ tracks the observed profiles while smoothly reaching the desired goal at the end of the movement, with a smooth transition between the two control objectives. The main difference with DMP is that the smooth transition between the two behaviors is directly optimized by the system, together with the parameters of the feedback controller that are also automatically optimized (incl. stiffness and damping ratio).

The proposed LQT-LS formulation of DMP allows the resulting controller to be formalized as a cost function, which allows the approach to be combined more easily with other optimal control strategies. First, the proposed approach can freely modulate the way in which the system reacts to perturbations, by weighting how important it is to follow the demonstrated position, velocity or acceleration profiles, respectively (or higher order derivatives, if desired). For example, setting a higher cost on the position profile will make the feedback controller more rapidly move back to the path instead of following the velocity and acceleration profiles. Moreover, the approach can be extended to multiple viapoints without any modification. It also allows multiple demonstrations of a movement to be used to estimate the feedback controller, which provides a probabilistic formulation of DMP. As a result, the (co)variations in the demonstrations are exploited to provide a minimal intervention control strategy that will selectively reject perturbations, based on the impact that these perturbations have on the task to achieve. This is effectively attained by automatically regulating the gains in accordance to the variations in the demonstrations, with low gains in parts of the movement allowing variations, and higher gains for parts of the movement that are invariant in the demonstrations.

Figure 3.7 presents an example by reproducing a demonstrated S-shaped trajectory, see [29] for the corresponding code example.



(a) Proposed LQT-LS (b) Standard DMP

Figure 3.7: Least squares solution of recursive linear quadratic tracking (LQT-LS) by using control primitives. To imitate a dynamical movement primitive (DMP), the cost of LQT-LS can be set to track both the demonstrated motion (in black) and the endpoint (in red). In this example, LQT-LS and DMP both use 15 radial basis functions. The two models (LQT-LS in blue, DMP in green) are tested with the same set of four random initial states and four random perturbations (orange arrow) occurring during the motion, showing the similarities of the two results.

3.5 GEOMETRIC STRUCTURES

In the previous sections, we have discussed learning and control problems in standard Euclidean spaces. In this section, we extend the proposed challenges by adopting a geometric perspective.

Data encountered in robotics are characterized by simple but varied geometries. These geometric structures are often underexploited in learning, planning, control and perception. Riemannian geometry provides a principled and simple way to extend algorithms initially developed for Euclidean data to other manifolds, by efficiently taking into account prior geometric knowledge about these manifolds [38, 39]. In particular, Riemannian manifolds can be connected to a wide range of problems in robotics, including problems relying on Gaussian distributions [21]. This includes techniques requiring uncertainty and statistical modeling to be computed on structured non-Euclidean data.

Examples of standard problems in robotics are depicted in Figure 3.8-*(a)* from a geometric perspective. Riemannian manifolds can be applied to various forms of data, as illustrated in Figure 3.9. Such data range from joint angles in revolving articulations [40], rigid body motions [41, 42], unit quaternions to represent orientations [43], and symmetric positive definite matrices, which can represent sensory data processed as spatial covariances [44], inertia [45, 46], stiffness/manipulability ellipsoids [47], as well as metrics used in the context of similarity measures.

By using Riemannian manifolds, data of various forms can be treated in a unified manner, with the advantage that existing models and algorithms initially developed for Euclidean data can be extended to a wider range of data structures. It can for example be used to revisit constrained optimization problems formulated in Euclidean

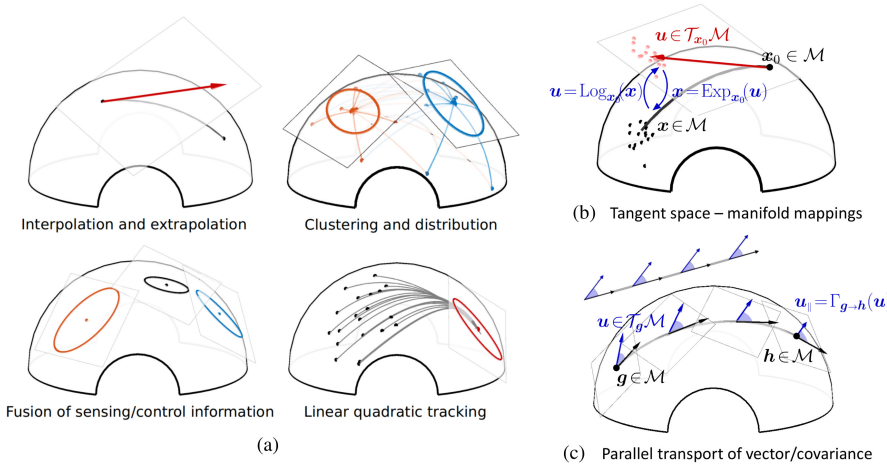


Figure 3.8: (a) Riemannian geometry can solve a variety of problems involving distributions and uncertainties. Applications in robotics using Riemannian manifolds principally rely on two principles: (b) exponential/logarithmic mapping between tangent space and manifold; and (c) parallel transport of a vector along a geodesic (see main text for details).

space, with a treatment as unconstrained problems inherently taking into account the geometry of the data.

A d-dimensional *Riemannian manifold* $\mathcal{M}$ is a smooth and differentiable manifold, which locally behaves like the Euclidean space $\mathbb{R}^d$, and which is equipped with a positive definite metric tensor to measure distances of points on the manifold. For each point $\boldsymbol{p} \in \mathcal{M}$, there exists a tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ that locally linearizes the manifold. The affine connection, computed from the metric, is a differential operator that provides a way to compute geodesics and to transport vectors on tangent spaces along any smooth curves on the manifold [38, 39]. The Cartesian product of two Riemannian manifolds is also a Riemannian manifold, which allows joint distributions to be constructed with any combination of Riemannian manifolds.

Two basic notions of Riemannian geometry are crucial for learning and control applications, which are illustrated in Figure 3.8-*(b-c)* and described below.

Geodesics: The minimum length curves between two points on a Riemannian manifold are called geodesics. The exponential map $\mathrm{Exp}_{\boldsymbol{x}_0} : \mathcal{T}_{\boldsymbol{x}_0}\mathcal{M} \to \mathcal{M}$ 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 $\mathrm{Log}_{\boldsymbol{x}_0} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$. Figure 3.8-*(b)* depicts these mapping functions.

Parallel transport: Parallel transport $\Gamma_{\boldsymbol{g} \to \boldsymbol{h}} : \mathcal{T}_{\boldsymbol{g}}\mathcal{M} \to \mathcal{T}_{\boldsymbol{h}}\mathcal{M}$ moves vectors between tangent spaces such that the inner product between two vectors in a tangent space is

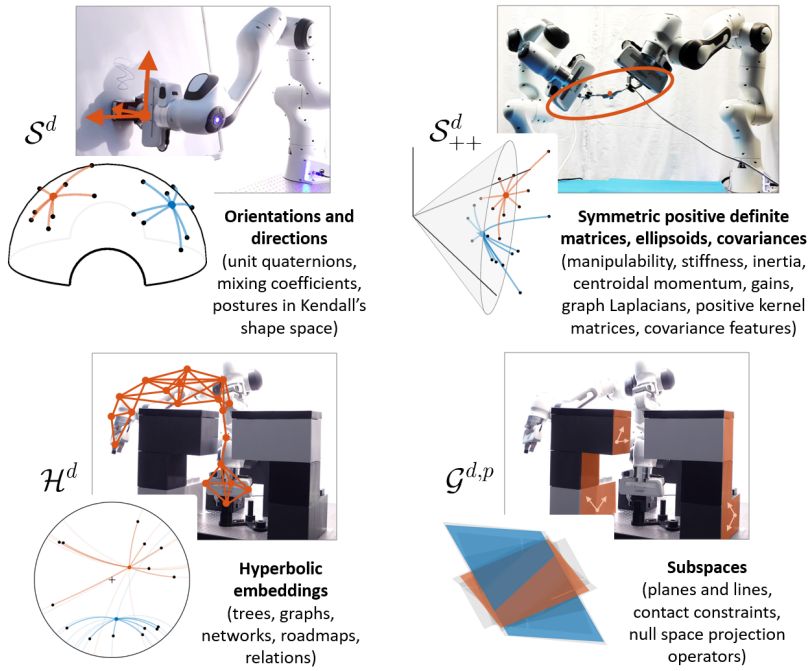


Figure 3.9: Several geometric structures are relevant for industrial robotics applications. $\mathcal{S}^3$ can be used to represent the orientation of robot endeffectors (unit quaternions). $\mathcal{S}^6_{++}$ can be used to represent manipulability ellipsoids (manipulability capability in translation and rotation), corresponding to symmetric positive definite (SPD) matrix manifolds. $\mathcal{H}^d$ can be used to represent trees, graphs and roadmaps. $\mathcal{G}^{d,p}$ can be used to represent subspaces (planes, nullspaces, projection operators).

conserved. It employs the notion of connection, defining how to associate vectors between infinitesimally close tangent spaces. This connection allows the smooth transport of a vector from one tangent space to another.

Figure 3.8-*(c)* depicts the parallel transport operation. The flat surfaces show the coordinate systems of several tangent spaces along the geodesic. The black vectors represent the directions of the geodesic in the tangent spaces. The blue vectors are transported from $\boldsymbol{g}$ to $\boldsymbol{h}$. Parallel transport allows a vector $\boldsymbol{u}$ in the tangent space of $\boldsymbol{g}$ to be transported to the tangent space of $\boldsymbol{h}$, by ensuring that the angle (i.e., inner product) between $\boldsymbol{u}$ and the direction of the geodesic (represented as black vectors) are conserved. At point $\boldsymbol{g}$, this direction is expressed as $\mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{h})$. This operation is crucial to combine information available at $\boldsymbol{g}$ with information available at $\boldsymbol{h}$, by taking into account the transformation of the coordinate systems along the geodesic (notice the rotation of the tangent spaces in Figure 3.8-*(c)*). In Euclidean space (top-left inset in *(c)*), parallel transport is simply the identity operator (a vector operation can be applied to any point without additional transformation).

By extension, a covariance matrix $\boldsymbol{\Sigma}$ can be transported by using a decomposition into eigenvectors, by transporting the eigenvectors, and by recomputing a covariance matrix with the transported eigenvectors. For many manifolds in robotics, this transport operation can be expressed as a linear mapping. The most common manifolds in robotics are homogeneous, providing simple analytic expressions for exponential/logarithmic mapping and parallel transport. For industrial robotics, these manifolds include orientations, ellipsoids, graphs and subspaces, see Figure 3.9.

Several approaches have been proposed to extend Gaussian distributions in Euclidean space to Riemannian manifolds. For robotics applications, a simple approach consists of estimating the mean of the Gaussian as a centroid on the manifold, and representing the dispersion of the data as a covariance defined in the tangent space of the centroid. Parameters estimation can be solved by a simple and fast Gauss–Newton iterative algorithm. Importantly, this geometric mean can be directly extended to weighted distances, which can be exploited for mixture modeling, fusion (product of Gaussians), regression (conditional distribution), movement primitives and planning problems, see Figure 3.8-*(a)* and [21] for a review.

The combination of statistics and Riemannian geometry offers many research opportunities, and can contribute to recent challenges in industrial robotics. Further work can be organized in two categories. Firstly, the field of industrial robotics is abundant of new research developments, due to the interdisciplinary aspect and to the richness of problems it involves (in particular, targeting cobots and bimanual manufacturing robots). The common factor in many of these developments is that they rely on some form of statistics and/or propagation of uncertainty. These models and algorithms are typically developed for standard Euclidean spaces, where an extension to Riemannian manifolds has several benefits to offer. Secondly, some Riemannian manifolds remain largely underexploited in robotics, despite the fact that some of them are mathematically well understood and characterized by simple closed-form expressions. Grassmann manifolds seem particularly promising to handle problems in robotics with high dimensional datapoints and only few training data, where subspaces are required in the computation to keep the most essential characteristics of the data. It is also promising for problems in which hierarchies are considered (such as inverse kinematics with kinematically redundant robots), because it provides a geometric interpretation of nullspace structures. Other Riemannian manifolds such as hyperbolic manifolds also seem propitious to bring a probabilistic treatment to tree-based structures, graphs, Toeplitz/Hankel matrices, autoregressive models and dynamical systems.

Next, we present an example exploiting Riemannian geometry for a robot application targeting skill transfer problems.

3.5.1 Learning, tracking and transfer of manipulability ellipsoids

Stiffness and manipulability ellipsoids are geometric descriptors that can be used to transfer manipulation skills to a robot. As these ellipsoids lie on *symmetric positive definite* (SPD) manifolds, Riemannian manifolds can be used to learn and reproduce these descriptors in a probabilistic manner [21, 47].

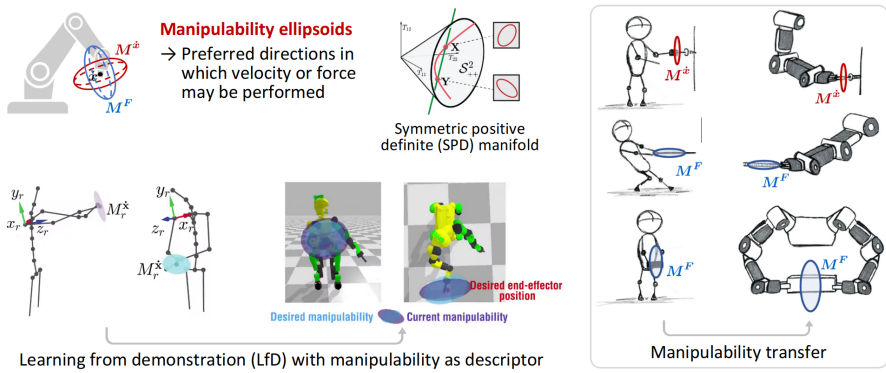


Figure 3.10: Manipulability ellipsoids used as task descriptors, exploited in the context of transferring skills from humans to robots of different morphologies, by using the industrial operation dataset from [49]. Figure adapted from [50].

Manipulability ellipsoids come in many different flavors due to the different types of Jacobian matrices that can be considered in robotics problems. They can be defined for different points of interest (end-effector, center of mass), for positions and forces (including the orientation/rotational part), or for different task priorities (nullspace structures). Manipulability can be described at either kinematic or dynamic levels (including inertia), for either open or closed linkages (e.g., for bimanual manipulation or in-hand manipulation), and for either fully actuated or underactuated systems [48]. This large set of manipulability ellipsoids provides rich descriptors to characterize robot skills for robots with legs and arms.

Manipulability ellipsoids are helpful descriptors to handle skills transfer problems involving dissimilar kinematic chains, such as transferring manipulation skills between humans and robots, or between two robots with different kinematic chains or capabilities. In such transfer problems, imitation at joint angles level is not possible due to the different structures, and imitation at endeffector(s) level is limited because it does not encapsulate postural information, which is often an essential aspect of the skill that needs to be transferred. Manipulability ellipsoids provide intermediate descriptors that allow postural information to be transferred indirectly, with the advantage that they allow different embodiments and capabilities to be considered in the skill transfer problem.

Figure 3.10 presents an example within the context of industrial manipulation tasks to be transferred from a human user to a bimanual quadruped robot (centaur robot), see [50] for the full description of the approach and experiments.

The manipulability ellipsoid $\boldsymbol{M}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})\boldsymbol{J}(\boldsymbol{x})^\top$ is a SPD matrix representing the manipulability at a given point on the kinematic chain defined by a forward kinematics function $\boldsymbol{f}(\boldsymbol{x})$, given the joint angle posture $\boldsymbol{x}$, where $\boldsymbol{J}(\boldsymbol{x})$ is the Jacobian of $\boldsymbol{f}(\boldsymbol{x})$. In robotics, the determinant of $\boldsymbol{M}(\boldsymbol{x})$ is most often employed [51]. It reduces the information to a scalar manipulability index indicating the volume of the ellipsoid, with the drawback of ignoring the specific shape of this ellipsoid. Such

a scalar descriptor can have limitations when used as a metric to avoid kinematic singularity, see e.g. [52, 53].

In [47], we showed that a geometric cost on manipulability can alternatively be defined with the geodesic distance

$$\begin{aligned} & c = \|\boldsymbol{A}\|_{\mathrm{F}}^2, \quad \text{with} \quad \boldsymbol{A} = \log\big(\boldsymbol{S}^{\frac{1}{2}} \boldsymbol{M}(\boldsymbol{x}) \boldsymbol{S}^{\frac{1}{2}}\big), \\ \Longleftrightarrow \quad & c = \mathrm{trace}(\boldsymbol{A}\boldsymbol{A}^\top) = \sum_i \mathrm{trace}(\boldsymbol{A}_i \boldsymbol{A}_i^\top) = \sum_i \boldsymbol{A}_i^\top \boldsymbol{A}_i = \mathrm{vec}(\boldsymbol{A})^\top \mathrm{vec}(\boldsymbol{A}), \end{aligned} \tag{3.12}$$

where $\boldsymbol{S}$ is the desired manipulability matrix to reach, $\log(\cdot)$ is the logarithm matrix function, and $\|\cdot\|_{\mathrm{F}}$ is a Frobenius norm. By exploiting the trace properties, (3.12) can be expressed in a quadratic form, where a vectorization operation $\mathrm{vec}(\boldsymbol{A})$ can be computed efficiently by exploiting the symmetry of the matrix $\boldsymbol{A}$. This is implemented by keeping only the lower half of the matrix, with the elements below the diagonal multiplied by a factor $\sqrt{2}$.

The derivatives of (3.12) with respect to the state $\boldsymbol{x}$ and control commands $\boldsymbol{u}$ can be computed numerically, by automatic differentiation, or by following an analytic approach as in [47]. The quadratic form proposed in (3.12) can be exploited to solve an optimal control problem in the form of iLQR, formulated as a Gauss–Newton optimization problem leveraging the Jacobian of $\mathrm{vec}(\boldsymbol{A})$, see [29] for a corresponding code example. Marić *et al.* demonstrated the advantages of a geometric cost as in (3.12) for planning problems, by comparing it with other widely used metrics such as the manipulability index and the dexterity index [53].

As mentioned in the above, manipulability can be defined at several points of interest, including endeffectors and centers of mass. It can also be defined at specific points on objects or tools held by the robot, allowing manipulability ellipsoids to be exploited as useful descriptors for object affordances. For example, for a robot holding a hammer, we can consider a manipulability ellipsoid at the head of a hammer. When the robot grasps the hammer, its endeffector is extended so that the head of the hammer becomes the extremity of the kinematic chain. In this situation, the different options that the robot has to grasp the hammer will have an impact on the resulting manipulability at the head of the hammer. Thus, grabbing the hammer at the extremity of the handle will improve the resulting manipulability. Such descriptors offer promises for learning and optimization in manufacturing environments, in order to let the robots automatically determine the correct ways to employ tools, including grasping points and body postures.

Optimal control with a geodesic cost on SPD manifolds can be used for manipulability ellipsoids, but it can also be extended to other descriptors represented as ellipsoids. In particular, stiffness, feedback gains, inertia, and centroidal momentum have similar structures. For the latter, the centroidal momentum ellipsoid quantifies the momentum generation ability of the robot [54], which is another useful descriptor for robot skills. Similarly to manipulability ellipsoids, it is constructed from Jacobians, which are in this case the centroidal momentum matrices mapping the joint angle velocities to the centroidal momentum, which sums over the individual link momenta after projecting each to the robot's center of mass. The proposed approach could also be extended to other forms of SPD matrices, such as kernel matrices used

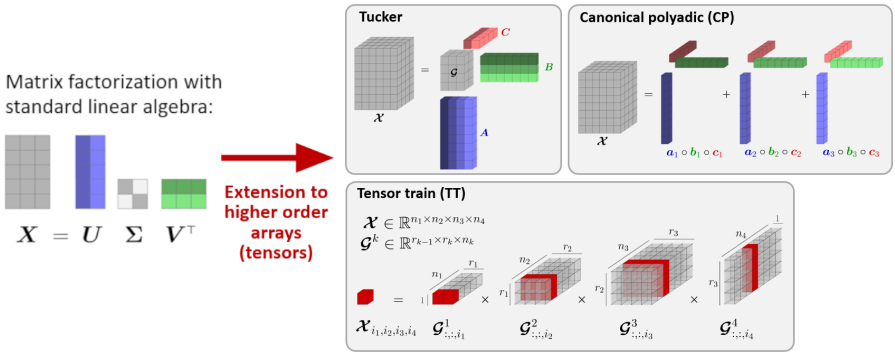


Figure 3.11: Tensor factorization structures that can be used to automatically extract essential information from multidimensional arrays of data.

in machine learning algorithms to compute similarities, or graph Laplacians (a matrix representation of a graph that can for example be used to construct a low dimensional embedding of a graph).

3.6 DATA STRUCTURES

Another type of structures that can be exploited in learning and control problems in robotics relates to the organization of data as multidimensional arrays (also called tensors). These data appear in various robotics tasks, either as the natural organization of sensory/motor data (tactile arrays, images, kinematic chains), as the result of standardized preprocessing steps (moving time windows, covariance features), as data in multiple coordinate systems, or in the form of basis functions decomposition.

Developed in the fields of *multilinear algebra* and *tensor methods*, tensor factorization techniques extend matrix factorization techniques such as singular value decomposition (SVD) to multilinear decomposition, without requiring the transformation of the tensors to vectors or matrices. Figure 3.11 presents a few examples of tensor factorization techniques. These techniques can be exploited to provide robots with the capability to learn tasks from only few human demonstrations [55] or from only few exploration trials [56] by effectively leveraging the multidimensional nature of the problem.

Next, we present an example of tensor factorization used in the context of industrial peg-in-hole insertion tasks.

3.6.1 Ergodic control for insertion tasks

A tracking problem in industrial robot application is conventionally characterized by a target point to reach, requiring a controller to be computed to reach this target. In ergodic control, instead of providing a single target point, a probability distribution is given to the robot, which must cover this distribution in an efficient way. *Ergodic control* thus consists of moving within a spatial distribution by spending time in

each part of the distribution in proportion to its density (illustratively, "tracking a distribution" instead of "tracking a point"). The resulting controller generates natural exploration behaviors. The underlying objective takes a simple form, corresponding to a tracking problem in the spectral domain (tracking of frequency components). The advantage of such a control formulation is that it can be easily combined with other control objectives and constraints.

For industrial robot applications, the approach can be exploited in a wide range of problems and results in the automatic exploration of regions of interest. This is particularly helpful when the available sensing information is not accurate enough to fulfill the task with a standard controller, but where this information can still guide the robot towards promising areas. In a collaborative task, it can also be used when the operator's input is not accurate enough to fully reproduce the task, which then requires the robot to explore around the requested input (e.g., a point of interest selected by the operator). For picking and insertion problems, ergodic control can be applied to move around the picking/insertion point, thereby facilitating the prehension/entry. It can also be employed for active sensing and localization (either detected autonomously, or with help by the operator). Here, the robot can plan movements based on the current information density. It can then recompute the commands when new measurements are available (i.e., updating the spatial distribution used as target).

In ergodic control, the problem is originally formulated as a *spectral multiscale coverage* (SMC) objective [57]. It requires the spatial distribution to be decomposed as Fourier series, with a cost function comparing the spectral decomposition of the robot path with the spectral decomposition of the distribution. The resulting controller allows the robot to explore the given spatial distribution in a natural manner, by starting from a crude exploration and by refining the search progressively (i.e., matching the Fourier coefficients with an increasing importance from low to high frequency components).

Figure 3.12 presents a 2D example. *(a)* shows the spatial distribution $\hat{g}(\boldsymbol{x})$ that the agent has to explore, encoded here as a mixture of two Gaussians (gray colormap in left graph). The right graphs show the corresponding Fourier series coefficients $\hat{w}_{\boldsymbol{k}}$ in the frequency domain ($K = 9$ coefficients per dimension). *(b)* shows the evolution of the reconstructed spatial distribution $g(\boldsymbol{x})$ (left graph) and the computation of the next control command $\boldsymbol{u}$ (red arrow) after $T/10$ iterations. The corresponding Fourier series coefficients $w_{\boldsymbol{k}}$ are shown in the right graph. *(c)* shows that after T iterations, the agent covers the space in proportion to the desired spatial distribution, with a good match of coefficients in the frequency domain (we can see that $\hat{w}_{\boldsymbol{k}}$ and $w_{\boldsymbol{k}}$ are nearly the same). *(d)* shows how a periodic signal $\hat{g}(\boldsymbol{x})$ (with range $[-L/2, L/2]$ for each dimension) can be constructed from an initial mixture of two Gaussians $\hat{g}_0(\boldsymbol{x})$ (red area). The constructed signal $\hat{g}(\boldsymbol{x})$ is composed of eight Gaussians in this 2D example, by mirroring the Gaussians along horizontal and vertical axes to construct an even signal of period L. *(e)* depicts the first few basis functions of the Fourier series for the first four coefficients in each dimension, represented as a 2D colormap corresponding to periodic signals of different frequencies along two axes.

The downside of the original ergodic control approach is that it does not scale well to problems requiring exploration in search space of more than two dimensions.

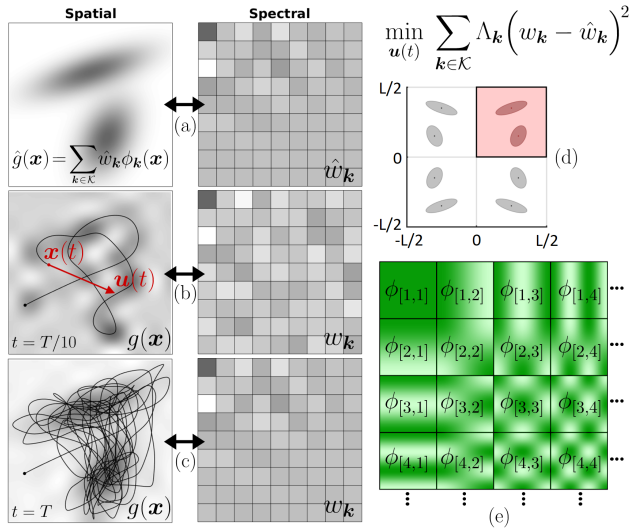


Figure 3.12: 2D ergodic control used to generate search behaviors (see main test for details). Figure adapted from [35].

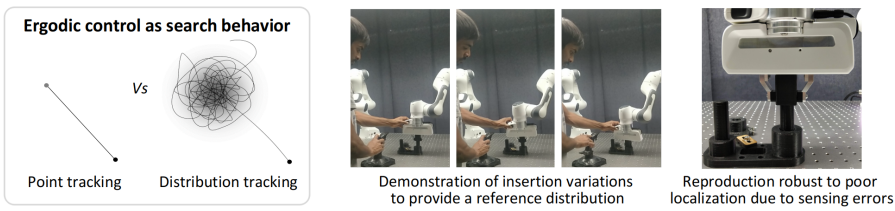


Figure 3.13: Insertion tasks with ergodic control by using the Siemens gears benchmark. Figure adapted from [60].

In particular, the original approach would be too slow to consider full distributions in 6D spaces, which would ideally be required. Indeed, both position and orientation of endeffector(s) matter in most industrial robot problems, including manipulation, insertion, welding, or the use of tools at the robot endeffectors. For this reason, alternative approaches mimicking similar behaviors, but departing from the original problem formulation, have been proposed [58, 59].

In [60], we demonstrated that the original problem formulation of [57] can be conserved by compressing the Fourier series decomposition with *tensor train* (TT) factorization. The proposed solution is efficient both computationally and storage-wise, hence making it suitable for online implementations, as well as to tackle robot learning problems with a low quantity of training data. The proposed approach using tensor trains and Fourier series to compress the spatial distributions has been tested in various experiments.

Figure 3.13 shows an overview of the insertion experiment with the Siemens gears benchmark, requiring full 6D endeffector poses, see [60] for details of the method and results. The experiment showed that the proposed approach can easily handle 6D spatial coverage problems (incl. position and orientation), with fast computation and small memory requirements.

3.7 Conclusion

This chapter presented an overview of the research directions that are currently investigated in the Robot Learning & Interaction group at the Idiap Research Institute to facilitate robot learning from demonstration. In contrast to the majority of ongoing work in machine learning, our research targets small data learning problems, with the goal of reducing the number of demonstrations that an operator has to provide to a robot, in order to let it acquire manipulation skills that can generalize to new situations. This objective led to the consideration of a diverse set of structures, including learning strategy structures, combination structures, geometric structures, and tensor data structures. This chapter showed how these structures can be beneficial to robot skill acquisition problems, with the goal of reducing the amount of data required to learn manipulations tasks, and of generalizing the learned tasks to new situations, to new robots and to new users. An overview of each approach was presented in the context of industrial tasks, by discussing research directions that still need to be investigated to enable further technology transfer to industrial robots.

References

[1] Billard AG, Calinon S, Dillmann R. Learning From Humans. In: Siciliano B, Khatib O, editors. Handbook of Robotics. Secaucus, NJ, USA: Springer; 2016. p. 1995–2014. 2nd Edition.

[2] Calinon S. Learning from Demonstration (Programming by Demonstration). In: Ang MH, Khatib O, Siciliano B, editors. Encyclopedia of Robotics. Springer; 2019. .

[3] Chatzilygeroudis K, Vassiliades V, Stulp F, et al. A Survey on Policy Search Algorithms for Learning Robot Controllers in a Handful of Trials. IEEE Trans on Robotics. 2020 Apr;32(2):328–347.

[4] Thomaz AL, Breazeal C. Teachable robots: Understanding human teaching behavior to build more effective robot learners. Artificial Intelligence. 2008 April;172(6-7):716–737.

[5] Cakmak M, DePalma N, Arriaga RI, et al. Exploiting social partners in robot learning. Autonomous Robots. 2010;29(3-4):309–329.

[6] Kulak T, Calinon S. Combining Social and Intrinsically-Motivated Learning for Multi-Task Robot Skill Acquisition. IEEE Trans on Cognitive and Developmental Systems. 2021;.

[7] Papageorgiou D, Dimeas F, Kastritsi T, et al. Kinesthetic Guidance Utilizing DMP Synchronization and Assistive Virtual Fixtures for Progressive Automation. Robotica. 2020;38(10):1824–1841.
[8] Makhataeva Z, Varol HA. Augmented Reality for Robotics: A Review. Robotics. 2020;9(2).
[9] Thoo YJ, Maceiras J, Abbet P, et al. Online and Offline Robot Programming via Augmented Reality Workspaces. arXiv:210701884. 2021;p. 1–8.
[10] Kimble K, Falco J, Messina E, et al. Benchmarking Protocols for Evaluating Small Parts Robotic Assembly Systems. IEEE Robotics and Automation Letters. 2020;5(2):883–889.
[11] Mussa-Ivaldi FA, Giszter SF, Bizzi E. Linear combinations of primitives in vertebrate motor control. Proc National Academy of Sciences. 1994 August;91:7534–7538.
[12] Pignat E, Silvério J, Calinon S. Learning from Demonstration using Products of Experts: Applications to Manipulation and Task Prioritization. International Journal of Robotics Research (IJRR). 2021;.
[13] Hinton GE. Products of Experts. In: Proc. Intl Conf. on Artificial Neural Networks (ICANN). vol. 1; 1999. p. 1–6.
[14] Calinon S. Stochastic learning and control in multiple coordinate systems. In: Intl Workshop on Human-Friendly Robotics. Genova, Italy; 2016. p. 1–5.
[15] Kelso JAS. Synergies: Atoms of Brain and Behavior. In: Sternad D, editor. Progress in Motor Control. vol. 629 of Advances in Experimental Medicine and Biology. Springer US; 2009. p. 83–91.
[16] Todorov E, Jordan MI. Optimal feedback control as a theory of motor coordination. Nature Neuroscience. 2002;5:1226–1235.
[17] Scholz JP, Schoener G. The uncontrolled manifold concept: identifying control variables for a functional task. Experimental Brain Research. 1999;126(3):289–306.
[18] Wolpert DM, Diedrichsen J, Flanagan JR. Principles of sensorimotor learning. Nature Reviews. 2011;12:739–751.
[19] Calinon S. A Tutorial on Task-Parameterized Movement Learning and Retrieval. Intelligent Service Robotics. 2016 January;9(1):1–29.
[20] Giusti A, Zeestraten MJA, Icer E, et al. Flexible Automation Driven by Demonstration: Leveraging Strategies that Simplify Robotics. IEEE Robotics and Automation Magazine (RAM). 2018 June;25(2):18–27.
[21] Calinon S. Gaussians on Riemannian Manifolds: Applications for Robot Learning and Adaptive Control. IEEE Robotics and Automation Magazine (RAM). 2020 June;27(2):33–45.
[22] Birk A, Doernbach T, Mueller CA, et al. Dexterous Underwater Manipulation from Onshore Locations: Streamlining Efficiencies for Remotely Operated Underwater Vehicles. IEEE Robotics and Automation Magazine (RAM). 2018;25(4):24–33.
[23] Ti B, Gao Y, Zhao J, et al. Imitation of Manipulation Skills Using Multiple Geometries. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS); 2022. .

[24] Silvério J, Calinon S, Rozo L, et al. Learning Task Priorities from Demonstrations. IEEE Trans on Robotics. 2019;35(1):78–94.

[25] Whiten A, McGuigan N, Marshall-Pescini S, et al. Emulation, imitation, over-imitation and the scope of culture for child and chimpanzee. Phil Trans R Soc B. 2009;364(1528):2417–2428.

[26] Akgun B, Thomaz A. Simultaneously learning actions and goals from demonstration. Autonomous Robots. 2016 Feb;40(2):211–227.

[27] Razmjoo A, Lembono TS, Calinon S. Optimal Control combining Emulation and Imitation to Acquire Physical Assistance Skills. In: Proc. IEEE Intl Conf. on Advanced Robotics (ICAR); 2021. p. 338–343.

[28] Calinon S, Lee D. Learning Control. In: Vadakkepat P, Goswami A, editors. Humanoid Robotics: a Reference. Springer; 2019. p. 1261–1312.

[29] Robotics codes from scratch (RCFS); 2022. https://robotics-codes-from-scratch.github.io/.

[30] Tanwani AK, Yan A, Lee J, et al. Sequential Robot Imitation Learning from Observations. International Journal of Robotics Research (IJRR). 2021;40(10–11):1306–1325.

[31] Lembono TS, Calinon S. Probabilistic Iterative LQR for Short Time Horizon MPC. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS); 2021. p. 556–562.

[32] Girgin H, Jankowski J, Calinon S. Reactive Anticipatory Robot Skills with Memory. In: Proc. Intl Symp. on Robotic Research (ISRR); 2022. .

[33] Paraschos A, Daniel C, Peters J, et al. Probabilistic Movement Primitives. In: Burges CJC, Bottou L, Welling M, et al., editors. Advances in Neural Information Processing Systems (NeurIPS). USA: Curran Associates, Inc.; 2013. p. 2616–2624.

[34] Jankowski J, Racca M, Calinon S. From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control. IEEE Robotics and Automation Letters (RA-L). 2022;.

[35] Calinon S. Mixture Models for the Analysis, Edition, and Synthesis of Continuous Time Series. In: Bouguila N, Fan W, editors. Mixture Models and Applications. Springer; 2019. p. 39–57.

[36] Ijspeert A, Nakanishi J, Pastor P, et al. Dynamical movement primitives: Learning attractor models for motor behaviors. Neural Computation. 2013;25(2):328–373.

[37] Broomhead DS, Lowe D. Multivariable Functional Interpolation and Adaptive Networks. Complex Systems. 1988;2(3):321–355.

[38] Pennec X. Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. Journal of Mathematical Imaging and Vision. 2006;25(1):127–154.

[39] Absil PA, Mahony R, Sepulchre R. Optimization Algorithms on Matrix Manifolds. Princeton University Press; 2007.

[40] Park FC, Bobrow JE, Ploen SR. A Lie Group Formulation of Robot Dynamics. The International Journal of Robotics Research. 1995;14(6):609–618.

[41] Selig JM. Geometric fundamentals of robotics. Springer; 2005.

[42] Barfoot TD, Furgale PT. Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems. IEEE Trans on Robotics. 2014 June;30(3):679–693.

[43] Zeestraten MJA, Havoutis I, Silvério J, et al. An Approach for Imitation Learning on Riemannian Manifolds. IEEE Robotics and Automation Letters (RA-L). 2017 June;2(3):1240–1247.

[44] Jaquier N, Calinon S. Gaussian Mixture Regression on Symmetric Positive Definite Matrices Manifolds: Application to Wrist Motion Estimation with sEMG. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS). Vancouver, Canada; 2017. p. 59–64.

[45] Lee T, Park FC. A Geometric Algorithm for Robust Multibody Inertial Parameter Identification. IEEE Robotics and Automation Letters. 2018;3(3):2455–2462.

[46] Traversaro S, Brossette S, Escande A, et al. Identification of fully physical consistent inertial parameters using optimization on manifolds. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS); 2016. p. 5446–5451.

[47] Jaquier N, Rozo L, Caldwell DG, et al. Geometry-aware Manipulability Learning, Tracking and Transfer. International Journal of Robotics Research (IJRR). 2021;40(2–3):624–650.

[48] Park FC, Kim JW. Manipulability of Closed Kinematic Chains. Journal of Mechanical Design. 1998;120(4):542–548.

[49] Maurice P, Malaisé A, Amiot C, et al. Human movement and ergonomics: An industry-oriented dataset for collaborative robotics. The International Journal of Robotics Research. 2019;38(14):1529–1537.

[50] Jaquier N, Rozo L, Calinon S. Analysis and Transfer of Human Movement Manipulability in Industry-like Activities. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS); 2020. p. 11131–11138.

[51] Yoshikawa T. Dynamic manipulability of robot manipulators. Robotic Systems. 1985;2:113–124.

[52] Dimeas F. Singularity Avoidance in Human-Robot Collaboration with Performance Constraints. In: Proc. Intl Workshop on Human-Friendly Robotics (HFR); 2020. p. 93–98.

[53] Marić F, Petrović L, Guberina M, et al. A Riemannian metric for geometry-aware singularity avoidance by articulated robots. Robotics and Autonomous Systems. 2021;145:103865.

[54] Orin DE, Goswami A. Centroidal Momentum Matrix of a humanoid robot: Structure and properties. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS); 2008. p. 653–659.

[55] Jaquier N, Haschke R, Calinon S. Tensor-variate mixture of experts for proportional myographic control of a robotic hand. Robotics and Autonomous Systems. 2021;142:103812.

[56] Shetty S, Lembono T, Löw T, et al. Tensor Train for Global Optimization Problems in Robotics. arXiv:220605077. 2022;.

[57] Mathew G, Mezic I. Spectral Multiscale Coverage: A uniform coverage algorithm for mobile sensor networks. In: Proc. IEEE Conf. on Decision and Control; 2009. p. 7872–7877.

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

[59] Abraham I, Prabhakar A, Murphey TD. An Ergodic Measure for Active Learning From Equilibrium. IEEE Transactions on Automation Science and Engineering. 2021;18(3):917–931.

[60] Shetty S, Silvério J, Calinon S. Ergodic Exploration using Tensor Train: Applications in Insertion Tasks. IEEE Trans on Robotics. 2021;.