UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Motor Primitives for Interaction with Robots in Complex Manipulation Tasks

Ali Kordia

Supervisor: Doctor Francisco António Chaves Saraiva de Melo

Thesis approved in public session to obtain the PhD Degree in

Computer Science and Engineering

Jury final classification: Pass with Distinction

2023

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Motor Primitives for Interaction with Robots in Complex Manipulation Tasks

Ali Kordia

Supervisor: Doutor Francisco António Chaves Saraiva de Melo

Thesis approved in public session to obtain the PhD Degree in

Computer Science and Engineering

Jury final classification: Pass with Distinction

Jury

Chairperson: Doutor José Carlos Alves Pereira Monteiro, Instituto Superior Técnico, Universidade de Lisboa.

Members of the Committee:

Doutor Manuel Fernando Cabido Peres Lopes, Instituto Superior Técnico, Universidade de Lisboa.

Doutor Luís Paulo Gonçalves dos Reis, Faculdade de Engenharia, Universidade do Porto.

Doutor Abbas Abdolmaleki, DeepMind, Google UK Ltd, UK.

Doutor Plinio Moreno López, Instituto Superior Técnico, Universidade de Lisboa.

Funding Institution

Global Platform for Syrian Students

2023

Abstract

As robots become a more common reality, they should be prepared to interact and collaborate with humans in their daily tasks. It becomes increasingly more challenging to pre-program a robot to deal with the multitude of contexts and tasks it faces when deployed. The ability of a robot to learn new tasks from demonstration is, therefore, a fundamental skill that robots should necessarily possess. These two competencies (learning from demonstration and collaborating with human users) are not disjoint and are anchored on the interaction between the robot and the human user.

The research in this thesis addresses several fundamental skills required of a robot in order to perform complex manipulation tasks in collaboration with a human user. Throughout the thesis, we represent robot motions as dynamic movement primitives (DMPs) or combinations thereof. Basic movements are learned from demonstrations provided by human users and stored in a library. The thesis discusses how to build new basic movements given the demonstrations from a user, how to combine them into novel, complex movements, and how to use the library of known movements to recognize observed movements. The contributions of the thesis are threefold:

- An end-to-end framework that enables a robot to observe movements demonstrated by a human user; break these down into "atomic movements" that are then stored in a library; compose these atomic movements into longer, more complex movements that are then optimized to achieve a given goal; execute the resulting movement while ensuring obstacle avoidance in real-time.

- An approach for recognition and prediction of observed movements. Given a library of previously learned movements, the proposed approach observes a partial movement of a human user. It matches it to the movements previously learned, with no need for time alignment. It can then predict the remainder of the user's movement. We illustrate the application of this approach in scenarios where the

robot uses the predicted user's movement to move and assist the human user proactively. We also propose an extension of the basic approach described above to compound movements corresponding to compositions of the movements in the robot's library.

- An approach that enables the consideration of multiple independent manipulators, extending our end-to-end framework to consider, for example, robots with multiple arms. Our framework enables the joint optimization of the motion of the different manipulators towards more complex motions.

We validate the proposed frameworks and methods both in simulation scenarios—designed to highlight the different capabilities of our methods—as well as in the real-world scenarios involving the Baxter robot.

Keywords: Robot Learning ⋄ Movements Manipulation ⋄ Robot Interaction ⋄ Dynamic Movement Primitives ⋄ Recognition and Prediction.

Resumo

À medida que os robôs se tornam uma realidade cada vez mais comum, devem estar preparados para interagir e colaborar com humanos nas diversas tarefas que estes realizam no seu dia-a-dia. Assim, torna-se cada vez mais desafiante pré-programar um robô para que este lide com a variedade de contextos e de tarefas com que eventualmente se irá deparar. A capacidade de um robô aprender novas tarefas a partir de demonstrações é, por isso, uma habilidade fundamental que os robôs devem necessariamente ter. Estas duas competências (aprendizagem por demonstração e colaboração com utilizadores humanos) não são disjuntas mas assentam na interação entre o robô e o utilizador.

Esta tese aborda várias das competências fundamentais que permitem a um robô executar tarefas de manipulação complexas em colaboração com um utilizador humano. Ao longo desta tese, o movimento do robô é representado como primitivas de movimento dinâmicas (*dynamic movement primitives*, ou DMPs) ou combinações destas. Movimentos básicos são aprendidos a partir de demonstrações fornecidas por utilizadores humanos e armazenados numa biblioteca de movimentos. Assim, a tese aborda como construir novos movimentos básicos a dadas demonstrações de um utilizador; como combinar diversos movimentos básicos em novos movimentos complexos, e como utilizar os movimentos na biblioteca para reconhecer movimentos observados. Esta tese tem, assim, três contribuições principais:

- Uma abordagem *end-to-end* que permite a um robô observar movimentos demonstrados por um utilizador; decompôr estes movimentos em "movimentos atómicos" que são guardados numa biblioteca; compor movimentos mais longos e complexos a partir destes movimentos atómicos, otimizando-os para um dado objectivo; executar o movimento resultante, evitando obstáculos que surjam em tempo de execução.

- Uma abordagem para o reconhecimento e previsão de movimentos observados. Dada uma biblioteca de movimentos previamente aprendidos, a abordagem proposta observa um movimento parcial executado por um utilizador. É feita a correspondência deste movimento parcial aos movimentos previamente aprendidos, sem necessitar de qualquer alinhamento temporal. Além disso, a abordagem proposta permite a previsão do restante movimento do utilizador. Ilustra-se a aplicação da abordagem em cenários nos quais o robô usa a previsão do movimento do utilizador para proativamente se mover e assistir o utilizador. É também proposta uma extensão da abordagem básica, descrita anteriormente, para movimentos compostos, correspondendo à composição de movimentos na biblioteca do utilizador.

- Por fim, é proposta uma abordagem que permite considerar múltiplos manipuladores independentes, estendendo a abordagem *end-to-end* acima descrita por forma a considerar, por exemplo, robôs com múltiplos braços. Esta abordagem permite a otimização conjunta do movimento de diferentes manipuladores, visando movimentos mais complexos.

As abordagens propostas e os métodos considerados são avaliados tanto em cenários de simulação—desenhados de forma a realçar as diferentes capacidades dos métodos—bem como em cenários reais com um robô Baxter.

Palavras-chave: Aprendizagem para Robôs ⋄ Manipulação de Movimentos ⋄ Interação Robô ⋄ Reconhecimento e Previsão ⋄ Primitivas Motoras Dinâmicas.

Acknowledgements

I am writing these acknowledgments as I am about to end this stage of my life, and I hope there are still people with us to read my words and my profound gratitude to them.

My first and immense appreciation goes to my supervisor, Prof. Francisco S. Melo. His patience, enormous assistance, enthusiasm, cooperation, and suggestions were fundamental in the creation of my research work. His brilliant, skillful supervision enriched my research well beyond my expectation. Prof. Francisco has been an inspiration as I hurdled through the path of this Ph.D. degree path. He is the true definition of a leader and the ultimate role model. This thesis would not have been possible without Prof. Francisco's constant support, guidance, and assistance. His level of patience, knowledge, and ingenuity is something I will always keep aspiring to. He is someone who always—remarkably, impossibly—made me feel excited about my work. I do not know how you did it, but you made my Ph.D. seem joyful despite all the hardships.

I thank all my thesis proposal committee members, Prof. Luís Paulo Reis, Prof. Plinio López, and Prof. Manuel Lopes. Your encouraging words and thoughtful, detailed feedback have been significant to me and helped shape this work.

I am also indebted to my parents, who are my supporters and friends. Baba, Hani Kordia, present yet absent. Thank you for all the strength you have given me with your love. Mama, Mariam Razouk, you are my inspiration and role model in life, a superb mathematician, and my first teacher.

I am fortunate to have been a part of the GAIPS group. Thank you all for the fun activities and fantastic company. I want to express my sincere gratitude to Prof. Ana Paiva. It is my honor to be a member of your group. I am also grateful to Prof. Rui Prada for your outstanding support and help. I express my deepest gratitude to all Professors at GAIPS: Francisco Santos, Alberto Sardinha, Andreas Wichert, Carlos

Martinho, and Joana Campos. Thanks to Sandra Sá, who helped with everything in the lab. Thanks to Prof. Agostinho Rosa for his support when I was at ISR.

Words cannot express my gratitude to my precious friend Miguel Vasco, and I want to tell you that our friendship makes me forget my feeling of estrangement. I would also like to express my deepest gratitude to Filipa Correia, my sweet friend, whose kindness and help were a constant presence in our lab. I am deeply indebted to Diogo Carvalho, my adventurer friend; we have shared exciting and funny stories. Thank you for being so supportive. I am highly grateful to João Ribeiro, a kind friend, and collaborative colleague as a lab manager. Thank you for the Guinness beer in Cascais.

Also, I greatly appreciate my friends and colleagues at GAIPS for their great feedback, excellent encouragement, and guidance. Thank you a lot, this endeavor would not have been possible without you. Thank you, Fábio Vital, for the friendly company in our office and the chess games. Thank you, Captain Raul Benites Pardadeda, for your humility and acceptance of playing football with me. Thanks for the fantastic maté cup and lollipop. Thank you, Raquel Oliveira, for the exciting conversations, your great support, and the fantastic companionship. Thank you, Silvia Tulli, for your fun and energetic personality that motivated me to return to activities I used to do; you are a superb person. Thank you, Alexis Jacq, for the fantastic company at Alfama with Pastis. Thank you, Pedro Pinto Santos, for our pleasant conversations and valuable sessions on RL. Thank you, Diogo Rato, for our lovely conversations, kindness, support, and humor. Thank you, Hang Yin, for your advice and kindness. It was super helpful to have you share with me your experience with Baxter. Thank you, Kim Baraka, for reminding me of everything beautiful in our countries. Thank you, Ricardo Rodrigues, for the delicious coffee and the deep and entertaining discussions. Thank you, Manuel Guimarães, for your support, fun, and beautiful poster that I will always keep close to my heart. Thank you, Luís Costa, for the strength and optimism and for supporting me. Thank you, Miguel Faria, for your kind company and your support. Thank you, Ramona Merhej, for sharing many funny stories and pleasant conversations. Thank you, Elmira Yadollahi, for your lovely companionship and support. Thank you, Maria José, for your advice, support, and kindness. Thank you, Daniel Tozadore, for the fun time we spent together. Thank you, Mojgan Hashemian, for your kindness, support, and Iranian desserts. Thank you, José Rocha, for the delicious BBQ and time at your

house. Thank you, Shruti Chandra, for your generous kindness.

A big thank you to all other GAIPS members: Samuel Gomes, Guilherme Varela, Jacopo Silvestrin, Samuel Mascarenhas, Tiago Ribeiro, Marta Couto, Fernando Santos, Catarina Moreira, Rui Silva, and Carla Guerra, for their constant support and help, whose sincerity and encouragement I will never forget. Thanks and good luck to the new generation of GAIPS: Ana Carrasco, Henrique Fonseca, Inês Lobo, Ana Antunes, Pedro Fernandes, and Inês Batina. I wish you all the best in your Ph.D. journey.

I would also like to express my gratitude to former Portuguese President Dr. Jorge Sampaio for his support and high nobility.

My warm and heartfelt thanks to my lovely family in Syria for the tremendous support and hope they have given to me. Without that hope, this thesis would not have been possible. Thank you, Amjad, Majd, and Hala Kordia, for all the strength you gave me. I love you all, and I am hoping to see you soon.

I would also like to thank my Syrian friends for their camaraderie and support. My precious cousin and childhood friend, Alaa Kordia, for your love and concern for me. Thank you to Ibrahim Wardeh and Ahmed Alhisri for the beautiful music and the excellent support. Thank you, Batraa Abo Aljadayel, Mai Wardeh, Salem Nizami, Hussein Faruque, Yara Alhisri, Hana Darouich, Tiago Ramos, and all the other friends. Thank you for your tremendous support. Our friendships made writing less of an isolated slog through hell.

Syria, my wounded country, was torn by war and became a place full of destruction and tragedy for over a decade. I wish my homeland all the happiness and peace. I hope the surviving Syrians will sleep reassured and comfortable after all this fatigue and hardships.

Portugal, where I found peace, safety, and good people. I wish you all prosperity, wealth, and headway.

Finally, I acknowledge the Ph.D. grant from the Global Platform for Syrian Students, IST, and INESC-ID.

I am so proud of each one of you, and I love you all! Thank you for providing me with enough love to do the impossible.

Lisbon, July 2022

I am updating this final version of my dissertation. It aligns with a tremendous and

terrifying earthquake that happened on the 6th of February, 2023, which destroyed vast areas of my beloved country Syria and of Turkey.

I send my most profound compassion and deepest sympathy to the families of the victims and the families of the afflicted; my sincere condolences to you! I want to tell them we are with you; you are not alone! I am melancholy about what happened to you, and I hope to be able to help more in the future.

I want to tell those still under the rubble and those who died that we are sorry for our inability to help you; your memories and names will remain in our minds.

Lisbon, February 2023

Contents

List of Figures

Chapter 1

Introduction

As robots become a more common reality, their applicability broadens. Consequently, it becomes increasingly more challenging to pre-program a robot to deal with the multitude of contexts and tasks it faces when deployed. The ability of a robot to learn new tasks from demonstration is, therefore, a fundamental skill that robots should necessarily possess [115].

There is a vast literature on learning from demonstration [8, 66]. It is possible, for example, to consider learning from a demonstration at the *task level*, where the learner seeks to infer the goal of the demonstration and adopt it as its own [16, 109]. Conversely, it is possible to address learning from demonstration at the *motor resonance level* [10, 33], where the learner simply seeks to replicate the motor behavior observed from the demonstration. Our work lies somewhere in the middle and considers learning from the demonstration as learning how to perform and combine "atomic actions", referred to as *motion primitives*—or, when disregarding perception issues, simply as *motor primitives* [76, 81].

In this work, we use demonstrations from the user to build a library of abstract atomic actions that are then combined to perform complex motions. Throughout the thesis we adopt the formalism of dynamic motor primitives [39] to represent such abstract atomic actions. DMPs are robust to perturbations and can be easily learned from demonstrations. Additionally, modulation (i.e., performing the same motion to different end-points) can easily be achieved by changing some of the driving terms of the underlying dynamical system.

Besides learning new movements, one of the essential aspects of human-robot collaboration is the ability of the robot to recognize what the human partner is doing and respond accordingly. In human-human collaboration, each person observes and "intuitively" recognizes/predicts the partner's motion, allowing the person to act in the best possible way. Humans are naturally flexible concerning a partner's actions and adapt, fostering a high-level collaboration and inter-partner trust. The key for collaboration is to understand the task and the intentions of the partner, which often consists of *recognizing/predicting* the goal of the movement. However, recognizing the goal of an observed movement early during its execution is a difficult problem in robotics; our work also addresses such problem, i.e., movement recognition and prediction.

Given the aforementioned library of previously observed movements, we recognize a

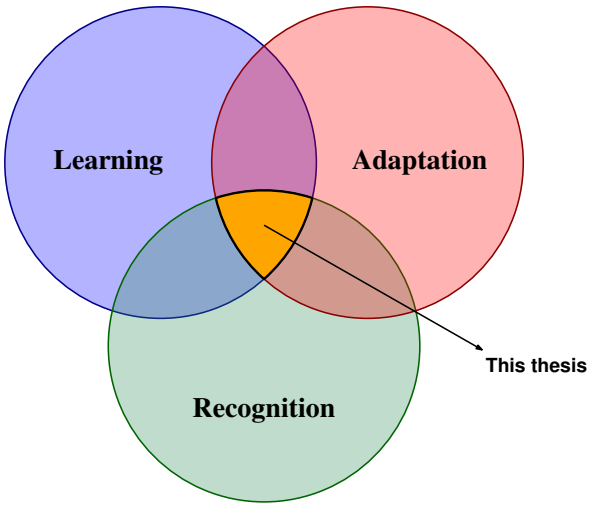


Figure 1.1: Positioning of the thesis contributions.

partially observed trajectory among those in the library, predicting its (unobserved) target point. The fact that our approach does not require the full trajectory to be observed is fundamental, as it can be used in run-time for actual human-robot interaction scenarios.

Finally, the robot needs to adapt and adjust its motion and behavior to new surrounding circumstances, not losing track of the primary task that it must perform. This work also considers the challenge of adaptation during task execution, in case the robot encounters changes to the conditions used during learning [120]. The robot also needs the ability to accommodate more complex tasks than those it is taught to do [113]. Specifically, the robot should be able to reuse its knowledge from one task to another, or from one environment to another.

The contributions of our thesis thus lie at the intersection of *motion learning*, *motion recognition* and *motion adaptation*, as highlighted in Fig. 1.1. We contribute novel models and methods to endow a robot with fundamental skills to successfully engage in collaborative tasks with human users. Human-robot collaboration requires the robot to *learn* new complex motions and *adapt* its knowledge to new conditions to succeed. Besides that, it is crucial that the robot is able to *recognize* the observed movements of a human user to respond correctly.

1.1 Problem Statement

From all the above, the central research question addressed in this thesis is the following:

How can a robot possessing a previously learned repertoire of primitive movements execute complex manipulation tasks, possibly in collaboration with a user?

We address the above research question by considering the following research challenges:

- How can a robot, endowed with a library of pre-learned primitive movements, address complex tasks that require the *composition*, *adaptation* and *optimization* of such primitive movements to address changes to the environment, while accommodating for the possibility of obstacle avoidance at execution time?
- How can a robot, endowed with a library of pre-learned primitive movements, recognize a movement by observing only a fraction thereof, predicting the endpoint of such movement?
- How can a robot, endowed with a library of pre-learned primitive movements, recognize a complex movement consisting of a *sequence* of primitive movements?
- Finally, how can a robot, endowed with a library of pre-learned primitive movements, generate multiple simultaneous *coordinated* robotic movements, adapting and optimizing those in the library, to complete one *collaborative task*?

The contributions of the thesis arise from addressing each of the above challenges individually.

1.2 Thesis Contributions

Figure 1.2 summarizes the thesis main contributions, showcasing their relation. At the core of our contributions is a library of primitive movements (shown in the figure as "DMP library"). This library is enriched with new movements learned from demonstration, and plays a central role in movement recognition and generation.

Summarizing, the main contributions of this thesis are:

- An end-to-end approach to learning and decomposing user demonstrations into DMPs, and then optimizing and combining the resulting DMPs to perform complex motions. Our framework comprises a library of DMPs and a set of different modules for (i) segmenting a demonstration provided by a human user, abstracting the computed sub-trajectories into new DMPs and adding them to the

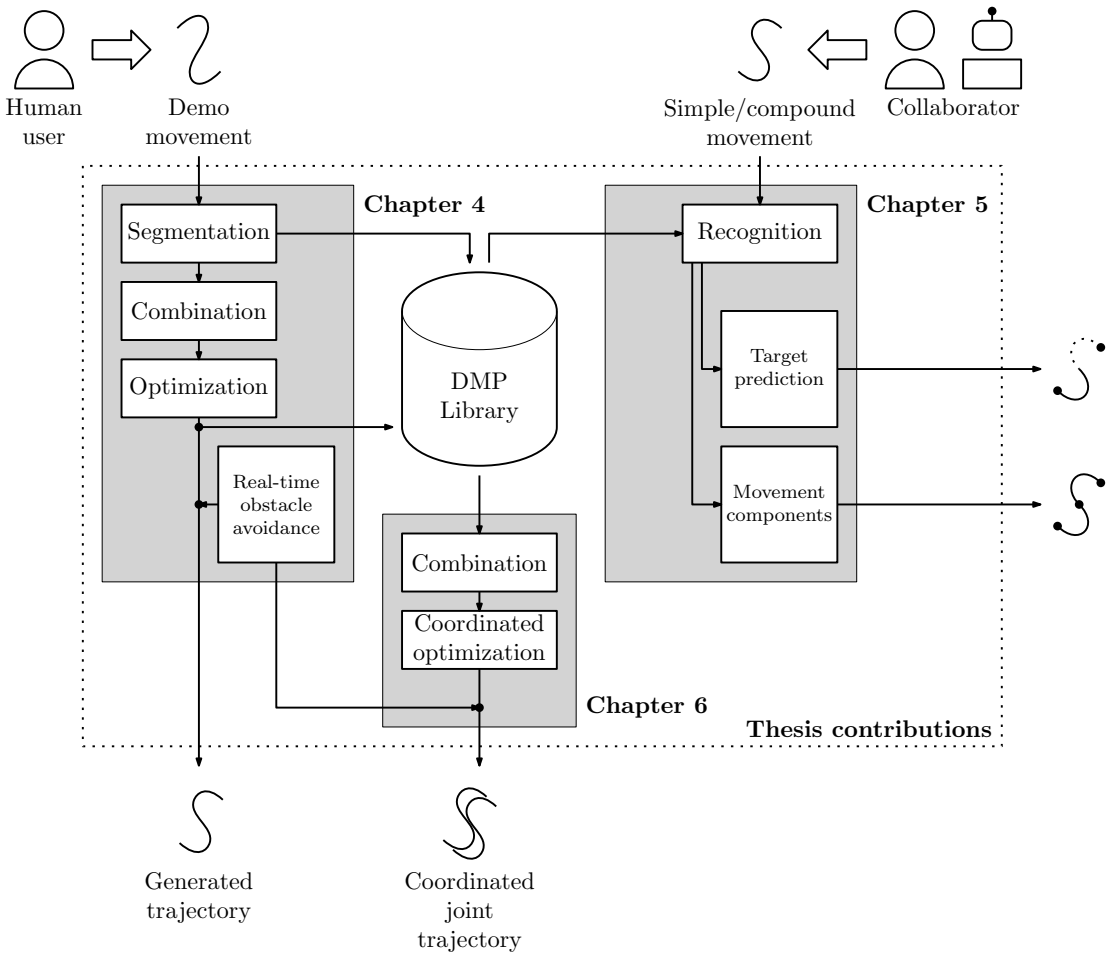


Figure 1.2: Overview of the thesis contributions.

library; (ii) a greedy algorithm that, given a goal point, combines the DMPs in the library into a single complex motion that is then optimized according to provided criteria (length of trajectory, obstacle avoidance, etc.); (iii) a module that, during execution, monitors the existence of obstacles and reshapes the trajectory to avoid them in run-time. This contribution is described in Chapter 4 and can be found in Kordia and Melo [52].

- A novel movement recognition method that, given a library of previously observed movements, identifies a partially observed trajectory from among those in the library, predicting its (unobserved) target point. The fact that our approach does not require the full trajectory to be observed is fundamental, as it can be used in run-time for actual interaction scenarios. We also contribute an extension of our recognition method to compound movements. These contributions are described in Chapter 5 and have previously been published in Kordia and Melo [54] and

Kordia and Melo [53].

- An extension of our end-to-end framework, consisting of an approach to optimizing and coordinating multiple movements, to be executed *simultaneously* by multiple manipulators in a coordinated fashion. From a library of single trajectories, a coordinated trajectory is built and optimized to perform a coordinated task. This contribution is described in Chapter 6.

1.3 Document Outline

The remainder of the document is structured as follows. Chapter 2 provides an overview of fundamental concepts for our work—such as dynamic movement primitives and the covariance matrix adaptation-evolution strategy (CMA-ES) algorithm. Chapter 3 discusses related work. Chapter 4 presents our first contribution: an end-to-end approach for building complex movements from a library of learned primitive movements. Chapter 5 addresses the problem of movement recognition, introducing our second contribution. Chapter 6 addresses the problem of optimizing and coordinating multiple DMPs under constraints for complex coordinated manipulation tasks. The document concludes in Chapter 7 with a discussion of the main conclusions and highlighting some directions for work beyond that in this thesis.

Chapter 2

Background

This chapter provides an overview of background concepts used throughout the thesis. Most notably, we provide an overview of dynamic movement primitives, discuss optimization using CMA-ES, and introduce the concept of critical points in a trajectory.

2.1 Dynamic Movement Primitives

Dynamic movement primitives (DMPs) are a popular representation for movements used in robot motion. The motivation for adopting DMPs as a movement representation is related to the numerous works relying on DMPs [114]. DMPs have a simple and rigorous mathematical formulation, ensure robustness to external perturbations and flexibility to adapt the motions to different tasks. A DMP consists of a stable nonlinear system with a force term that shapes the trajectory of the system and which can easily be used to adapt complex motions with the guarantee of reaching the desired target [114].

Formally, a *dynamic movement primitive* [37, 38] is a representation of a smooth trajectory using a stable dynamical system. A one-dimensional DMP corresponds to the nonlinear system of equations

$$\ddot{y}(t) = \alpha_y(\beta_y(y_{\text{ref}} - y(t)) - \dot{y}(t)) + f(x(t)), \tag{2.1a}$$

$$\dot{x}(t) = \alpha_x x(t), \tag{2.1b}$$

where $y(t)$ is the state of the DMP and $x(t)$ is the so-called *phase variable*. In the formulation above, (2.1a) is known as the *transformed system*, while (2.1b) is known as the *canonical system*. Under a suitable choice of α_y, β_y and α_x, $y(t) \to y_{\text{ref}}$ and $\dot{y}(t) \to 0$ as $x(t) \to 0$.

The DMP (2.1) behaves as a spring-damp system, as illustrated in Fig. 2.1. The particular shape of the trajectory by which $y(t) \to y_{\text{ref}}$ depends on the force function f, which is usually represented as the linear combination of a set of *basis functions*,

$$f_w(x) = \frac{\sum_{n=1}^{N} w_n \Psi_n(x)}{\sum_{n=1}^{N} \Psi_n(x)} x. \tag{2.2}$$

The weights $w_n, n = 1, \ldots, N$, can now be adjusted to yield a smooth trajectory between $y(0)$ and y_{ref}. For example, we can have *periodic DMPs* by selecting a the basis functions

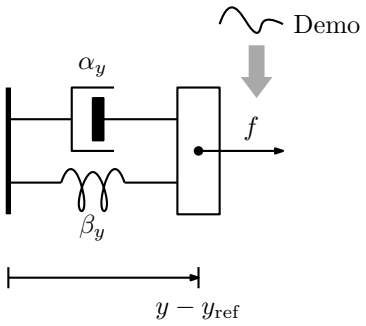


Figure 2.1: Depiction of the dynamical system used as the basic representation of a dynamic motion primitive. A DMP is a damped spring system, where the state of the system, y, represents the quantity of interest (robot position, joint position, etc.). The forcing function f is used to control the shape of the trajectory to match that provided by an expert demonstration.

Ψ_n to be periodic on x. Conversely, by selecting Ψ_n to be non-periodic, we obtain general non-periodic motions—in what we refer to as a *discrete DMP*.

The standard basis functions consist of Radial Basis Functions (RBFs) of the form

$$\Psi_n(x) = \exp\{-h_n(x - c_n)^2\},$$

where c_n is the *center* of the nth basis function, and h_i its width.

In the context of robot motion, each degree of freedom of the robot is controlled by an individual one-dimensional DMP, and synchronization is achieved through a shared canonical system. The use of a canonical system instead of a synchronized clock ensures that the resulting system is time-homogeneous, which in turn brings advantages in terms of control.

The canonical system provides the phase for the system, and enables the manipulation of time during the execution of the DMP [37]. The standard formulation is that in (2.1b), corresponding to an *exponentially decaying* phase variable. However, other alternatives have been proposed that seek to mitigate the fast drop in the phase variable observed in this formulation.

For example, Kulvicius et al. [58] proposed a *sigmoidal decaying* phase to combine DMPs using overlapping kernels. Using a sigmoid function instead of the exponential function prevents the velocity of each DMP from reaching zero prematurely, which is advantageous when combining multiple DMPs. In fact, with using sigmoidal functions, the average velocity of the system can be kept approximately constant during the

transition from one DMP to the next, allowing the following DMP to resume from that velocity. Using a sigmoidal decaying function, the canonical system takes the form

$$\dot{x}(t) = -\frac{\alpha_x \exp\{\alpha_x(T-t)/\Delta t\}}{1+\exp\{(\alpha_x(T-t)/\Delta t)^2\}}$$

where α_x controls the steepness of the sigmoidal function and Δt is the sampling rate.

Finally, the transformed system can be augmented with an additional component used to accommodate (smooth) shifts in the target/reference position. Such component can be described as a first-order dynamical system

$$\dot{y}_{\text{ref}}(t) = \begin{cases} \frac{\Delta t}{T}(y_0 - y^*) & \text{if } t \leq T \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

with y_0 and y^* representing the initial and final target positions.

2.1.1 Properties of DMPs

We adopt DMPs to represent robot motion for the many appealing properties that such representation entails.

- **Stability:** The DMP system (2.1) consists of a second-order differential equation driven by a force term $f(t)$ that controls the "shape" of the trajectory. The parameters of the second-order equation, α_y, β_y and α_x are selected or learned to guarantee that the dynamical system is critically damped and thus stable. A system that results from the incorporation of a group of subsystems (such as the robot in which each degree of freedom moves according to a DMP), sequentially or in parallel, is stable if these subsystems are in turn stable [65]. As a consequence, the DMP system is stable as a whole.

- **Invariance Properties:** The DMP is a general dynamical system; it can be used to recreate a learned motion but also to *modulate* the motion to a new goal point, which originally consisted of the last state of the reference demonstration of the learned motion, y_{ref}. The DMP representation enables such modulation while maintaining the overall shape of the trajectory [118].

- **Multiple Degrees of Freedom:** The tasks we consider in robotic applications involve multiple degrees of freedom (DoFs)—for example, corresponding to the different joints of the robot. As previously discussed, the use of DMPs with a shared canonical system and individual transformed systems for each DoF generates a "coordinated" motion that is stable and which relies on the canonical system as a "central clock" [117].

Besides the advantages listed above, DMPs also offer other appealing properties for the challenges addressed in this thesis. Unlike more complex models (e.g., neural networks), DMPs can be easily optimized by adjusting the weights of the basis functions defining the force function. In particular, it is possible to use optimization to adapt DMPs to new environments and even new tasks (as we do in this thesis). They can be learned from a single trajectory, meaning that they are extremely data efficient, unlike other more complex models such as neural networks.

However, DMPs also have some disadvantages. For example, DMPs are deterministic models that do not consider uncertainty, unlike other models such as probabilistic movement primitives [97]. Additionally, adaptation to difficult environments may significantly deteriorate the shape of the resulting movement, which will bare little resemblance with the original DMP movement. Our approach in Chapter 4 seeks to address this limitation, to some extent, by considering the composition of multiple DMPs.

2.1.2 Learning from Demonstration

Learning a DMP from demonstration consists of computing the weights $w_n, n = 1, \ldots, N$, for the forcing function f such that the DMP trajectory matches an observed movement (a demonstration) as closely as possible [39, 91].

Let $\xi = \{y_0, \ldots, y_M\}$ denote a target trajectory, observed in some finite set of points $\mathcal{T} = \{t_0, \ldots, t_M\}$. Let $\{\dot{y}_m, m = 1, \ldots, M\}$ and $\{\ddot{y}_m, m = 1, \ldots, M\}$ denote the corresponding first and second derivatives at those time points. We want to compute f such that the trajectories of the system (2.1) are such that $y(t_m) = y_m$, $\dot{y}(t_m) = \dot{y}_m$ and $\ddot{y}(t_m) = \ddot{y}_m$ as much as possible, for $m = 1, \ldots, M$. Since, from (2.1),

$$f(x(t)) = \ddot{y}(t) - \alpha_y(\beta_z(y_{\text{ref}} - y(t)) - \dot{y}(t)),$$

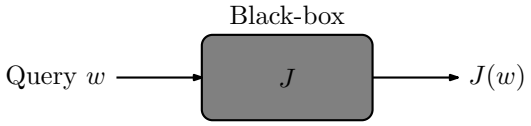


Figure 2.2: Basig interaction in black-box search.

we compute, for each $t_m \in \mathcal{T}$,

$$f_m = \ddot{y}_m - \alpha_y(\beta_z(y_{\text{ref}} - y_m) - \dot{y}_m),$$

with $y_{\text{ref}} = y_M$. Such computation yields a dataset $\{(t_m, f_m), m = 1, \ldots, M\}$ and we can now use standard supervised learning to compute the weights w of the forcing function. For example, using *locally weighted regression* [11, 116], we compute the weights $w_1, \ldots, w_N$ that minimize the loss

$$L(w_n) = \sum_{m=1}^{M} \sum_{n=1}^{N} \Psi_n(t_m)(f_m - w_n x)^2.$$

We compute the weights that minimize the locally weighted quadratic error criterion as it is a weighted linear regression problem:

$$w_n = \frac{\mathbf{x}^T \mathbf{\Psi}_n \mathbf{f}}{\mathbf{x}^T \mathbf{\Psi}_n \mathbf{x}}, \tag{2.4}$$

with

$$\mathbf{x} = \begin{bmatrix} x(t_1) \\ \vdots \\ x(t_M) \end{bmatrix}, \qquad \mathbf{\Psi}_n = \begin{bmatrix} \Psi_n(t_1) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \Psi_n(t_M) \end{bmatrix}, \qquad \mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_M \end{bmatrix}.$$

2.2 CMA-ES

In executing complex tasks, we may eventually want to *optimize* the movement of the robot to meet some optimality criterion—often imposed either by the environment or by the task itself.

Black-box optimization (BBO) algorithms work as depicted in Fig. 2.2: they query the value of the objective function, J, at different query points, w. The values $J(w)$ are then used to pick new query values and the process repeats. In our work, we use

BBO algorithms to optimize trajectories represented as DMPs, namely by adjusting the weights of the force function controlling the DMP.

Our choice of BBO is driven by the specific nature of the problems considered in this thesis—namely, the optimization of trajectories. Our search domain (the space of weights) is high-dimensional; the objective function is highly nonlinear and often even discontinuous (for example, when considering obstacles). In light of all these challenges, we adopt a specific BBO algorithm known as *covariance matrix adaptation evolutionary strategy* [31], or CMA-ES. CMA-ES can be used to optimize a function that is not known in advance, but which can be evaluated at any desired query points. CMA-ES belongs to a large family of evolutionary strategy algorithms that seek to optimize the objective function while minimizing the number of required function evaluations.

For the sake of concreteness, let us suppose that we are interested in maximizing a real-valued objective function J defined over $\mathbb{R}^N$. At each iteration i, CMA-ES proceeds by:

1. **Sampling:**
   - Sample K points $\{w_{i,k}, k = 1, \ldots, K\}$ from a multivariate Gaussian, $w_{i,k} \sim \text{Normal}(\mu_i, \sigma^2\Sigma_i)$, with $\mu_i \in \mathbb{R}^N$ and $\Sigma_i \in \mathbb{R}^{N\times N}$;
2. **Evaluating:**
   - Compute $J(w_{i,k})$ for $k = 1, \ldots, K$;
   - Select the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$;
3. **Adapting:**
   - Update the mean μ_i to the weighted average of the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$ selected in the previous step;
   - Update the covariance matrix Σ_i to increase the likelihood of previously successful search steps.

Figure 2.3 illustrates a possible evolution of CMA-ES. The figure shows how the samples tend to concentrate around the maximum of the target function (the circular concentric lines correspond to level-sets of the objective function J). Several recent works have further extended its applicability and properties [1, 32, 125].

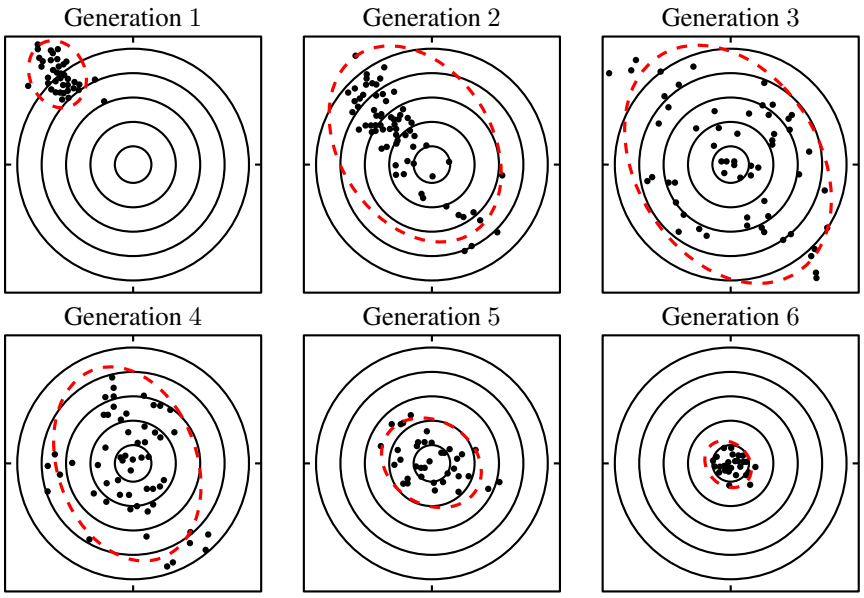


Figure 2.3: Evolution phases of of search process using Covariance Matrix Adaptation-Evolution Strategy, or CMA-ES (image adapted from [127]).

Algorithm 1 CMA-ES algorithm [12].

Require: $K, K_0, c_\sigma, c_c, c_{\text{cov}}, d_\sigma, \{\alpha_k, k = 1, \ldots, K_0\}$
Require: $\mu_0 \in \mathbb{R}^N, \sigma_0 \in \mathbb{R}_+$
Initialize $\Sigma_0 = I$, $p_c = p_\sigma = 0$, $i = 0$
while not terminate do
  for $k = 1, \ldots, K$ **do**
    Sample $w_k \sim \text{Normal}(\mu_i, \sigma_i^2 \Sigma_i)$
    Compute $J(w_k)$
  Sort $w_1, \ldots, w_K$ according to $J(w_k)$, with $w_{k:K}$ denoting the kth best sample out of K
  Set

$$\mu_{i+1} = \mu_i + \sum_{k=1}^{K_0} \alpha_k (w_{k:K} - \mu_i)$$

  Set $p_\sigma = (1 - c_\sigma) p_\sigma + \frac{\rho}{\sigma_i} \sqrt{1 - (1 - c_\sigma)^2} \Sigma_i^{-\frac{1}{2}} (\mu_{i+1} - \mu_i)$, where $1/\rho = \sqrt{\sum_{k=1}^{K_0} \alpha_k^2}$
  Set

$$\sigma_{i+1} = \sigma_i \exp\left\{ \frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma\|}{\mathbb{E}\|\text{Normal}(0, I)\|} - 1 \right) \right\}$$

  Set $p_c = (1 - c_c) p_c + \frac{\rho}{\sigma_i} \sqrt{1 - (1 - c_c)^2} (\mu_{i+1} - \mu_i)$
  Set

$$\Sigma_{i+1} = (1 - c_{\text{cov}}) \Sigma_i + \frac{c_{\text{cov}}}{\rho^2} p_c p_c^T + \frac{c_\mu}{\sigma_i^2} \sum_{k=1}^{K_0} \alpha_k (w_{k:K} - \mu_i)(w_{k:K} - \mu_i)^T \tag{2.5}$$

  Set $i = i + 1$
return μ_i

Pseudo-code for CMA-ES is provided in Algorithm 1. It receives, as parameters, the number of samples per generation, K, the number of samples used for updating the mean, K_0, with $K_0 < K$ (typically, $K_0 \leq K/2$), and a number of constants used in the covariance matrix update (c_σ, c_c, c_{cov} and d_σ). It also receives as parameters the weights $\alpha_1, \ldots, \alpha_{K_0}$ used in the update of the mean.

As outlined above, the mean is updated as a convex combination of the K_0 best samples (lines 6 and 7). The covariance matrix update (lines 10 and 11), on the other hand, comprises several main components:

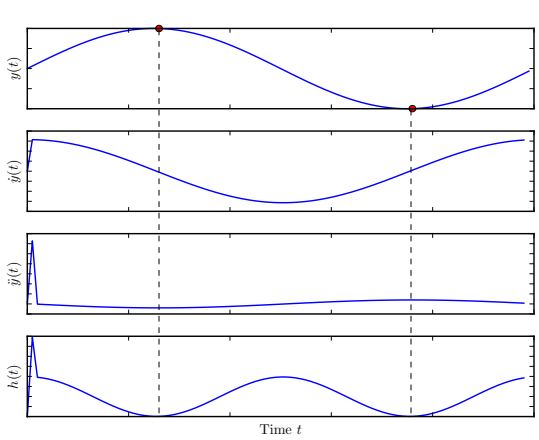
- The weighted covariance of the best samples—corresponding to the last term in (2.5); this term focuses the distribution towards more promising regions of the search space.

- A *bootstrapping term*—corresponding to the first term in (2.5); this term helps to ensure that Σ_{i+1} retains rank N, by using information from previous estimates, Σ_i.

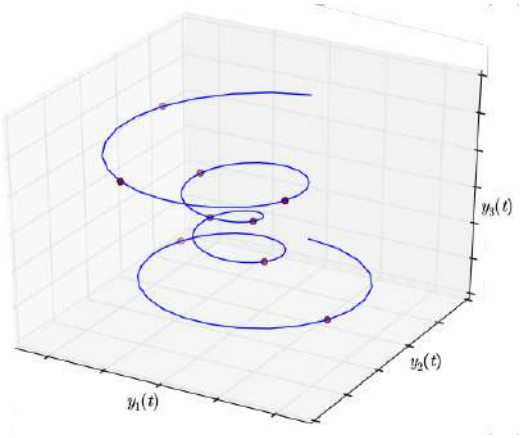
- A *cumulation term*—corresponding to the second term in (2.5); p_c tracks successive updates to the mean, and is, therefore, called an *evolution path*. The cumulation term is a rank 1 term that accounts for the general "direction" that the algorithm has moved.

Besides the covariance matrix Σ_i, CMA-ES also uses a *step-size* σ_i that is updated as the algorithm progresses (lines 8 and 9). The step-size is used, among other things, to avoid that the samples collapse too soon. Its update makes use of a conjugate evolution path, p_σ, which is compared with the direction-independent length, $\mathbb{E}\|\text{Normal}(0, I)\|$.

Using CMA-ES in optimization has benefits in that this algorithm doesn't need vast parameter tuning [1] and is suitable for working with robotic tasks; the robotic problem task generally has a loftier performance using BBO than using Reinforcement Learning [124].



(a) Critical points and segmentation of a simple 1D trajectory.



(b) Critical points and segmentation of a 3D trajectory.

Figure 2.4: Illustration of the segmentation module using critical points.

2.2.1 Constrained CMA-ES

This method extends to the CMA-ES method for constrained optimization problems. Suppose that we want to optimize a function $J : \mathbb{R}^P \to \mathbb{R}$, while ensuring that

$$h_j(w) \leq 0, \quad j \in \{1, \ldots, m\},$$

for some set of functions $h_j : \mathbb{R}^P \to \mathbb{R}$. The above formulation is quite general, in that it also allows for the consideration of equality constraints of the form $g(w) = 0$, simply by requiring, simultaneously, that $g(w) \geq 0$ and $g(w) \leq 0$. In our settings, the constraints will typically arise from the coordination required between the two manipulators, although other constraints may also be considered.

At each iteration of CCMA-ES, the algorithm samples a number of tentative points and verifies whether these points verify the constraints. If it does, it proceeds as in standard CMA-ES. If not, the new points are used to align the covariance matrix with the violated constraints, forcing the algorithm to produce samples that eventually verify all constraints.

To describe the algorithm in further detail, let us again consider the three steps outlined in the high-level description of CMA-ES, in page 16. CCMA-ES keeps a *fading record* $v_j \in \mathbb{R}^P$ for each constraint $h_j, j = 1, \ldots, m$. These records are updated to trace (with forgetting) which constraints are violated by the different samples, and to update the covariance matrix so that it remains aligned with the constraint surfaces.

CCMA-ES can then be summarized as follows [9].

1. **Sampling:**

   - Sample K points $\{w_{i,k}, k = 1, \ldots, K\}$ from a multivariate Gaussian, $w_{i,k} \sim \text{Normal}(\mu_i, \sigma^2\Sigma_i)$, with $\mu_i \in \mathbb{R}^P$ and $\Sigma_i \in \mathbb{R}^{P\times P}$;

2. **Verifying constraints and resampling:**

   - Update records v_j for which h_j is violated for at least one sample $w_{i,k}$;
   - Adjust covariance matrix Σ_i according to the records v_j;
   - Re-sample the points that violate constraints and repeat, until no violations are observed;

3. **Evaluating:**

   - Compute $J(w_{i,k})$ for $k = 1, \ldots, K$;
   - Select the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$;

4. **Adaptating:**

   - Update the mean μ_i to the weighted average of the best K_0 samples from $\{w_{i,k}, k = 1, \ldots, K\}$ selected in the previous step;
   - Update the covariance matrix Σ_i to increase the likelihood of previously successful search steps.

For each point $w_{i,k}$ such that $h_j(w_{i,k}) > 0$, v_j is updated as

$$v_j = (1 - c_v)v_j + \frac{c_v}{\sigma}(w_{i,k} - \mu_i), \tag{2.6}$$

where c_v is an adjustable scalar for fading rate. Moreover, writing

$$a_j(w) = \begin{cases} 1 & \text{if } h_j(w) > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad a_0(w) = \sum_{j=1}^{m} a_j(w),$$

The covariance matrix is updated according to

$$\Sigma_i = \Sigma_i - \frac{\beta}{a_0(w_{i,k})} \sum_{j=1}^{m} a_j(w_{i,k}) \frac{v_j v_j^T}{\|v_j\|^2}. \tag{2.7}$$

In a nutshell, CCMA-ES is obtained from CMA-ES (Algorithm 1) by including an intermediate step where all samples are checked against all constraints and, if a violation is detected, the covariance matrix is instantly adjusted and a new sample generated. This process is repeated until no constraints are violated, ensuring that the solution obtained by CCMA-ES verifies all constraints [7].

2.3 Critical Points

We conclude by introducing the concept of *critical point of a trajectory*, which plays a key role in several of the contributions of this thesis. It is indispensable to use critical points in our work, as they were relied upon in the segmentation process because they are points in the path that contain essential and sensitive information and provide the general shape of the path. It also benefits alignment in online observation while observing a part of the trajectory. It also saves time as it shortens the search and comparison process on critical points instead of examining all track points.

Given a trajectory $y(t)$, with $t \in [0, T]$, a critical point is a point that lies between two segments of the trajectory, each corresponding to a "homogeneous" movement, in a sense. We follow Meier et al. [83] and identify critical points as *minima in velocity and acceleration*. In particular, letting

$$h(t) = \|\ddot{y}(t)\|^2 + \|\dot{y}(t)\|^2, \tag{2.8}$$

where $\dot{y}(t)$ is the velocity of the trajectory, we choose the local minima of $h(t)$ as critical points. Figure 2.4 illustrates the process of choosing the critical points in some simple trajectories. In Chapter 4, we use critical points to segment long trajectories; in Chapter 5 we use critical points to perform movement recognition. We defer to Chapter 3 the justification for using critical points to determine the segments comprising a given trajectory.

Chapter 3

Related work

In this chapter, we provide a survey of the work most directly related to the thesis, framing our contributions in the landscape of existing research. We also discuss how our work builds on and improves existing work. The discussion of related work is organized into different subsections, roughly corresponding to the different areas in which our work contributes.

Section 3.1 provides a (necessarily narrow) overview of noteworthy works involving human-robot collaboration, which can be seen as the overarching motivation for our research. Sections 3.2 and 3.3 discuss several existing approaches that address each of the individual components of our framework for complex movement generation (Chapter 4) and coordinated movement generation (Chapter 6). Our main contribution, in this context, is the combination of these different modules into a unified end-to-end framework for robot motion segmentation, representation, combination, and optimization.

Finally, Section 3.4 discusses the literature on trajectory recognition and prediction, relevant for our work in Chapter 5. In this line of work, our contribution is a novel method for trajectory recognition and target prediction that circumvents the need for temporal alignment.

3.1 Collaboration with Robots

In this section, we discuss several works that address collaboration between humans and robots. Although our work does not consider actual scenarios where robots work with or interacting with humans, the research problems addressed herein—and outlined in Chapter 1—are all concerned with fundamental skills expected of a robot that is prepared to interact and collaborate with humans. The present discussion of related work discusses a number of works featuring scenarios where the work of this thesis could have a meaningful impact.

◇

As robots enter our lives, we—as researchers in robotics—need to promote/facilitate the collaboration between humans and robots [126]. Such collaboration involves not only understanding the *mechanisms of collaboration*, but also understanding its challenges, its impact on the working environment, and the tasks in which collaboration can take place.

In parallel, it also involves developing robotic platforms that can accommodate all these factors while carrying out new and complex tasks [27]. Natural human limitations—often accentuated by external factors such as health, physiological condition, and the surrounding environment—highlight the relevance of collaboration between humans and robots. Robots can *assist* humans in their daily activities, fostering consistency and quality, and minimizing the impact of changing human or environmental factors on the performance of the human [95]. But for robots to join in and participate of everyday human activities, we must provide them with both the hardware and software necessary for them to succeed in new tasks and acquire new knowledge, as non-expert users cannot, in general, *program* a robot to do so.

The ability of a robot to *learn from demonstration* [8, 66] provides a natural means to succeed in new tasks and acquire new knowledge and one which a non-expert user can generally use [115]. Learning from demonstration has been widely investigated in the robotics literature, and different approaches rely on different inputs from the human demonstrator (e.g., teleoperation, kinesthetic teaching, etc.) [8, 66]. However, at a higher level, the ability of a robot to *recognize* movements performed by a human user as movements already known by the robot, or the ability of the robot to *predict/infer* the end-goal of such movements is a fundamental skill that can push the paradigm of learning from demonstration one step further. Collaborative tasks between humans and robots, for example, also highlight the importance of a robot being able to recognize the movement of human collaborators and predict their end-goal. Motivated by such scenarios of human-robot collaboration, in Chapter 5 we address the problem of movement recognition and prediction. We consider how a robot can leverage its prior knowledge to recognize a movement that it has seen before—either on its own or as part of a complex trajectory.

There are different methods to address recognition problems, depending on the nature of the recognition itself. For example, distinguishing gestures requires access to gesture signs (as samples) that can be used, for example, to train a recurrent neural network [90]. As another example, to recognize sounds (instruction commands), we can use a hidden Markov model (HMMs) [108], which also requires the pre-existing training samples. In yet another example, Luo et al. [68] used unsupervised learning to learn Gaussian Mixture Models to recognize a trajectory and Gaussian mixture regression to

predict the target of the trajectory. Much like the previous methods, Luo et al. [68] also requires many samples and a training phase. Our contribution sets itself apart from such data-intensive approaches as we build on a set of pre-learned movement primitives, and recognition is done directly on top of such builtin knowledge, without requiring an explicit "training phase".

Teaching a robot by demonstration, however, is not without its challenges. For example, there is great interest in exploiting robots in space discovery. Robots can do several tasks in space—like exploration—and help humans by carrying out the most dangerous tasks. Therefore, the National Aeronautics and Space Administration (NASA) designed several robots specifically for these tasks, such as *Robonaut* to assist astronauts [30]. NASA recently sent *Rover* to Mars to explore and perform several additional tasks [41]. Collaboration between robots and humans has also been explored in industry [131], healthcare [100], assisted living [4], and other activities [85, 107]. Robots are used to increase performance consistency, or to operate in scenarios where the conditions are harsh, dangerous, or unsuitable for humans. However, in all these application areas, it may be challenging to provide an accurate demonstration to teach a robot, particularly since the conditions that the robot will encounter at execution time may be difficult to foresee at demonstration time. The robot should, therefore, be able to leverage its prior knowledge (for example, movements it has previously learned) and adapt them to environments and circumstances that differ from those for it was originally programmed. In Chapter 4 we propose a framework that allows a robot to leverage its prior knowledge—i.e., movements it has learned before—and compose them and optimize them to its current circumstances.

More broadly, leveraging the collaboration between humans and robots raises many important issues [21]. The safety of humans is still, perhaps, the most crucial one, and having a robot engage in a collaborative task with a human must ensure safety for the human [131]. For this reason, Merckaert et al. [84] impose constraints on the controller of a robot operating in a dynamic environment to achieve collaboration between humans and robots while ensuring human safety. The paper measures the effectiveness of the proposed approach in terms of collaboration—both related to humans and robots—by reporting the task execution time and the energy consumed by the robot.

However, in human-robot collaboration, other indicators are essential, such as the

emotional and cognitive components of behavior, speaking, and communication [22, 28]. Designing robots for collaboration with humans involves reasoning about these indicators both in terms of hardware and software. For example, Chandrasekaran and Conrad [21] discuss the types of robots and their embodiment, which may vary depending on the tasks they are designed for and the nature of the people they will collaborate with to accomplish their goals. The medical robots that perform surgical operations have shapes closer to the doctor's hands with high control capabilities, accuracy and comfort, and they have specialized communication methods to collaborate with doctors [73], or even they use robotic machines to support disabled people in physical activities [136]. We also note the difference between robots dedicated to rescuing and firefighting [79] with social robots that communicate and interact with humans [13]. Taking such robot diversity into consideration, in this thesis we adopt the widely applicable representation framework of dynamic movement primitives. Although our application scenarios focus on manipulation tasks, the ideas presented herein can be more broadly applicable and are not specific to a robotic platform.

In collaborative tasks, team members work together to accomplish a particular goal. Such teams may be composed of humans and robots, possibly more than one robot or more than one human. Collaboration towards the common goal requires coordinated actions between the members according to the stages of task completion and their skills. Usually, the skills of team members are complementary to each other to accomplish the required task [14]. In human-robot collaborative tasks, the human mainly defines the goal of the task. The robot needs to know or understand the human's intentions, which can be communicated by the human to the robot, either explicitly or implicitly. In the case of explicit communication, humans can use speech, gesture, actions, or haptic signals, for example, and the robot requires interpretation abilities [90]. In the case of implicit communication, the robot needs to predict the human's intentions, so the robot requires prediction abilities [14] that, as mentioned above, we address in Chapter 5.

In collaborative tasks, the uncertainty about the surrounding environment and its dynamics, as well as the complexity of some movements, change between tasks and among peers. For the robot to overcome such changes, it must adapt during task execution. Machine learning can be used as an effective toolset to reduce the impact of the aforementioned changes. Using supervised learning, unsupervised learning [17], or

reinforcement learning [42], robots can learn how to adapt to humans and complete its task [14]. Reinforcement learning (RL) is particularly relevant for adaptation [132]. The optimization method used in the thesis, CMA-ES (Chapters 4 and 6), can be seen as an form of RL that performs direct policy optimization in a model-free setting. Previous works used, for example, deep reinforcement learning to train a robot how to collaborate with humans [see the works of 25, 110], although such approaches were, again, data-intensive.

It is also possible to teach the robots how to collaborate using imitation learning, which will save a significant effort in the training phase [40]. After learning, the robots can improve their behaviors during the task using interactive learning [86]. Teaching a robot can also rely on augmented reality (AR) technologies. AR enhances communication between humans and robots by combining the physical and virtual worlds. It can be used both to train the robot and design scenarios that can be difficult to deploy just using the real world. There are many benefits to using AR in human-robot collaboration (HRC); for more details, we refer to the work of Green et al. [30]. The previous reviews show the importance of our thesis and our contribution to the field we are searching in collaborating with robots.

Before finishing this section, it is important to establish what we mean by *collaboration*, *cooperation*, and *coordination*, as these are central concepts appearing in the context of joint tasks involving multiple agents (such as human-robot collaborative settings). These terms define the types of relation between the actors in the joint task, as well as their functional layout in terms of the task.

If all actors share the same objectives, and there is a common outcome for the task, we refer to such situation as a *collaborative task*. In such situation, all actors "own" the task. Conversely, in a *cooperative task*, there is one "owner" for the task; the other actors may have distinct objectives but still provide help or a service to the "task owner" (i.e., they cooperate) [51]. Finally, *coordination* usually refers to collaborative situations, but where the main focus is on the process of action/subtask selection. In a sense, coordination can be seen as as a "lower-level" collaboration [51].

◇

So far, we discussed the importance of human-robot collaboration, and introduced

some important challenges that arise in the context of such collaboration. In the continuation, we discuss existing work on each of the challenges addressed in the thesis, beginning with movement processing (segmentation, combination, etc.).

3.2 Movement Processing

The contributions of this thesis—as depicted in Fig. 1.2 (page 6)—build heavily on a library of pre-learned movements, represented as dynamic movement primitives. The library is used for movement generation (Chapters 4 and 6), movement recognition, and prediction (Chapter 5). This section discusses works on *movement segmentation*—used to build novel primitive movements from demonstration—and *movement combination*—used to build novel complex movements from simpler ones.

3.2.1 Segmentation

Movement segmentation approaches can be roughly divided into three families of methods [133]. *Boundary detection* approaches rely on the acceleration, velocities, and curvature of the movement to detect segment start and end points [80, 111].

Sliding window methods, in contrast, scan the whole trajectory, splitting it in overlapping windows, where each window is then classified as belonging to one of several possible movement types. Segmentation is then obtained by joining together consecutive windows classified as belonging to the same movement type [26, 43, 44]. Sliding window methods are generally computationally intensive and cannot be used at training time (i.e., when movement classification is not yet available).

A third type of segmentation methods are *grammar concatenation approaches*, which use dependence structures (such as hidden Markov models, HMMs, or conditional random fields, CRFs) to model the transition dynamics between movement segments to detect segmentation points. For example, Lv and Nevatia [69] use HMMs, and Morency et al. [88] use CRFs as the underlying structure used to detect segmentation. While less computationally expensive than sliding window methods, grammar concatenation approaches typically require large amounts of data to learn the dependence structures used for segmentation.

The segmentation method used in this thesis falls into the first category, and relies on

the detection of *critical points*, as defined in Chapter 2. Critical point-based movement segmentation provides several important benefits for the applications envisioned in this thesis [133]:

- Unlike grammar concatenation approaches, it does not require training data, using only information directly available in the movement itself. Such property renders critical point-based segmentation particularly suitable in the context of Chapter 4, where segmentation is used to learn new primitive movements from a demonstration;

- Unlike sliding window methods, it is simple and computationally efficient, making it suitable for real-time applications such as movement recognition (addressed in Chapter 5.

Considering the more specific context of learning from demonstration, several approaches have been proposed for movement segmentation, which can also be classified as belonging to one of the families discussed above.

For example, Meier et al. [82] segment complex movements into a sequence of simpler movements by recognizing individual segments of the larger trajectory as instances of the movements in a given library of DMPs. Meier et al. [83] use the idea of critical points to segment an observed movement and then perform recognition. Both approaches use the movement features to determine the segment boundaries, and roughly fall into the first category above.

In another work, closer to the category of grammar concatenation approaches, Niekum et al. [94] propose the use of a Bayesian non-parametric model (an auto-regressive hidden Markov model) to describe the observed demonstration, and use the estimated latent discrete states to segment the trajectory into different sub-trajectories. Each sub-trajectory is then represented as a DMP and used to build a library that can latter be used for planning.

Our approach in Chapter 4 also uses the trajectories provided by the demonstrator to build a library of DMPs that can then be used for planning. Our segmentation approach builds on the one proposed by Meier et al. [83], in that we use velocity and acceleration information to determine critical points, used for segmentation.

3.2.2 Combining Movements

One of the benefits of using motor primitives is the fact that by combining multiple motor primitives, we can create one larger motor primitive to perform a complex task [115].

We can look at motion combination from at least two different perspectives. On the one hand, motions can be generated by *mixing* motor primitives. For example, Mulling et al. [89] propose an approach that learns how to build mixtures of motor primitives to more effectively hit a moving table tennis ball. Their work proposes the *mixture of motor primitives* (MoMP) method that learns how to combine multiple DMPs from a library, in order to achieve complex tasks. Much like our approach in this thesis, the MoMP method requires a library of movement primitives and, at execution time, it builds new movements that depend on the external stimuli. They employ a gating network to generate weights for each motor primitive in the library, and these weights control the generation of the resulting movement.

On the other hand, complex motions can be built by *sequencing* motor primitives [57, 58, 92]. For example, Kulvicius et al. [58] propose a simple approach in which two DMPs can be combined sequentially in a smooth manner by adding a common term that drives both DMPs in the vicinity of the transition points.

Pastor et al. [99] proposed merging DMPs depending on the DMP velocity values. They define a threshold for the velocity; when the first DMP velocity is on this threshold, their approach initiates the next DMP. This yields a sequence of DMPs that, however, can lead to discontinuous movements that seem erratic.

The method of Kober et al. [50] requires the definition of an intermediate velocity and an intermediate target to combine two DMPs. This method is used to reach the target at a particular velocity, so they modify the DMP form to achieve this purpose. It is used for some movements like hitting the ball, where the movement should reach the ball position at a specific velocity to accomplish its goal. So they determine an intermediate velocity, which is reached at the end of the first DMP and will be the starting velocity of the second DMP.

In Chapter 4 we combine DMPs following Kulvicius et al. [58], because the resulting movement will be smooth and without any discontinuities, and does not require the computation of intermediate velocity terms.

3.3 Trajectory Optimization

In this work, we describe the robot motion using dynamic movement primitives (DMPs). DMPs are one among different ways to represent movement primitives, and are convenient in that they yield smooth motions that can be easily learned and modulated, as discussed in Chapter 2. Such representation is also convenient as they allow the combination of different MPs, with parallel or sequential execution, to carry out complex tasks [62, 97].

There are other alternative representations for movement primitives—for example, probabilistic representations that capture how likely a given trajectory is under a movement primitive. *Probabilistic movement primitives* (ProMPs) are also easily learned and modulated using standard probability manipulation operations. They can also be combined into new primitives, taking advantage of the common representation [97].

In many scenarios, however, we may want to optimize a particular movement after being learned by demonstration. Such optimization can be done using standard reinforcement learning approaches. For example, Peters et al. [103] proposed the *natural actor-critic* algorithm, which has found significant application in robot learning [49, 102]. Bhatnagar et al. [15] further discuss natural actor-critic algorithms and their relative advantages.

Other approaches use gradient-based reinforcement learning to adapt/adjust pre-learned robot movements [130], but a significant volume of work has considered the use of more general *black-box optimizers* to adjust the parameters of pre-learned movements [34, 112, 119]. For example, Heidrich-Meisner and Igel [34] use black-box optimization to find the parameters of a trajectory that can maximize a fitness function evaluating different measures of quality of a given trajectory. We refer to Stulp and Sigaud [123] for a comparative view of different approaches to robot movement optimization. In this thesis, we adopt the CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy) algorithm for trajectory optimization [82].

3.3.1 Optimize and Coordinate Multiple DMPs

Many complex manipulation tasks require coordinated action from multiple robots, which in turn requires synchronization and coordination at the motor level, if the task

is to be successfully completed [98]. For example, in robotic assembly lines, several robotic manipulators operate in a coordinated fashion to complete their assigned task, such as welding or assembling a part [2]. As such, the problem of coordinated control has been widely investigated in the literature [59, 64, 104, 135].

Other examples can be found in locomotion, where the movement of the multiple legs of a robot must be adequately synchronized and coordinated to ensure successful movement of the robot. Movement in locomotion are often *rythmic* and can be generated in a synchronized fashion using *central pattern generators* [36], a concept borrowed from the theory of motor control in animals [78].

However, as robots move from industrial environments and labs into our homes and everyday lives, it becomes increasingly important that they are able to *learn* new tasks from non-expert users. As argued in Section 3.1, learning from demonstration has been proposed a fundamental skill expected of robots for everyday use [8].

In Chapter 6 we are interested in tasks that require the coordinated movement of two manipulators, and in which the movement to be performed by the robot has previously been taught by demonstration. We consider the situation where the user may not have the ability of providing a demonstration for the two manipulators. This can be the case, for example, if the two manipulators are complex and difficult to handle simultaneously by a single user. Instead, the robot is taught the basic movement in a single manipulator through kinesthetic teaching, and must then determine how to adapt the learned movement to the task at hand, involving the coordinated movement of two manipulators. The original movement taught by demonstration by the human user is represented as a DMP [39], which is then optimized and executed synchronously by the two manipulators taking into consideration the specific task constraints. The DMPs canonical system acts as a common phase signal, ensuring synchronization [114].

Like our work, several previous approaches take advantage of DMPs shared canonical system to facilitate coordination and synchronization. As an example, DMPs have widely been used as central pattern generators [91]. In a different work, Kormushev et al. [56] propose an approach where a coordinated movement represented as a single DMP is initially learned by demonstration and then optimized using reinforcement learning. Their approach uses a *coordination matrix* that ensures that no mis-coordinations are introduced in the optimization problem. Colomé and Torras [23] extend the previous

approach to problems with many dimensions, using linear dimensionality reduction to render the optimization more sample efficient.

In Chapter 6 we build on the framework from Chapter 4 and consider manipulation tasks involving two manipulators.[1] Our approach is related to the aforementioned works that perform optimization on top of a DMP learned by demonstration. Importantly, unlike those approaches, the DMP learned by demonstration is only used to define the *shape of the movement*, and not to provide an initial, coordinated approach [5, 23, 55]. Instead, we take advantage of the natural symmetry in the two manipulators to execute the same motion in both manipulators, and use a constrained optimization approach to ensure coordination. In this sense, our approach is also related to classical control-based approaches that relied on constrained optimization to ensure coordination [2, 135], as well as path-planning-based approaches to coordinated movement [129].

3.3.2 Obstacle Avoidance

Realistic environments are variable and subject to many changes. Since we expect robots to work with us in realistic environments, their operation is directly affected by these diverse conditions and restrictions. Among these changing conditions, fixed and moving elements act as obstacles for the robot. The robot must avoid them and ensure that its basic tasks are completed without collision or damage. The process of avoiding collision must be transient and, if possible, not end or change the tasks assigned to the robot. All of the above require a system that can dynamically adapt to changing environmental variables.

Hoffmann et al. [35] propose a modification to the differential equation of the DMP by adding a term to velocity and position dynamic equations. The new velocity is computed applying a rotation matrix on the current velocity. The rotation matrix, in turn, is designed to modify the DMP to prevent hitting an obstacle, based on the angle between the velocity vectors of the current DMP and the obstacle. This approach operates with a point-mass obstacle, which is unsuitable to work in realistic environments, where the obstacle is not properly represented as a point-mass. To use such approach in our work, we ought to add a considerable safety margin to avoid any

[1] We note that, although we do not explore such a setting in this work, the proposed approach could, in principle, be extended to scenarios involving more than two manipulators.

possible collision, which is inappropriate.

Khansari-Zadeh and Billard [46, 47] propose a related method that computes a *modulation matrix*, which modifies the DMP to avoid the obstacles. The modulation matrix is computed in real-time and updates the DMP during the execution. Because using a modulation matrix is practical and efficient during execution time, we use it in our work in Chapter 4.

3.4 Recognition and Prediction

One of the essential challenges in human-robot collaboration is the ability of the robot to recognize what the human partner is doing and move accordingly. In human-human collaboration, each person observes and "intuitively" recognizes the partner's motion, allowing the person to act in the best possible way. Humans are naturally flexible concerning a partner's actions and are able to adapt, fostering high-level collaboration and inter-partner trust [63].

For such high-level collaboration to be possible, robots must also recognize and adapt to the observed movement of a human partner. Several works in the literature address the problem of trajectory recognition in the context of human-robot interaction, and many of these works use various forms of the hidden Markov model (HMM) [60, 96, 128]. However, such methods require large data-sets for training, which are costly to obtain in human-robot collaboration scenarios. As an example, Yoon et al. [134] proposes an approach to recognize gestures from video data. They describe a gesture with several invariant features to transition, rotation, and scaling (e.g., location, velocity, angle of motion) and use these data to build an HMM to recognize various hand gestures. In another work, Black and Jepson [18] use a condensation algorithm to compute a probability distribution over a set of models given the input trajectory. The distribution is computed from the difference between the amplitude of the input and model motions, after adequate time scaling. The computed distribution can then be used to combine the prediction of the different models. The same approach was also tested in a face recognition scenario to recognize facial expressions from facial motions [19].

Also related is the work of Kherallah et al. [48], where stroke movement is used to recognize Arabic digits. In particular, the authors propose using Beta-elliptical

representations to describe the writing motion and then use a neuro-fuzzy system for digit recognition. Maeda et al. [72] introduce *interactive probabilistic movement primitives* (I-ProMPs) to represent collaborative human-robot motions. In this work, the authors assume that all motions have a constant rate of change and, therefore, time alignment can be achieved by simple scaling. This reduces recognition to the estimation of the human action phase, circumventing the need for time alignment.

Dermy et al. [24] use a set of pre-learned motor primitives (MP) for recognition. The use of an MP library is similar to our approach. However, unlike our work, the authors assume that the observed movement is an exact match to one of the pre-learned MPs, thus disregarding some critical challenges in motion recognition (such as time alignment).

The I-ProMPs of Maeda et al. [71] represent collaborative trajectories between a human and a robot by using a probabilistic model, capturing through standard regression a Gaussian representation for both the human and robot trajectories. At recognition time, the observed trajectory—corresponding to the human action—is used to compute the corresponding robot trajectory by a simple probabilistic operation (conditioning). However, in order to build a robust I-ProMP model, a rich set of demonstrations is required. Their approach is also sensitive to speed variability. In Chapter 5 we propose a novel approach that is robust to speed changes and, moreover, does not require the observation of the entire human trajectory to compute the robot's movement. Instead, it uses a prediction made from the partial observation of the movement as the target for the robot's own movement.

Perhaps closest to our work is that of Pérez-D'Arpino and Shah [101]. In their approach, they also use a library of learned movements and use time series analysis to estimate the target of the observed motion. The estimation requires time alignment of the observed trajectory to those in the library, which is achieved through online dynamic time warping (DTW). Since this process is time-consuming, they use a parallel architecture to perform the DTW with the different motions in the library. However, they do not support target modulation, assuming that the observed motions are simply perturbed versions of those in the library. Along the same lines, Mainprice and Berenson [74] also perform recognition from a library of known movements. They use *occupancy* for movement recognition, overcoming the need for time alignment but, once again,

assuming that the observed motions are perturbed versions of those in the library.

Finally, other works have explored motion prediction for human-robot interaction. Mainprice et al. [75] consider prediction in the context of a single task, to avoid interference between the robot's motion and that of the human. Prada et al. [105] consider handover tasks, where the movement of the robot is adjusted in run-time to that of the human.

In Chapter 5, we propose an approach for trajectory recognition and prediction that, given a library of previously observed movements, identifies a partially observed trajectory from among those in the library, predicting its (unobserved) target point. The fact that our approach does not require the entire trajectory to be observed is fundamental, as our approach can use it in run-time for actual human-robot interaction scenarios. Our approach is also able to recognize modulated movements, i.e., movements that share the shape of those in the library but which are aimed at a different target.

3.4.1 Compound Movement Recognition using DMPs

During infancy, humans build a "library" of new movements by learning to combine simpler movements [45]. This method of learning complex movements allows humans to learn how to carry out complex tasks. The ability to combine simple motions into more complex ones has also been a topic of intense research in the robotics community [58, 89, 92], and is also a key ability of our framework, as discussed in Chapter 4.

In Chapter 5, we address the problem of movement recognition and prediction, considering not only *single, primitive motions*, but also *compound motions*. The ability of a robot to break down a movement into its simpler components is fundamental for recognition purposes: suppose the robot wants to replicate the observed movement instead of learning the new movement from scratch. In that case, it can use simpler movements already learned (and recognized by observation) and compose them to perform a more complex one.

All works on recognition discussed in Section 3.4 assume that the observed trajectory is atomic and corresponds (or is close) to one of a set of pre-learned trajectories. Such methods are not designed to deal with compound trajectories. Some works do address complex movement recognition, assuming that the observed movement is obtained as the combination of *complete* simpler motions [24]. Maeda et al. [71] use a probabilistic

model (I-ProMPs) to distinguish movements, whereby the observed path is approached with the closest human action to it. Ma et al. [70] work at the task level: the observed movements of the human body are divided into parts that may overlap. An algorithm is built to capture these parts by extracting them from video frames. A bag-of-words representation is then used to distinguish the partial movements, using the temporal and spatial relationship between the parts.

Our proposed approach, discussed in Chapter 5, does not assume that the observed motion is composed of full atomic movements. Instead, we accommodate the case where the observed movement arises as to the composition of several *movement segments*, possibly coming from different movement primitives. We assume that each such segment is either the initial segment of a new movement primitive or the continuation of the previous segment. Our method identifies whether the observed movement is new or compound and predicts its most likely target, should the current movement be executed to the end. We can use the recognition of the observed movement as the combination of simpler movements in a framework for movement combination, and optimization [58, 67, 122, 123].

◇

Having provided a broad overview of the landscape of current research in the different topics relevant to this thesis, in the following chapter we introduce to our first contribution: an end-to-end framework to learn from demonstration, segment, compose, and optimize movements.

Chapter 4

Learning and Generating Complex Robot Motions from Demonstration

This chapter describes an end-to-end framework that can learn and decompose complex movements provided by a human demonstrator and generate new complex motions. Our approach analyzes the demonstration by the human expert and uses geometric criteria to decompose the observed movement into segments that are stored as dynamic movement primitives (DMPs) in a library. Then, given a new environment configuration, our system autonomously composes different motion primitives to construct an optimized trajectory that meets the constraints imposed by the new environment. Our system is therefore able to construct new DMPs to execute complex motions in environments that differ from the one where the original motions were taught. Our approach is also compatible with existing run-time obstacle avoidance approaches. We illustrate the application of our approach both in simulation and with a Baxter robot.

The key contribution of this chapter is the system as a whole. The added value of the proposed architecture is the combination of all the functionalities in a single framework for learning and generating complex motions from human demonstrations. In addition, the greedy approach proposed to compose atomic DMPs is of interest per se. The use of DMPs greatly facilitates the interconnection of the different components of our architecture, since DMPs are amenable to optimization [122], can easily be combined to generate complex motions [58], and can be exploited to ensure the avoidance of obstacles in run-time [46, 47]. Our framework comprises a library of DMPs and a set of different modules for (i) segmenting a provided demonstration, abstracting the computed sub-trajectories into new DMPs and adding them to the library; (ii) a greedy algorithm that, given a goal point, combines the DMPs in the library into a single complex motion that is then optimized according to provided criteria (length of trajectory, obstacle avoidance, smoothness); (iii) a module that, during execution, monitors the existence of obstacles and reshapes the trajectory to avoid such obstacles in run-time.

4.1 A Framework for Motion Decomposition, Learning and Generation

Our framework is summarized in Fig. 4.1 and comprises several modules for motion decomposition, learning and combination. Underlying all the modules is a common representation of "atomic" robot motions—the dynamic motion primitive, or DMP.

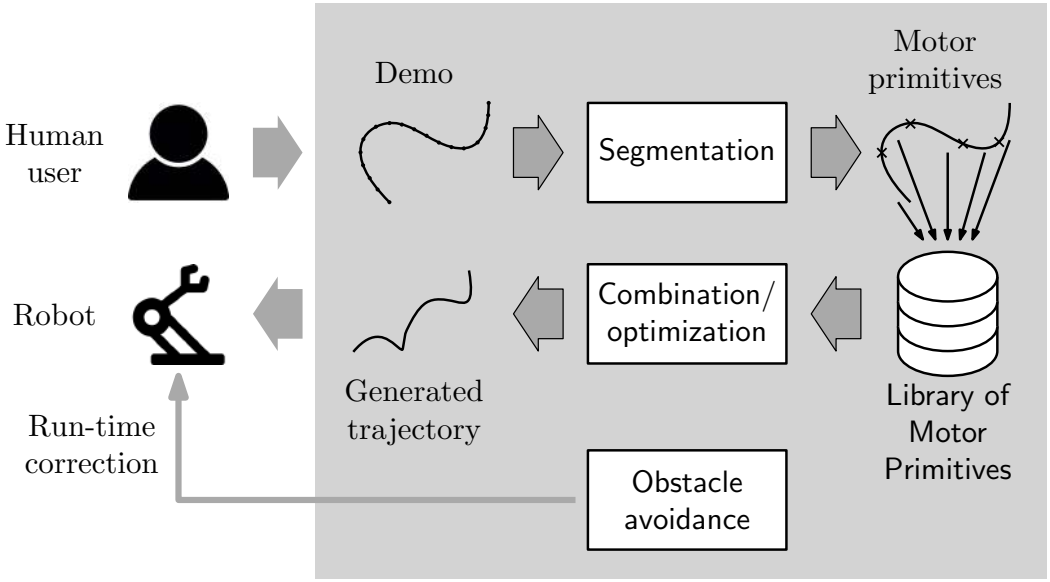


Figure 4.1: Overview of the proposed system architecture for end-to-end learning, decomposing, optimizing and combining robot motions.

In the continuation, we discuss how the different modules of our system are built on top of the DMP representation just described.

4.1.1 Segmentation

Given an initial demonstration $\{y(t), t = 0, \ldots, T\}$, with $y(t) \in \mathbb{R}^P$ for some $P \geq 1$, we segment the trajectory into multiple segments, each of which is then turned into one DMP. To perform segmentation, we follow Meier et al. [83], where the segment points consist of the critical points of the trajectory, and identify such critical points as the minima of the function $h(t)$ in (2.8) (Chapter 2, page 21). Therefore, after computing $h(t)$ for all $t = 0, \ldots, T$, we compute local minima of $h(t)$ and mark those points as final points of a sub-trajectory and initial point of a new sub-trajectory.

After determining the set of sub-trajectories obtained from a demonstration, our system builds one DMP for each sub-trajectory. The learning of the DMPs consists of adjusting the weights of the force function so that the resulting trajectory matches the demonstration as closely as possible [77]. The different learned DMPs are then stored in the library for future use.

4.1.2 Optimization

Given the library of learned DMPs, whenever the user provides the robot with a new target, the robot must now use the learned DMPs to reach the new target while accommodating potential changes to the environment configuration (e.g., new obstacles, etc.). In order to do that, the robot builds a sequence of DMPs that may meet the desired objectives (reaching the target, avoiding obstacles).

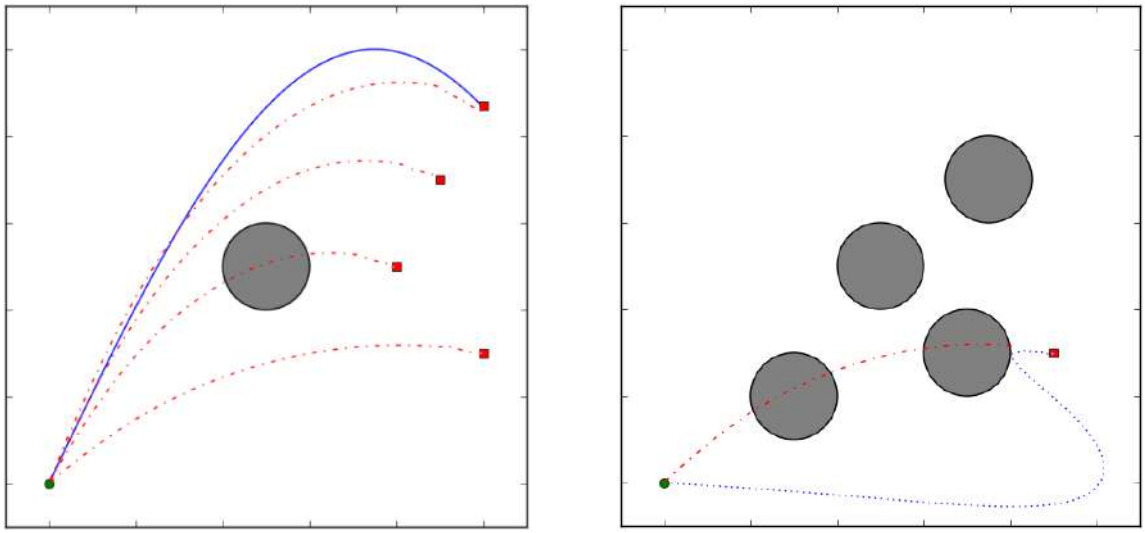
The identification of which DMPs can be used to reach a given target relies critically on the robot's ability to optimize a DMP to reach such target in light of environment constraints. We adopt the Covariance Matrix Adaptation Evolutionary Strategy—or CMA-ES—to that purpose. CMA-ES is a derivative-free optimizer that has been successfully used in robot learning and reinforcement learning [67, 122, 123].

Given a target position y^* and a set of M obstacles, $\{o_1, \ldots, o_M\}$, we define the objective function for evaluating the black-box output for the CMA-ES samples:

$$J = \sum_{t=0}^{T} \left[c_{\text{goal}} \|y(t) - y^*\|^2 + c_{\text{vel}} \|\dot{y}(t)\|^2 + c_{\text{obst}} \sum_{m=1}^{M} \mathbb{I}(y(t) \in o_m) \right], \tag{4.1}$$

where c_{goal}, c_{vel} and c_{obst} weight the relative importance of reaching the goal, avoiding large velocities and avoid obstacles, and $\mathbb{I}(y(t) \in o_n)$ is a binary value that indicates whether, at time step t, the trajectory $y(t)$ hit obstacle o_n.

Given a learned DMP—encoded as an N-dimensional weight vector w that defines the force term in the DMP representation—CMA-ES proceeds, at each iteration i, by sampling a set of new candidate weight vectors, $\{w_{i,1}, \ldots, w_{i,K}\}$, from a multivariate Gaussian distribution centered around the current weight vector, μ_i, and with covariance Σ_i. The trajectories resulting from each candidate weight vector $w_{i,k}, k = 1, \ldots, K$, are evaluated according to the cost function J in (4.1), and a new weight vector μ_{i+1} is computed from the weighted mean of the candidate vectors. The covariance matrix Σ_i is also adapted accordingly (we refer to Chapter 2 for further details on CMA-ES). Figure 4.2 illustrates the result of the optimization process in a simple environment with an obstacle and different targets.



(a) (b)

Figure 4.2: Learning and optimization of simple DMPs from demonstration. A solid blue line corresponds to a demonstration; red dashed-dotted lines correspond to the modulated DMP to different targets. Since the DMP is unaware of the existence of obstacles, the modulated trajectory goes through the obstacle. A blue dotted line corresponds to an optimized DMP, given the shaded obstacles. CMA-ES is effectively able to avoid the obstacle while reaching the target.

4.1.3 Combination

We now discuss the combination module in our framework. Such module can be broken down in two components:

- A first component that selects from the library the DMPs to be combined;
- A second component that adjusts the selected DMPs to obtain a single, smooth trajectory.

We describe each of these two components in detail.

Selecting the DMPs to Combine

Our approach greedily selects a sequence of DMPs that drives the robot from an initial state y_0 to a target state y^* while respecting the constraints imposed by the environment, as illustrated in Figure 4.3. The selection process relies critically on the optimization approach described in Section 4.1.2 and works as follows.

Let y_0 denote the origin of the desired motion, and y^* the desired target. Using CMA-ES, we optimize each DMP in the library to move from y_0 to y^* while respecting

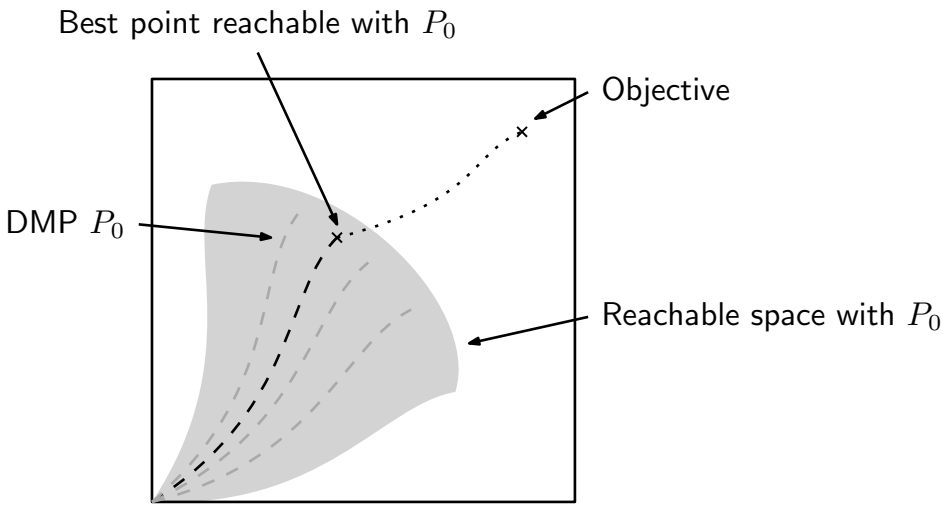


Figure 4.3: Greedy approach to determine which DMPs to combine in order to achieve a new target, while accommodating a new environment configuration.

the constraints imposed by the environment. The best DMP—in terms of the associated cost J—is then selected. If the target is reached the process concludes. Otherwise, the ending point of the selected DMP is set as the new origin and the process repeats until the target is reached.

Combining Multiple DMPs

Once a sequence of DMPs is identified that yields a combined trajectory from y_0 to the target y^*, it is necessary to combine these DMPs together to yield a novel trajectory, as smooth as possible. We adopt the approach of Kulvicius et al. [58].

Given the sequence $\{m_1, \ldots, m_D\}$ of DMPs, selected as described above, the combined trajectory runs for an extended time-frame

$$T' = \sum_{d=1}^{D} T_d,$$

where T_d is the length of the dth selected DMP. The forcing term for the combined trajectory is now composed from $N \times D$ basis functions, where basis function $\Psi_{n,d}$ is now a Gaussian kernel centered around

$$c_{n,d} = \begin{cases} \frac{T_1}{T'} \cdot \frac{n-1}{N-1} & \text{if } d = 1 \\ \frac{T_d}{T'} \cdot \frac{n-1}{N-1} + \frac{1}{T'} \sum_{\ell=1}^{d-1} T_\ell & \text{otherwise,} \end{cases}$$

with a width

$$\sigma_{n,d} = \sigma_n \frac{T_d}{T'},$$

where σ_n is the width of the original basis function Ψ_n. The extended set of basis functions now spans the whole time frame T' and the corresponding widths ensure some level of overlap which, in turn, leads to a smoother resulting trajectory. The weight associated with each basis function $\Psi_{n,d}$ is the nth weight of the dth DMP.
Finally, we also modify (2.3) to reflect the dynamics of the successive (intermediate) targets as

$$\dot{y}_{\text{ref}}(t) = \begin{cases} \frac{\Delta t}{T_d}(y_{0,d} - y_d^*) & \text{if } \sum_{\ell=1}^{d-1} T_\ell \leq t \leq \sum_{\ell=1}^{d} T_d, \\ 0 & \text{otherwise,} \end{cases}$$

where $y_{0,d}$ and y_d^* represent the origin and target of the dth DMP. The resulting combined motion is a novel (complex) DMP which can also, in turn, be optimized using the approach described in Section 4.1.2.

4.1.4 Execution-time Avoidance of Moving Obstacles

Finally, during trajectory execution, it is possible to adapt run-time obstacle-avoidance approaches such as the one of Khansari-Zadeh and Billard [46]. The general idea is that the dynamics of the DMP are modified by a *dynamic modulation matrix* M that locally changes the shape of the DMP trajectory to avoid the moving obstacle.

Formally, let us write the DMP as a first-order dynamical system, yielding the following equivalent formulation:

$$\dot{z}_0(t) = -\frac{\alpha_x \exp\{\alpha_x(T' - t)/\Delta t\}}{1 + \exp\{(\alpha_x(T' - t)/\Delta t)^2\}}, \tag{4.2a}$$

$$\dot{z}_1(t) = z_2(t), \tag{4.2b}$$

$$\dot{z}_2(t) = \alpha_y(\beta_y(z_3(t) - z_1(t)) - z_2(t)) + f(z_0(t)), \tag{4.2c}$$

$$\dot{z}_3(t) = \begin{cases} \frac{\Delta t}{T_d}(y_{0,d} - y_d^*) & \text{if } \sum_{\ell=1}^{d-1} T_\ell \leq t \leq \sum_{\ell=1}^{d} T_d, \\ 0 & \text{otherwise,} \end{cases} \tag{4.2d}$$

where $z_0(t) = x(t)$, $z_1(t) = y(t)$, $z_2(t) = \dot{y}(t)$ and $z_3(t) = y_{\text{ref}}(t)$. The dynamical system

in (4.2) takes the general form

$$\dot{z}(t) = f(z(t), t),$$

and describes the unmodified DMP movement. In the presence of an obstacle o, the trajectory of the system is described by the modified dynamical system

$$\dot{z}(t) = M(z, o) f(z(t), t),$$

where M is the aforementioned modulation matrix.

Given an obstacle o centered at z_o, let $\tilde{z} = z - z_o$ for $z \in \mathbb{R}^P$, and let $G(\tilde{z}) = 1$ denote an implicit representation of the obstacle boundary, where G is assumed to be a smooth function $G : \mathbb{R}^P \to \mathbb{R}$.[1] In other words, a point $\tilde{z} \in \mathbb{R}^P$ such that $G(\tilde{z}) < 1$ lies in the interior of the obstacle; while a point $\tilde{z} \in \mathbb{R}^P$ such that $G(\tilde{z}) > 1$ lies in free space. Moreover, given a point $\tilde{z}_b$ in the boundary of the obstacle, let $n(\tilde{z}_b)$ denote the normal vector to the obstacle at $\tilde{z}_b$, given by $n(\tilde{z}_b) = \nabla_z G(\tilde{z}_b)$.

The modulation matrix M is computed at an arbitrary point $\tilde{z}$ in free space by defining a *deflection hyperplane* that is orthogonal to the vector $n(\tilde{z})$—defined for vectors $\tilde{z}$ in free space as $n(\tilde{z}) = \nabla_z G(\tilde{z})$. A basis for such deflection hyperplane is the set $\{e_q(\tilde{z}), q = 1, \ldots, P-1\}$, with each vector e_q defined component-wise as

$$e_{q,p}(\tilde{z}) = \begin{cases} -\frac{\partial G(\tilde{z})}{\partial z_q} & \text{if } p = 1; \\ \frac{\partial G(\tilde{z})}{\partial z_1} & \text{if } p = q \neq 1; \\ 0 & \text{if } p \neq 1 \text{ and } p \neq q. \end{cases}$$

The modulation matrix can now be decomposed as

$$M(\tilde{z}) = E(\tilde{z}) D(\tilde{z}) E^{-1}(\tilde{z}), \tag{4.3}$$

where

$$E(\tilde{z}) = \begin{bmatrix} n(\tilde{z}) & e_1(\tilde{z}) & \ldots & e_{P-1}(\tilde{z}) \end{bmatrix}$$

[1] As in the work of Khansari-Zadeh and Billard [46], we assume convex smooth obstacles.

and

$$D(\tilde{z}) = \begin{bmatrix} \lambda_1(\tilde{z}) & & 0 \\ & \ddots & \\ 0 & & \lambda_P(\tilde{z}) \end{bmatrix}.$$

The values λ_p are defined as

$$\lambda_p(\tilde{z}) = \begin{cases} 1 - \frac{1}{|G(\tilde{z})|} & \text{if } p = 1; \\ 1 + \frac{1}{|G(\tilde{z})|} & \text{otherwise.} \end{cases}$$

and control the amount of deflection that the original trajectory suffers as a function of how close it is to the center of the obstacle.

The dynamic modulation matrix M can now be computed—as a function of $\tilde{z}(t)$—from (4.3) to warp the trajectories of the system at run-time away from the obstacle.

Additionally, as discussed in [46], it is possible to adjust both the reactivity of the deflection and the "safety margin" around the obstacle through two simple adjustments during the computation of M. Specifically, we can take $\eta \in \mathbb{R}^P$ such that $\eta_p \geq 1, p = 1, \ldots, P$, and use the modulation matrix

$$M(\tilde{z}) = E(\tilde{z}_\eta) D(\tilde{z}_\eta) E^{-1}(\tilde{z}_\eta),$$

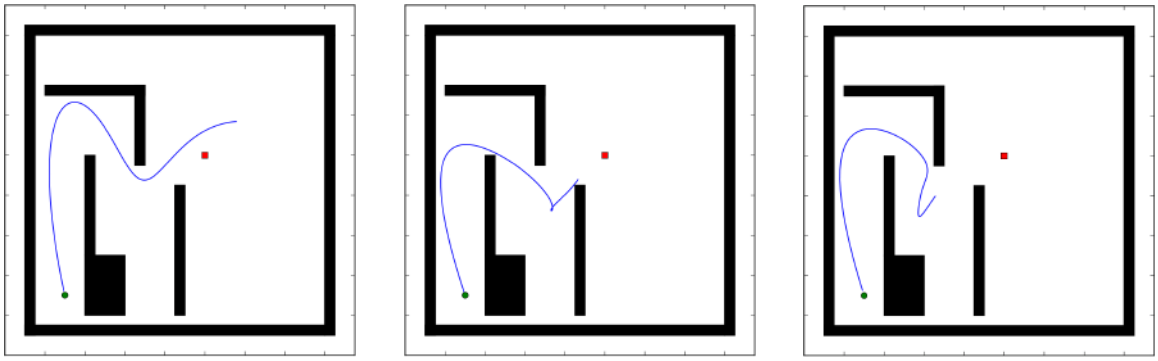
where $\tilde{z}_\eta$ is the vector obtained from $\tilde{z}$ by setting its pth coordinate to $\tilde{z}_{eta,p} = \tilde{z}_p / \eta_p$. The scaling of $\tilde{z}$ by η effectively "inflates" the object o, and we can use η to control the desired safety margin. On the other hand, we can instead consider

$$\lambda_p(\tilde{z}) = \begin{cases} 1 - \frac{1}{|G(\tilde{z})|^{1/\rho}} & \text{if } p = 1; \\ 1 + \frac{1}{|G(\tilde{z})|^{1/\rho}} & \text{otherwise.} \end{cases}$$

The positive parameter ρ controls the "reactivity" of the deflection process: larger values for ρ further extend the deflection induced by the obstacle.

4.2 Experiments

We now illustrate our framework both in simulated and real trajectory generation. Our results showcase the ability of our framework to generate complex motions from learned



(a) (b) (c)

Figure 4.4: Several runs of CMA-ES in a cluttered environment. The trajectories are generally unable to reach the target (the red square).

DMPs. We illustrate both the performance of the individual modules and of the system as a whole.

4.2.1 Learning Simple DMPs from Demonstration

We start by showcasing the process of learning a simple DMP from demonstration.

Figure 4.2a illustrates the DMP resulting from a simple 2D demonstration. In the example, the dashed line corresponds to the DMP learned from the solid line. If we change the target of the DMP, the resulting trajectory will maintain the same overall shape, while ending in the desired target, but it will unable to avoid the obstacle.

We note, however, that although the DMP illustrated in Fig. 4.2a was learned in the presence of an obstacle (corresponding to the gray circle), it contains no information regarding such obstacle; therefore, if modulated to a target across the obstacle, the resulting trajectory will not avoid the obstacle, as illustrated in Fig. 4.2b.

The use of CMA-ES readily handles such situation. By including obstacle information in the cost function J, CMA-ES departs from the learned DMP and construct a new DMP trajectory that is able to successfully reach the target and avoid the obstacles, as illustrated in Fig. 4.2b.

4.2.2 Composition of DMPs for Cluttered Environments

Although CMA-ES is able, to some extent, to address the constraints imposed by the environment to the DMP trajectories, in very cluttered environments it may not be enough. Figure 4.4 illustrates several cases in which CMA-ES was unable to attain the

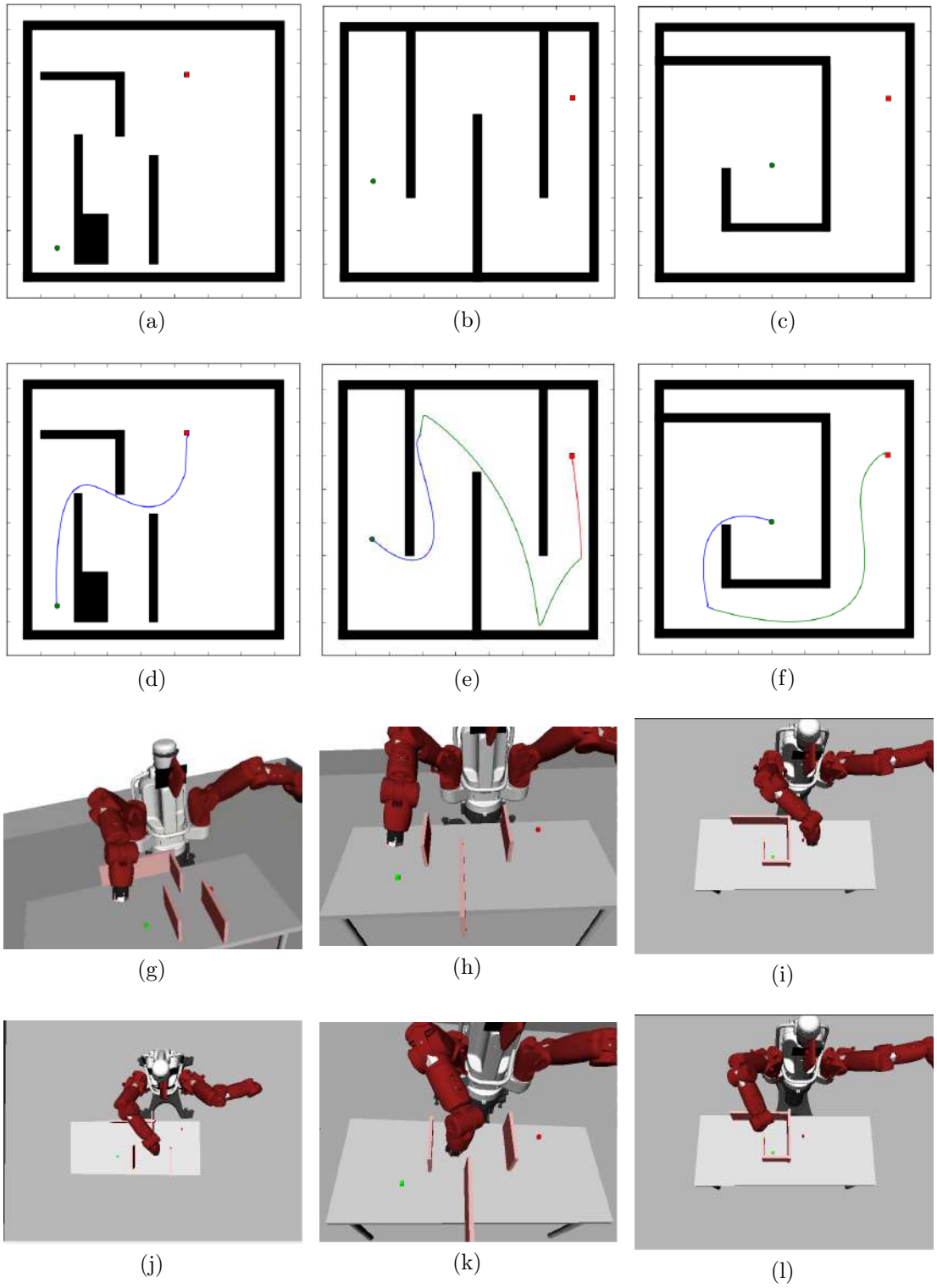
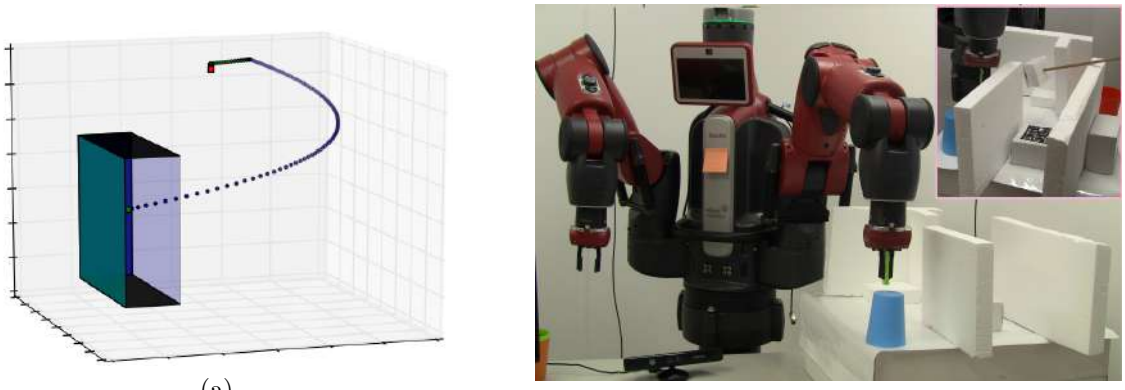


Figure 4.5: Successful examples where combined DMPs reached the target position. Result is depicted in geometric simulation and using the Baxter simulator. (a)-(c) Geometric simulation environment layout; (d)-(f) Execution in geometric simulation environment; (g)-(i) Baxter simulation environment layout; (j)-(l) Execution in Baxter simulation environment. The DMP library contains five Dmps.



(a) (b)

Figure 4.6: 3D environment: Successful examples where combined DMPs reached the target position. (a) Result in geometric simulation. The green circle and red square correspond to the start and end points, respectively. (b) Setup for the real-world experiment using the Baxter robot.

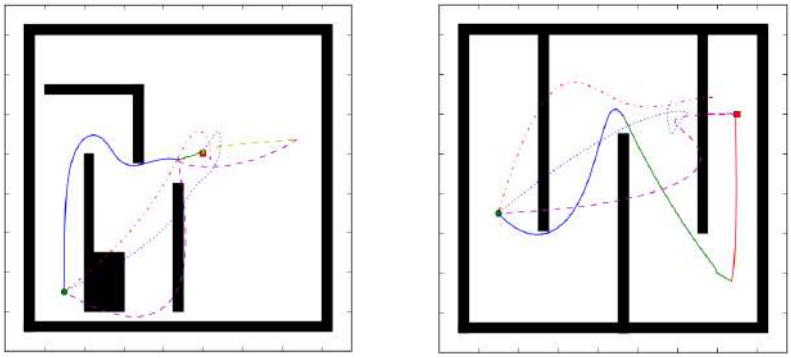
target position due to the constraints imposed by the environment.

In contrast, our approach is able to successfully generate trajectories that reach the target while avoiding the obstacles, as illustrated in Figs. 4.5 and 4.6a.

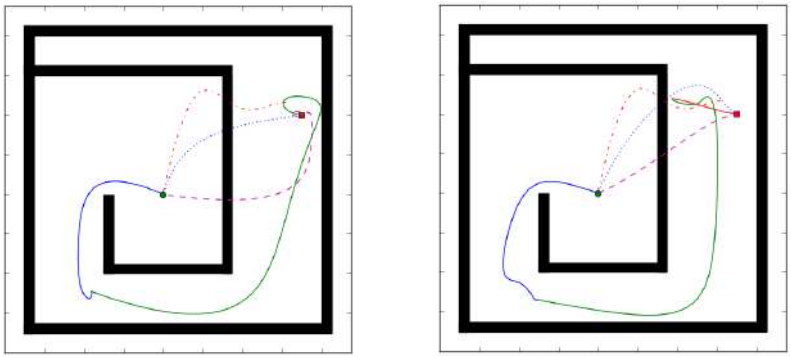
Figure 4.7 illustrates the impact of each individual component of the framework on the performance as a whole. Our framework has succeeded in all environments and conditions, while the alternatives (that do not consider at least one of the components in the framework) either could not reach the target or could not avoid the obstacles and get a safe path to the target. In the experiment, the DMP library contains four DMPs.

We note that the combined trajectory can be further optimized (again with CMA-ES) to obtain a yet smoother trajectory. Figure 4.8 showcases the trajectory resulting from applying CMA-ES to the combined trajectory.

Finally, we illustrate in Fig. 4.9 the dependence of our approach on the optimization module. In particular, we depict the results obtained by using a different number of iterations and step-size during the combination process. As evidenced in the plots, running CMA-ES for a larger number of generations allows for "straighter" trajectories; similarly, an adequately selected step-size provides good exploration for the environment (Fig. 4.9).



(a) (b)



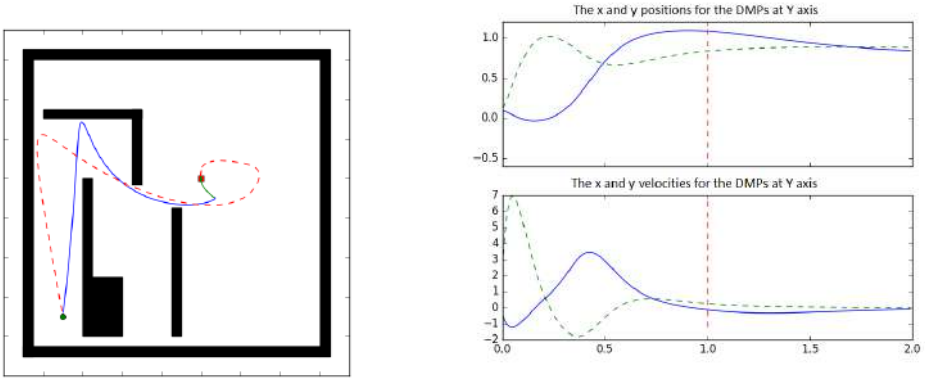
(c) (d)

Figure 4.7: Examples where our approach could successfully combine multiple DMPs to reach the target position in different environments and different conditions. The solid blue line corresponds to the result of our approach. The dashed line features no optimization (just combination); the red dashed-pointed line is a single DMP without optimization; the blue-pointed line is a single DMP with optimization. The green circle and red square correspond to the start and end points, respectively. (a), (b), (c) depict different environment configurations. (d) depicts the resulting trajectory in the environment (c) when optimization is stopped early. The DMP library contains five Dmps.

4.2.3 Execution-time Obstacle Avoidance

During the execution of a trajectory, it may happen that the robot needs to accommodate a moving obstacle. For example, during the interaction with a human user, the user may move and the robot should be careful to avoid hitting the user. In other words, the robot should treat the user as a "moving obstacle" and adjust its motion to the motion of the human.

As discussed in Section 4.1, our framework accommodates one such module, that addresses precisely the avoidance of obstacles at execution time. We illustrate in Fig. 4.10 a sequence of frames in which the trajectory executed by the robot is slowly adjusted to avoid the obstacle.



(a) (b)

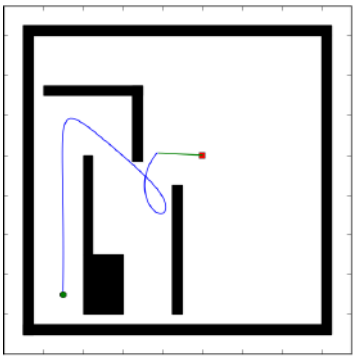
Figure 4.8: Application of CMA-ES to the combined trajectory. (a) Upon applying CMA-ES to the combined trajectory (depicted with a solid line), we obtain the significantly smoother dashed trajectory. The green circle and red square correspond to the start and end points, respectively. (b) Position and velocity profiles for the optimized combined trajectory. The dashed red line indicates the point at which the two original DMPs were put together.

Finally, we also applied our framework in the Baxter robot, as illustrated in Fig. 4.6b. In the experiment, the robot must transfer the blue object to the right side of the obstacle, having only been taught how to move its arm in the absence of obstacles. Once again, Baxter is able to successfully move the blue object without knocking down the obstacles (see also supplementary video).

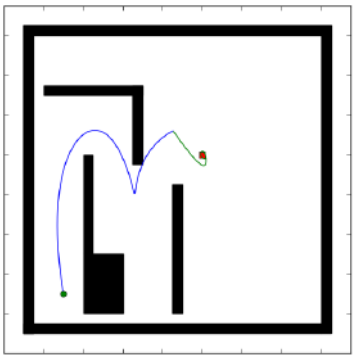
4.3 Concluding Remarks

In this chapter, we described a novel end-to-end framework to learning and generating complex motions from a library of "atomic" actions (DMPs) learned by decomposing a demonstration from a human expert. Our approach relies critically on the robot's ability to optimize a DMP's trajectory to reach a target while complying with the environment constraints. Our approach is also able to accommodate execution-time obstacle avoidance. Although our results involve relatively simple domains, they are, nevertheless, illustrative of the potential of our proposed approach.

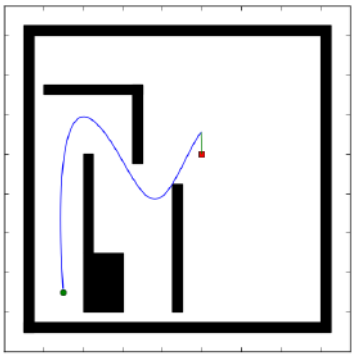
Our research also opens important research questions. Given the library of movements that the robot has available, can the robot *recognize* a movements from such library upon observing it? Going one step further, in light of the robot's ability to combine the different movements in the library, is the robot able to recognize a *sequence*



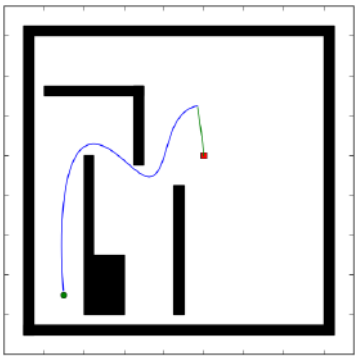
(a) 25 CMA-ES generations.



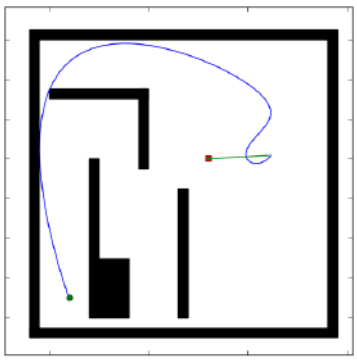
(b) 50 CMA-ES generations.



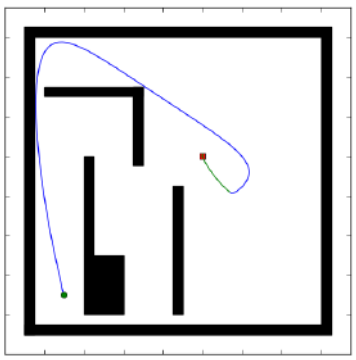
(c) 250 CMA-ES generations.



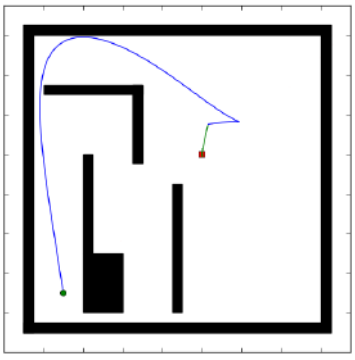
(d) 300 CMA-ES generations.



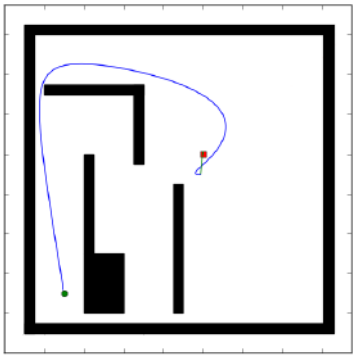
(e) 100 generations.



(f) 150 generations.



(g) 200 generations.



(h) 300 generations.

Figure 4.9: Dependence of the combined trajectory on the number of iterations for (a)-(d) smaller step-size; (e)-(h) larger step-size. The green circle and red square correspond to the start and end points, respectively. The DMP library contains five Dmps.

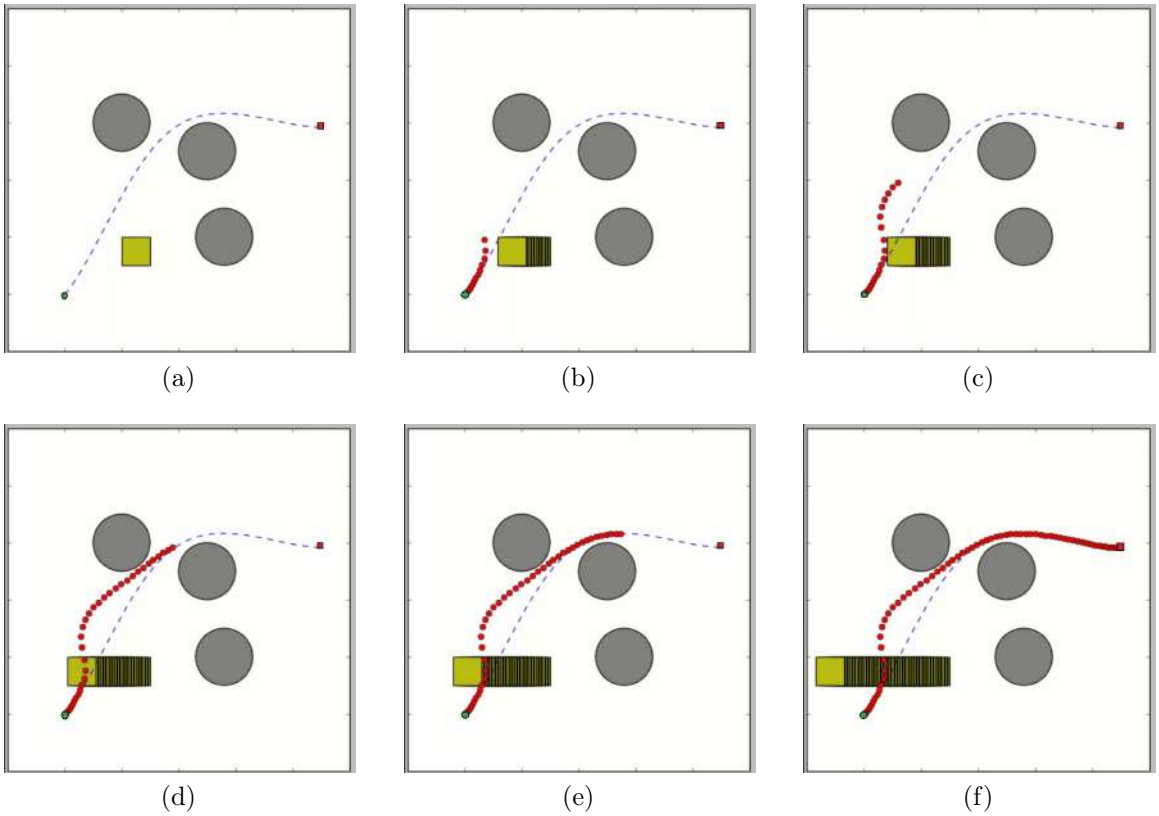


Figure 4.10: Sequence of frames showing execution-time obstacle avoidance. The dashed blue line corresponds to the optimized trajectory, and the green circle and red square correspond to the start and end points, respectively. The yellow square represents an obstacle that is moving leftwards, eventually colliding with the optimized trajectory. The sequence of red dots corresponds to the actual trajectory executed by the robot. Note that, as soon as the obstacle is out of the way, the robot resumes its initially planned trajectory.

of such movements upon observing it? In the next chapter, we address precisely these questions.

Chapter 5

Recognition and Prediction Using Dynamic Movement Primitives

This chapter describes an approach for (a) recognizing an observed trajectory from a library of pre-learned motions; (b) predicting the target position of such trajectory; and (c) recognizing compound movements. We use critical points from the observed trajectory to time-align it with those in the library. To match the observed trajectory with those in the library, we compare the changes in velocity orientation between consecutive critical points. The proposed approach is computationally light and, as such, can be performed at execution time. As for the prediction, we adopt a similar approach: after matching the observed trajectory to one in the library, we use the latter to predict the target point, modulating it to match the observed trajectory. We then discuss the application of these key ideas in the recognition of compound movements.

Both recognition and prediction approaches are probabilistic and, as such, provide a measure of certainty/uncertainty in the recognition/prediction process. Such a measure of uncertainty is important in tasks involving human-robot collaboration, as it allows the robot to decide when it is sufficiently certain to act, conditioned on the estimated trajectory. We illustrate our approach both in simulation and in a human-robot interaction scenario involving the Baxter robot.

Our proposed approach comprises two steps: in a first step, we rely on the notion of *critical points* [83] for time alignment and recognition. In particular, we compare the observed trajectory with those in the library at only those critical points, thus avoiding more computationally intensive time alignment procedures (such as DTW) or too stringent assumptions (such as linear time scaling between the observed trajectory and those in the library). We compute the probability of the observed trajectory given each movement in the library assuming a maximum entropy model, and use the probabilistic representation to assess the level of certainty of the system on the recognition. Target prediction is then computed from the expected target of the different motions in the library.

To recognize compound movements, we build on the previous recognition framework and assume that each segment in the observed trajectory is either the initial segment of a new primitive or the continuation of the previous segment. Then, given a partial trajectory, our method can predict the most likely next target—i.e., the end-point of the movement currently being executed, if the latter is executed to the end. Using an effective search tree, our approach can run at execution time and provide an efficient way

for action recognition and prediction, which has applications in interaction scenarios.

5.1 Recognition and Prediction Using Critical Points

We address the problem of observing a (partial) trajectory and recognizing it as being part of one of a set of movements in a library, each represented as a DMP. Once the DMP that best matches the observed trajectory is identified, we want to predict the end-point of the trajectory. We consider each of the two problems (recognition and prediction) separately.

5.1.1 Recognition

Ideally, our approach to recognition should be able to accommodate trajectories that differ from those in the library in the (linear and angular) velocity of execution, and changes in the time scale and target. We therefore adopt a representation that is invariant to those transformations.

As seen in Chapter 2, given a trajectory $y_{0:T} = \{y(t), t = 0, \ldots, T\}$, *critical points* are defined as minima of the function

$$h(t) = \|\dot{y}(t)\|^2 + \|\ddot{y}(t)\|^2.$$

In other words, the minima of h above correspond to "inflection" points of the trajectory. Such critical points were already used in Chapter 4 as suitable candidates for trajectory segmentation (also following the work of Meier et al. [83]).

In this chapter, we describe a segment between two critical points by its velocity and orientation changes. The orientation is defined in a 2D environment using a single angle, a velocity angle. In other words, given a trajectory $y_{0:T}$ with critical points $\tau_0, \ldots, \tau_{M+1}$, we compute

- The velocity, $v(\tau_m) = \dot{y}(\tau_m)$, at each of the critical points;
- The orientation, $\theta(\tau_m)$, at each of the critical points.

Segment m, between critical points τ_m and τ_{m+1}, is then described by the tuple

$$s^{(m)}(y_{0:T}) = \big(\delta_v^{(m)}(y_{0:T}), \delta_\theta^{(m)}(y_{0:T})\big), \qquad m = 0, \ldots, M,$$

where

$$\delta_v^{(m)}(y_{0:T}) = v(\tau_{m+1}) - v(\tau_m)$$

and

$$\delta_\theta^{(m)}(y_{0:T}) = \theta(\tau_{m+1}) - \theta(\tau_m).$$

By considering a trajectory only at critical points, we overcome the need for explicit time alignment. Additionally, to render the representation invariant to rotations, scaling and target, we define the normalized segment representation

$$\bar{s}^{(m)}(y_{0:T}) = \left(\bar{\delta}_v^{(m)}(y_{0:T}), \bar{\delta}_\theta^{(m)}(y_{0:T})\right), \qquad m = 1, \ldots, M.$$

with

$$\bar{\delta}_v^{(m)}(y_{0:T}) = \frac{\delta_v^{(m)}(y_{0:T})}{\delta_v^{(0)}(y_{0:T})},$$
$$\bar{\delta}_\theta^{(m)}(y_{0:T}) = \frac{\delta_\theta^{(m)}(y_{0:T})}{\delta_\theta^{(0)}(y_{0:T})}.$$

A trajectory $y_{0:T} = \{y(t), t = 0, \ldots, T\}$, with critical points $\tau_0, \ldots, \tau_{M+1}$ is thus summarized as a sequence

$$S(y_{0:T}) = \{\bar{s}^{(m)}(y_{0:T}), m = 1, \ldots, M\}.$$

Now given a library of N trajectories represented as DMPs, $\{y_n, n = 1, \ldots, N\}$, let

$$S(y_n) = \{\bar{s}^{(m)}(y_n), m = 1, \ldots, M_n\}$$

denote the summarized representation of y_n. Given an arbitrary (partial) trajectory $y_{0:t}$, described by the summarized representation

$$S(y_{0:t}) = \{\bar{s}^{(m)}(y_{0:t}), m = 1, \ldots, M_*\},$$

we compute the error between segment m of $y_{0:t}$ and y_n as

$$\epsilon_n^{(m)} = \left\|\bar{\delta}_v^{(m)}(y_n) - \bar{\delta}_v^{(m)}(y_{0:t})\right\| + \left\|\bar{\delta}_\theta^{(m)}(y_n) - \bar{\delta}_\theta^{(m)}(y_{0:t})\right\|.$$

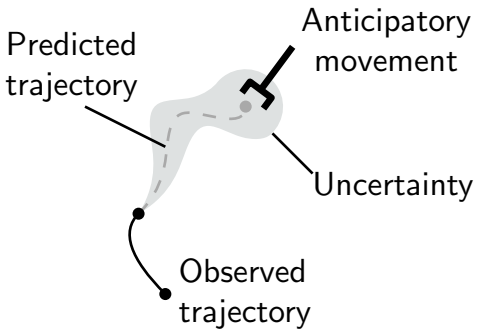


Figure 5.1: Prediction and anticipatory movement. The robot observes part of a trajectory and, using the information in its library, predicts the full trajectory and its target. Once the uncertainty regarding such prediction goes below some threshold, the robot can then move in an anticipatory fashion the predicted target.

Finally, we define the recognition error of DMP y_n to be

$$E_n = \sum_{m=1}^{M_*} \epsilon_n^{(m)}. \tag{5.1}$$

We note that, if $M_n < M_*$, then we define $\epsilon_n^{(m)}, m > M_n$, to take a constant large value.

Finally, we associate with each DMP y_n a probability

$$\omega_n = \Pr(y_n \mid y_{0:t}) = \frac{\exp(-E_n)}{\sum_{\ell=1}^{N} \exp(-E_\ell)}, \tag{5.2}$$

indicating the probability that it was DMP y_n to yield the partial trajectory $y_{0:t}$.

5.1.2 Prediction

The purpose of prediction is to infer the target position of an observed movement. Prediction is useful, for example, in the context of human-robot interaction, as it allows the robot to adjust its movement to that of the human user by predicting the target and timing of the later (see Fig. 5.1). Such anticipatory behavior will foster successful collaboration between robot and human, closer to that observed between humans.

We discuss two distinct approaches to prediction. The first is based on the critical point information already used for recognition (Section 5.1.2), while the second relies on the formalism of dynamic motion primitives (DMP) to estimate the target of the motion (Section 5.1.2).

◇

Prediction occurs at run-time, as we make successive observations of the trajectory, and uses the output of the recognition approach described in Section 5.1.1. Let $y_{0:t}$ denote a partial trajectory, observed up to time step t, and let $w_n(t)$ denote the probability

$$\omega_n(t) = \Pr(y_n \mid y_{0:t}),$$

computed as described in Section 5.1.1. In other words, as we observe larger portions of the trajectory $y_{0:t_1}, y_{0:t_2}, \ldots$, we compute a sequence of probabilities $\{\omega(t_1), \omega(t_2), \ldots\}$, where each $\omega_n(t)$ is computed from $y_{0:t}$ and we write $\omega(t)$ to denote the vector with nth entry given by $\omega_n(t)$.

The predicted trajectory at each time step t, $\hat{y}_{0:T}$, can now be computed as a weighted combination of the trajectories predicted from the movements in the library as

$$\hat{y}_{0:T} = \sum_{n=1}^{N} \omega_n(t)\hat{y}_n, \tag{5.3}$$

where $\hat{y}_n$ is the prediction from the nth trajectory in the library. The predicted target of the trajectory corresponds to $\hat{y}(T)$. The uncertainty in the prediction can also be computed by computing the (co)variance of the prediction at each point $\hat{y}(t)$ of $\hat{y}_{0:T}$ as

$$\Sigma(t) = \sum_{n=1}^{N} \omega_n(t)(\hat{y}(t) - \hat{y}_n(t))^T(\hat{y}(t) - \hat{y}_n(t)).$$

It is worth noting that the longer we observe the trajectory $y_{0:t}$, the more reliably our approach will be able to estimate the probabilities $\omega_n(t), n = 1, \ldots, N$, and the more accurate the prediction in (5.3) will be.

In the continuation, we discuss two ways of computing the predictions $\hat{y}_n$ according to the movements in the library.

Prediction using Position Ratios

In this method, we again use critical point information to compute the predicted trajectory. In particular, given the partial trajectory $y_{0:t}$, let τ_m denote the last observed critical point. Let y_n denote a movement in the library, and define

$$\bar{\delta}_y(y_n) = \frac{y_n(\tau_{M+1}) - y_n(\tau_m)}{y_n(\tau_m) - y_n(\tau_0)}. \tag{5.4}$$

We rely on the simple intuition that, if the movement $y_{0:t}$ that we have observed is, indeed, a partial execution of y_n, then the ratio in (5.4) remains constant and we have that

$$\frac{y(\tau_{M+1}) - y(\tau_m)}{y(\tau_m) - y(\tau_0)} = \bar{\delta}_y(y_n).$$

Therefore, we can compute the prediction

$$\hat{y}_n(\tau_{M+1}) = y(\tau_m) + \frac{y(\tau_m) - y(\tau_0)}{y_n(\tau_m) - y_n(\tau_0)} \cdot (y_n(\tau_{M+1}) - y_n(\tau_m))$$

Such predictions are then combined using (5.3).

Target Prediction Using the DMP Equations

In this method, we rely on the representation of trajectories as dynamic motion primitives (DMPs) and use the DMP parameter information (from both the observed trajectory and the DMPs in the library) to build our prediction. From the observed trajectory we can compute the parameters of DMP by using information regarding the starting point, target point (which is unknown), the velocity or the acceleration. The parameters from the DMP are the weights, the canonical system, the basis functions and the constants α and β [39].

Let y_n denote a trajectory/DMP in the library. We need to select one point from the observed trajectory and apply the DMP equation in this state. We select the last observed critical point, τ_m, to recover the canonical system variable (x) as

$$x = \exp\left(\alpha_x \frac{m}{M}\tau\right),$$

Where α_x is a constant of the canonical system, M is the number of critical points, τ is a time-scaling factor, m is the selected critical point [6].

We must now compute the weights of the basis functions for the DMP y_n, $w_{i,n}$, and the value of those basis functions at the computed canonical state, $\psi_i(x)$. The dynamic equations of the DMP at the selected critical point become [39]

$$\ddot{y}_n = \alpha_y(\beta_y(g - y_n) - \dot{y}_n) + \frac{\sum_i \psi_i(x) w_{i,n}}{\sum_i w_{i,n}} x(g - y_n(\tau_0)),$$

where α_y, β_y are constants and g represents the target. Letting

$$A = \frac{\sum_i \psi_i(x) w_{i,n}}{\sum_i w_{i,n}},$$

we can isolate the target (g) to get our prediction

$$g = \hat{y}_n(\tau_{M+1}) = \frac{\ddot{y} + \alpha_y \dot{y} + \alpha_y \beta_y y + Axy(\tau_0)}{Ax + \alpha_y \beta_y} \tag{5.5}$$

Applying the approach above to all the DMPs in the library, we get a set of predictions $\hat{y}_n(\tau_{M+1}), n = 1, \ldots, N$, which again can be combined using (5.3).

5.1.3 The Threshold

In real life, humans do not need to wait for a partner to finish its movement to take the decision and start their own action, or watch all of the companion's trajectory to start the next required task. Likewise, we should have our robot act when there is sufficient data to know with some confidence what the movement of the user will be. In our approach, we allow for some margin of error and define a threshold (ϵ) such that, if the level of uncertainty in the observed trajectory's target is below the threshold, the robot will move. The threshold driving such decision translates the notion of "acceptable error", in terms of the robot's ability to recognize the movement/DMP generating the observed trajectory.

To verify the level of (un)certainty in the recognition we compute the entropy $H(t)$ of the distribution $\{\omega_1(t), \ldots, \omega_N(t)\}$ after each critical point,

$$H(t) = -\sum_n \omega_n(t) \log \omega_n(t).$$

When the entropy falls below some pre-defined threshold ϵ, the robot moves towards the predicted target. Using a weighted combination for predicting the target makes our method robust to small differences, poor execution, or noise. Moreover, if the observed trajectory was not from the library, our method can still leverage all the DMPs in the library to somehow predict the target, which provides our method reliability and stability.

5.1.4 Extending Recognition for 3D Trajectories

The approach described so far was designed with 2D trajectories in sight. However, the fundamental ideas can be extended to 3D settings as follows.

A segment m in a trajectory $y_{0:T}$ (i.e., the part of the trajectory between critical points τ_m and τ_{m+1}) is described by the normalized segment representations $\bar{s}^{(m)}(y_{0:T}$, and contains information regarding:

- The change in linear velocity in the current segment relative to the change in linear velocity in the initial segment, $\bar{\delta}_v^{(m)}$;
- The change in orientation of taking place in the current segment with respect to the change in orientation in the initial segment, $\bar{\delta}_\theta^{(m)}$.

When considering a 3D trajectory, the definition of $\bar{\delta}_v^{(m)}$ remains fundamentally unchanged: it measures the change in linear velocity in the current segment relative to the change in linear velocity in the initial segment.

Extending the $\bar{\delta}_\theta^{(m)}$ to 3D requires a little more care. When considering 2D trajectories, the change in orientation $\delta_\theta^{(m)} = \theta(\tau_{m+1}) - \theta(\tau_m)$, used to compute $\bar{\delta}_\theta^{(m)}$, can seen as a rotation of an angle $\delta_\theta^{(m)}$ about an axis orthogonal to the movement plane. When considering 3D movements, a change in orientation can still be interpreted as a rotation of an angle $\delta_v^{(m)}$, but now about some axis $u^{(m)}$.

We can now extend our recognition approach to 3D by now considering that a trajectory y with critical points $\{\tau_1, \ldots, \tau_M\}$ is summarized as a sequence

$$S(y) = \{\bar{s}^{(m)}(y), m = 1, \ldots, M\},$$

with

$$\bar{s}^{(m)}(y) = (\bar{\delta}_v^{(m)}(y), \bar{\delta}_\theta^{(m)}(y), u^{(m)}(y)),$$

and defining the error between segment m of two trajectories y and y' as

$$\epsilon^{(m)} = \left\|\bar{\delta}_v^{(m)}(y) - \bar{\delta}_v^{(m)}(y')\right\| + \left\|\bar{\delta}_\theta^{(m)}(y) - \bar{\delta}_\theta^{(m)}(y')\right\| + \left\|u^{(m)}(y) - u^{(m)}(y')\right\|.$$

Using the above definitions, it is now possible to extend the ideas in Sections 5.1.1 and 5.1.2 to 3D trajectories.

5.2 Compound Movement Recognition Using DMPs

We now build on the recognition framework of Section 5.1 and extend it to handle trajectories comprising multiple segments from possibly different movements. We assume that each segment in the observed trajectory is either the initial segment of a new primitive or the continuation of the previous segment. Then, given a partial trajectory, our method can predict the most likely next target—i.e., the end-point of the movement currently being executed, if the latter is executed to the end. Using an effective search tree, our approach can run at execution time and provide an efficient way for action recognition and prediction, which has applications in human-robot interaction scenarios. We validate our approach both in simulation and in a human-robot interaction scenario involving the Baxter robot.

5.2.1 Recognition and Prediction of Compound Movements

We address the problem of movement recognition, when the observed movement is possibly a compound trajectory comprising segments from multiple simpler movements. We henceforth refer to these simpler movements as *primitive movements*. The purpose of our approach is to recognize the individual components forming the observed movement, out of a library of pre-learned (primitive) movements. We assume that each segment in the trajectory is either (i) the initial segment of a possibly different primitive movement; or (ii) the continuation of the previous primitive movement. We consider a segment to be the part of a trajectory lying between two critical points.

Besides recognition, we are also interested in predicting the endpoint of the observed trajectory. Unfortunately, since the trajectory can be formed by an arbitrary number of segments coming from different primitive movements, it is not possible to predict, beforehand, the endpoint of the movement—unless if we know exactly how many and which segments compose the trajectory. As such, we instead predict *the most likely end-point for the current primitive movement*, assuming that it is performed all the way to the end.

Our approach is extending our work to consider partial trajectory recognition and prediction. Much like in that work, our method relies on the identification of the critical points extracted from the observed trajectory, comparing them with the corresponding

critical points in the trajectories in the library.

Motion Recognition

We can interpret our previous approach as deploying a "sequence search": at each critical point, the current "sequence of segments" is compared with each movement in the library, and recognition thus consists of identifying the movement in the library that best matches the observed "sequence".

In our proposed approach, we instead build a *search tree*. The depth of the tree corresponds to the number of observed segments, and each node corresponds to a primitive movement in the library. The search tree is built as follows:

- Each node of the tree has as many children as there are primitive movements in the library plus one.
- The children corresponding to the primitive movements in the library represent the situation where the current segment is the initial segment of the respective primitive movements.
- The additional child represents the situation where the current segment is a continuation of the current primitive movement.
- Each node in the tree is associated with a segment error, as defined in (5.1).

Figure 5.2 shows an example of a possible search tree for the case where the movement library contains only 3 primitive movements. By using the search tree, we can now determine the sequence of primitives that minimizes the total cost. After each new critical point is discovered, the tree grows a level in depth and the node values are computed.

Note that our approach considers the several possible cases: that the current segment corresponds to the beginning of a new primitive movement; or that the current segment is the continuation of the previous primitive movement. Additionally, to discourage the search process from excessively switching between movement primitives, we add a small cost of c to all the nodes corresponding to changes in the movement primitives.

To render searching the tree a more efficient process, in our approach we do not grow the complete tree simultaneously. Instead, we expand only the nodes corresponding

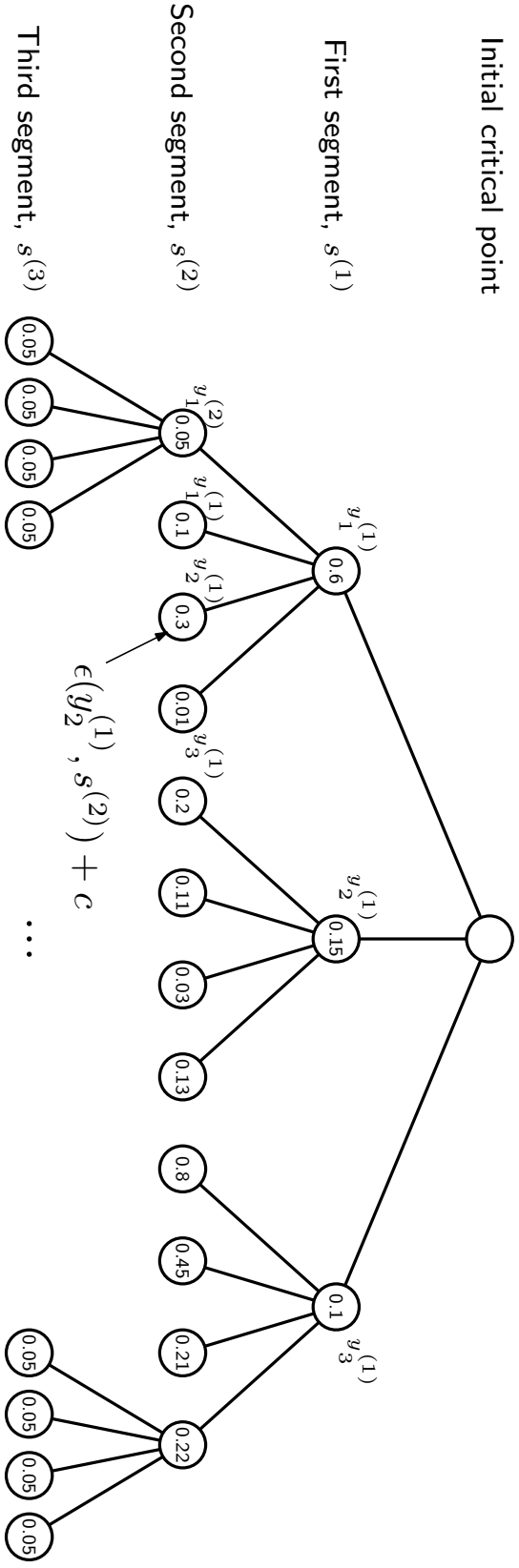


Figure 5.2: Example of a search tree. Each node is associated with a segment of a primitive movement in the library, and contains the error associated when comparing the current segment with the primitive movement segment corresponding to that node (the numbers in the nodes).

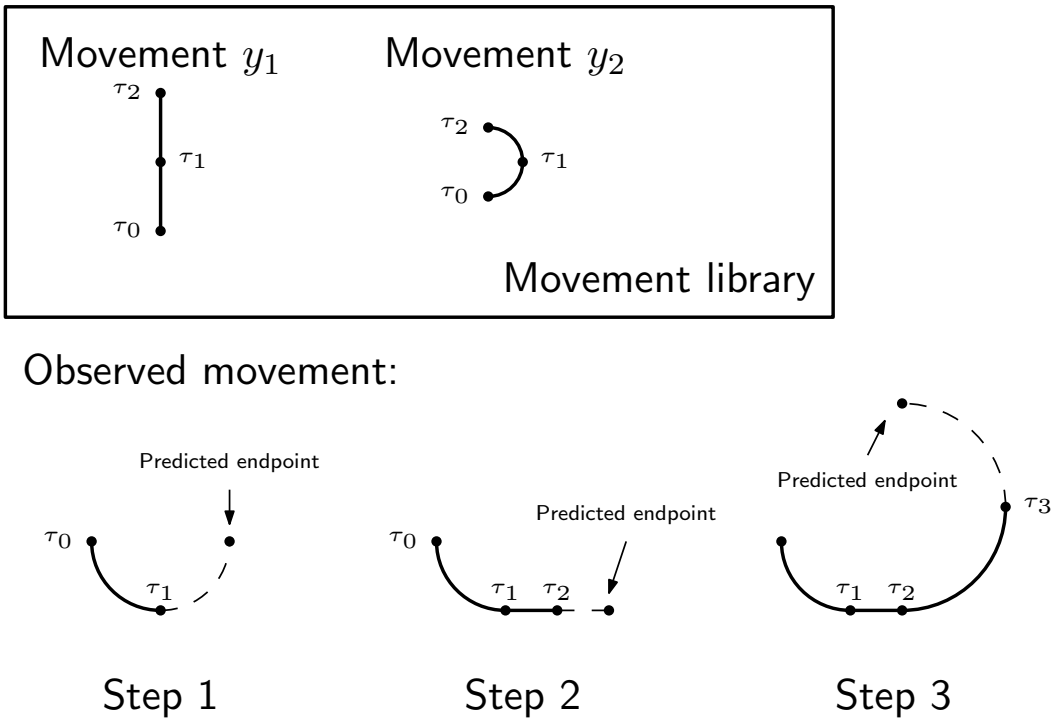


Figure 5.3: Example of what the prediction process looks like. After observing a segment and recognizing to which primitive movement such segment "belongs", we predict the endpoint as the endpoint of the corresponding primitive movement.

to the path with the smallest average cost per node. While this heuristic for growing the tree still leads to the expansion of some unnecessary nodes, it is significantly more efficient than performing a full breadth-first search.

Motion Prediction

As previously discussed, the observed trajectory can be formed by an arbitrary number of segments coming from different primitive movements. As such, it is not possible to predict, beforehand, the endpoint of the complete movement. In our approach, we predict *the most likely end-point for the current primitive movement*, assuming that it is performed all the way to the end.

The prediction process uses the basic approach described in Section 5.1.2. At each critical point, after associating the most recent segment with a segment from one of the primitive movements in the library, we compute the endpoint of such primitive movement as the predicted endpoint of the trajectory, using (5.5). The process is illustrated in Fig. 5.3.

5.3 Experiments and Results

In this section, we test our proposed approach both in simulation and with the Baxter robot. Specifically, we test our proposed approach in two scenarios. The first is when our method tries to recognize a simple trajectory, and the second is when our method tries to recognize a compound trajectory.

5.3.1 Experiments with Simple Trajectories

We start by testing our proposed framework for recognition of simple trajectories. We conducted three different experiments, each of which is intended to show the effectiveness, accuracy, and features of the proposed approach.

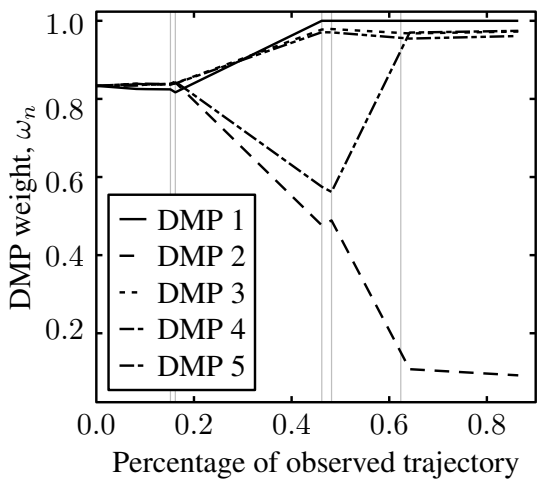
The first experiment illustrates the recognition process during the discovery of new critical points and the accuracy of the prediction of the target point. The second experiment illustrates the scalability of our approach when the library contains a large number of primitive movements. Finally, the fourth experiment showcases the application of our approach in a real world scenario featuring the Baxter robot.

In the first two experiments, we use a 2D geometric simulation environment. The library contains trajectories stored in the form of DMPs. A trajectory is gradually observed, and our approach performs recognition and prediction in runtime.

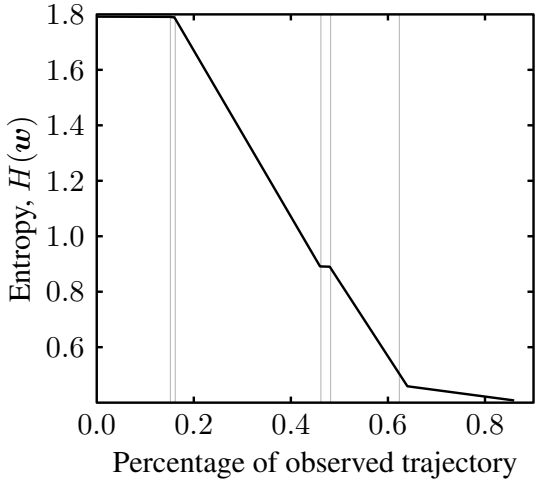
Recognition

As we observe successive critical points in the trajectory, the recognition module has more trajectory information and the movement in the library identified as most likely to produce the observed trajectory can change. In other words, as more critical points are observed in the trajectory, some of the movements in the library are deemed more likely to have produced such trajectory while others are deemed less likely.

Figure 5.4a shows the recognition results for a library of 5 DMPs in two dimensions, as we observe a larger percentage of the trajectory. We depict the evolution of the weights $\omega_n, n = 1, \ldots, 5$, as we observe successive critical points. We can observe when a new critical point is observed, as the plot abruptly changes at those points—critical points are most informative to identify the movement producing the observed trajectory. In the example, the library includes 4 movements similar to the one being observed



(a) Recognition weights ω_n for all DMPs at successive critical points (vertical gray lines).



(b) Entropy of the prediction with the observation of new critical points.

Figure 5.4: Weight and entropy evolution during the recognition process, as larger portions of the trajectory are observed.

and one very different movement. The weights associated with the similar movements approach 1, while the weight associated with the different movement steadily decreases.

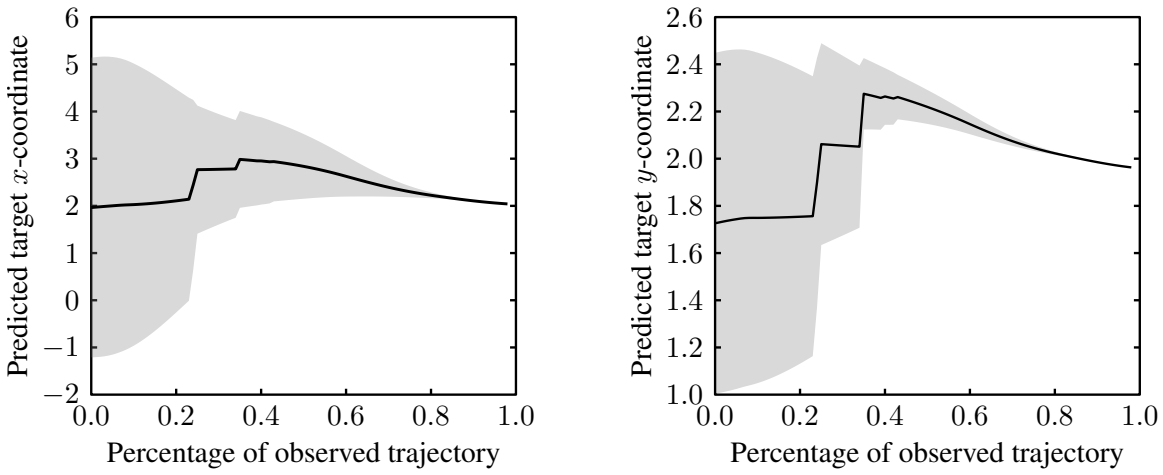
In Fig. 5.4b we can also observe a steady decrease in the entropy after observing approximately 20% of the trajectory. However, the fact that several DMPs are similar to the target movement implies that the entropy does not go to zero.

Prediction

As the trajectory to be predicted unfolds and new critical points are observed, the recognition module is able to provide more accurate identification of the library movement that best explains the observed trajectory. The identified movement can then be used to predict the target of the observed trajectory, following the approaches discussed in Section 5.1.2.

Figure 5.5 showcases how the prediction evolves as a larger portion of the trajectory is observed. We depict the uncertainty in each of the two coordinates (x and y): The solid line corresponds to the evolution of the predicted target, $\hat{y}(\tau_{M+1})$; the shaded area depicts the variance in the prediction.

We can observe the improvement in prediction, as the variance of the predicted targets decreases as additional critical points are observed. From this figure, we can observe sharp changes at the critical points, since the available information for recognition changes significantly at these points. This observation supports our claim



(a) Prediction of the target x coordinate.

(b) Prediction of the target y coordinate.

Figure 5.5: Uncertainty in the prediction of the target for the two dimensions of the trajectory. The solid line corresponds to the mean predicted target, $\hat{y}(\tau_{M+1})$, while the shaded area corresponds to the variance in the prediction.

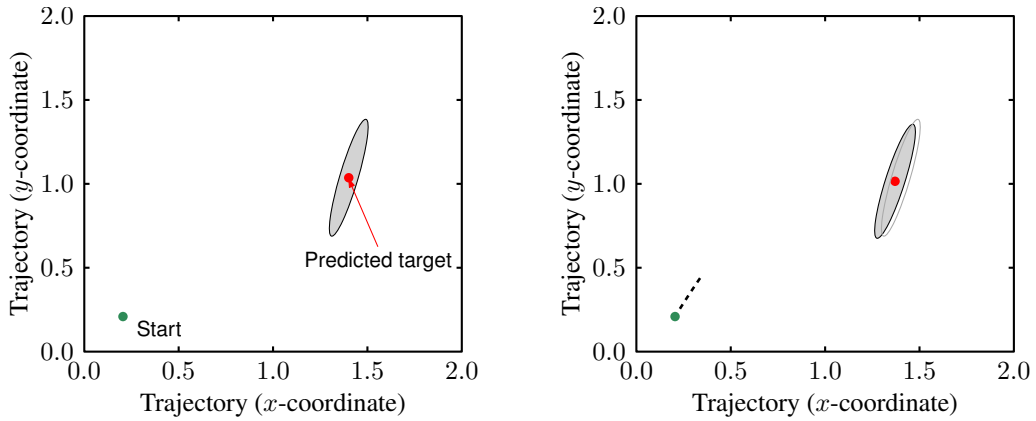
that critical points indeed provide useful information for recognition. We can also observe how the uncertainty in the prediction decreases significantly, as more of the trajectory is observed.

Figure 5.6 shows the evolution of the prediction as new critical points are observed. We can see how the distribution area changes and gets smaller as we increase the observed part of the trajectory, until finally arriving at the target point (the mean of this distribution).

Uncertainty and Anticipatory Movement

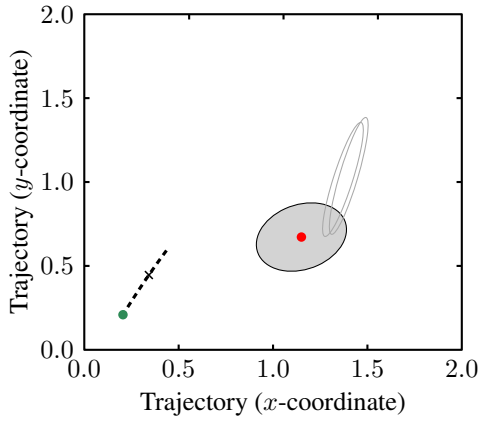
While observing the trajectory, a key question is: when can the robot take the decision and start its response? As discussed before, the robot will make the decision when the possibility of failure is small, or when there is little variation between the possible predictions considered by the robot. In terms of our approach, this translates into the situations where the distribution of the predicted targets has small variance. By computing the entropy of the recognition weights, we can have an estimate of the uncertainty in our predictions. As discussed in Section 5.1.2, we adopt a threshold ϵ and allow the robot to act in an anticipatory manner if the entropy is below such specified threshold. In our experiments, the threshold value is constant and was selected experimentally [3].

Figure 5.4b depicts how the entropy in the prediction evolves as we observe larger

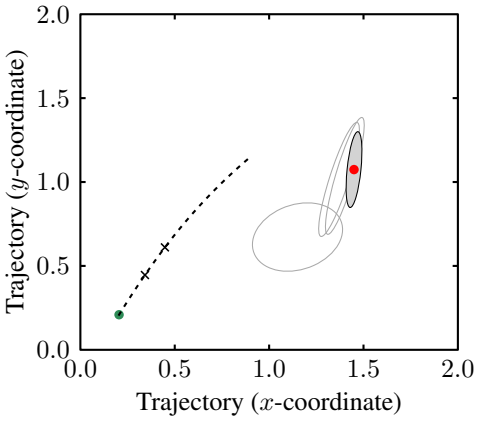


(a) Prediction point τ_0.

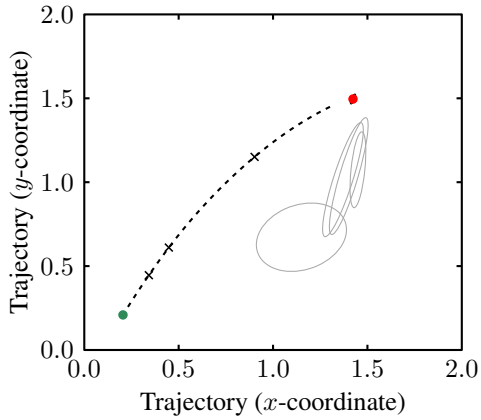
(b) Prediction point τ_1.



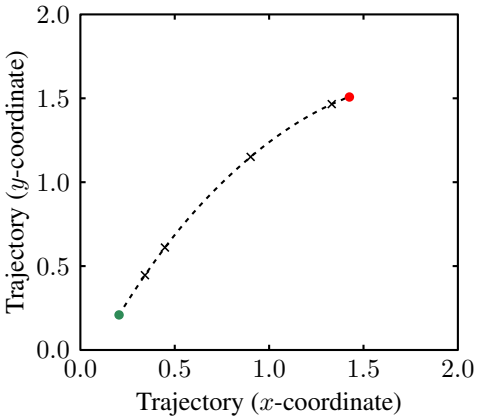
(c) Prediction point τ_2.



(d) Prediction point τ_3.



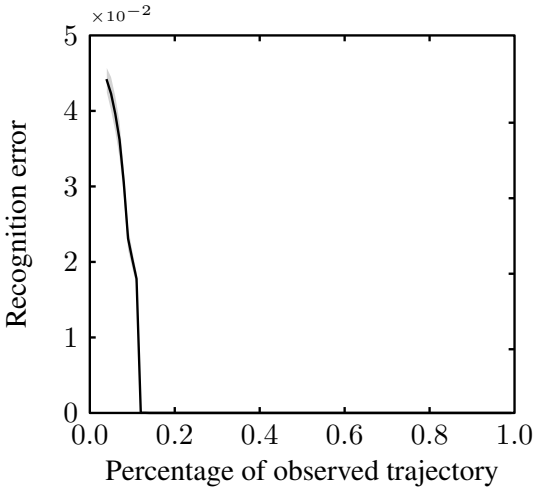
(e) Prediction point τ_4.



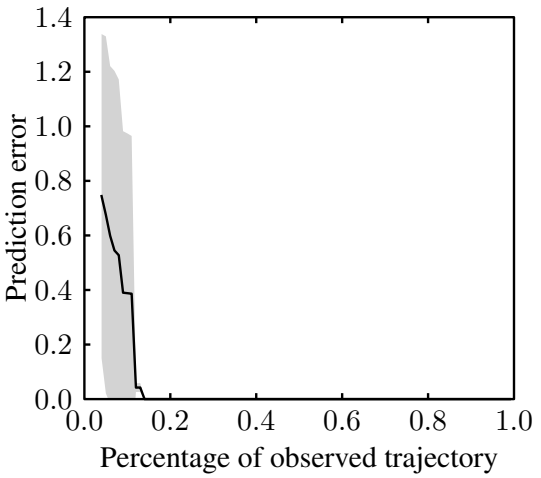
(f) Final trajectory.

Figure 5.6: Illustrative example of the prediction process, as larger portions of the trajectory are observed by the predictor. The red dot corresponds to the mean prediction and the ellipsoid to the corresponding variance.

portions of the trajectory. The point where the robot can initiate its anticipatory action will be when the entropy reaches a small value.



(a) Recognition error, given by $|\omega_{\max} - \omega_*|$, with $\omega_{\max}$ is the weight associated with the most likely DMP, and ω_* is the weight for the *correct* DMP.



(b) Prediction accuracy, given by $\|\hat{y}(\tau_{M+1}) - y(\tau_{M+1})\|$, where $\hat{y}(\tau_{M+1})$ is the predicted end-point and $y(\tau_{M+1})$ is the actual end point.

Figure 5.7: The errors in recognition and prediction for the 25 observed trajectories, as a function of the observed trajectory. The solid lines depict the mean error values, while the shaded area depicts the variance in these errors.

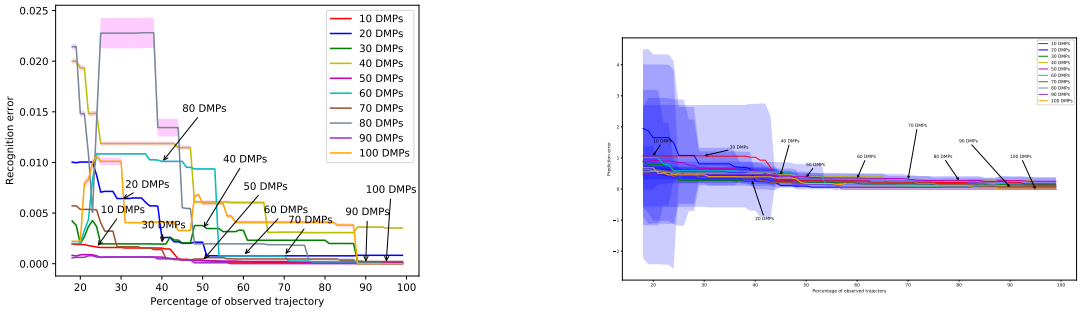
Accuracy

To illustrate the robustness of our framework, we run a statistical study to check the recognition and prediction and know the accuracy of our approach evolves as larger portions of a trajectory are observed.

The tests in the previous experiments were obtained with a single observed trajectory while fixing the number of DMPs in the library to 5 DMPs. We now consider a more extensive library comprising 85 distinct DMPs. We then performed 25 different trajectories from this library, each generated from one DMP in the library but modulated to different targets and execution speeds. We compute the error in recognition and prediction as the different trajectories unfold and report the mean error and error variance as a function of the percentage of trajectory observed.

Figure 5.7 depicts the results of this study. The solid lines represent the mean error, while the shaded areas correspond to the variance in the observed prediction/recognition error.

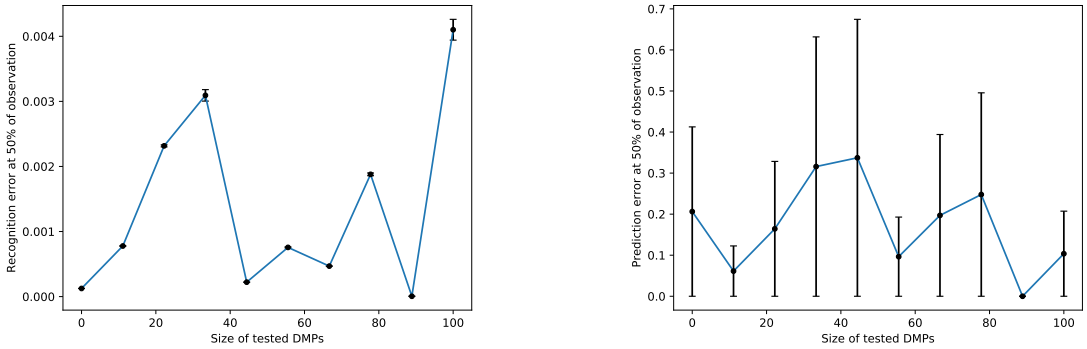
As can be observed from the plot, both the recognition and prediction error quickly converge to zero, as does the variance in the prediction. This establishes the ability of our approach to identify and predict the target of an observed trajectory.



(a) Recognition accuracy.

(b) Prediction accuracy.

Figure 5.8: The errors in recognition and prediction for 10, 20, 30, 40, until 100 observed trajectories within a 100 DMP library. The solid lines depict the mean error values, while the shaded area depicts the variance in these errors.



(a) Recognition error.

(b) Prediction error.

Figure 5.9: The errors in recognition and prediction at a specific point from 10, 20, 30, 40, until 100 observed trajectories within a 100 DMP library.

To further assess the performance of our recognition approach, we investigate the dependence of the performance on the size of the library. Figure 5.8 illustrates the recognition and prediction errors observed for different sizes of the DMP library. Each line corresponds to the average error over 10 DMPs selected randomly from the library, as the size of the library from 10 to 100 DMPs. Although larger libraries tend to lead to larger initial recognition errors, we can observe that in all cases the recognition error quickly decreases. the plot clearly shows, Our method can still recognize the trajectories and predict their target points within different sizes of libraries.

Figure 5.9 offers a different view of the same results, showcasing the average recognition and prediction errors after observing 50% of 10 DMPs selected at random from the library—this time as a function of the size of the library. We cannot observe a clear trend, suggesting that our approach scales well to libraries of different sizes.

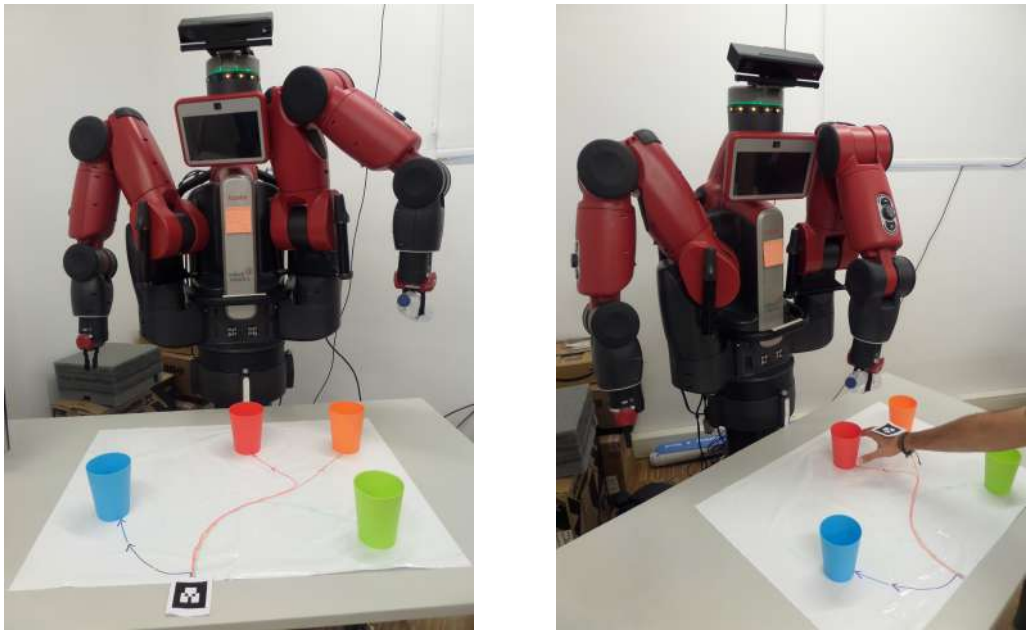


Figure 5.10: Illustration of the human-interaction scenario considered in this chapter, where BAXTER moves in anticipation to the user to fill the cup that the user is aiming for.

Implementation in the BAXTER Robot

We now describe the application of our proposed approach in a human-robot interaction scenario involving a BAXTER robot. In this task, the human user approaches one of the cups in the table (Fig. 5.10a). Upon realizing which cup the human is reaching, BAXTER performs an anticipatory move towards such cup, "pouring" water in that cup (Fig. 5.10b). We use the recognition and prediction approaches described in Sections 5.1.1 and 5.1.2.

Figure 5.11 depicts a sequence of frames that illustrate the our approach at work in the actual robot. The DMPs in the robot library correspond to the motions represented in the table in different colors. While the initial motion of the user could head to any of the red, orange or green cups, the robot remains still, as its uncertainty regarding the target is above the specified threshold. However, as soon as the user's hand moves past the common segment of the three movements, the robot immediately identifies the movement as belonging to the DMP that leads to the orange cup, and initiates the corresponding anticipatory motion, succeeding in "serving" the orange cup before the user actually reaches it.

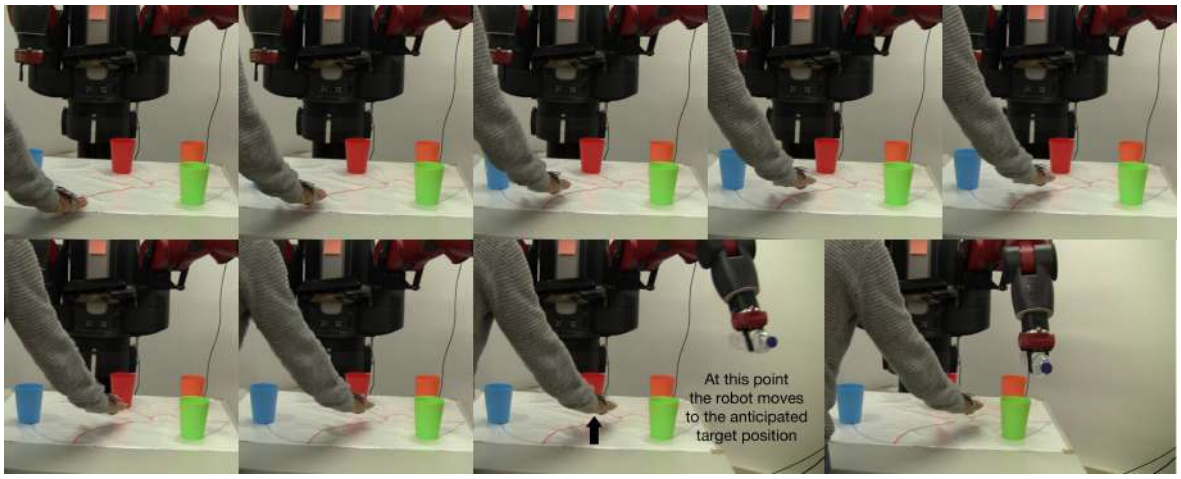


Figure 5.11: Successive frames depicting the movement of the user towards the orange cup. When it becomes clear to BAXTER that the target is the orange cup, the robot initiates its own anticipatory movement. At the point indicated by the black arrow, the robot moves to the anticipated target position.

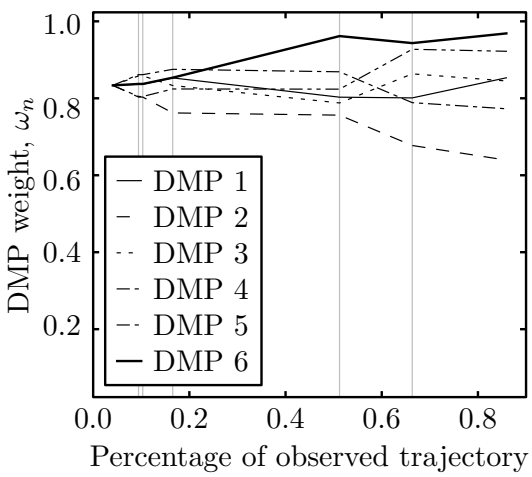
5.4 Experiments with Compound Trajectories

We now describe the experiments conducted to test our proposed framework for recognition of compound trajectories. We now conducted four experiments that illustrate the main features of our proposed approach for compound movement recognition.

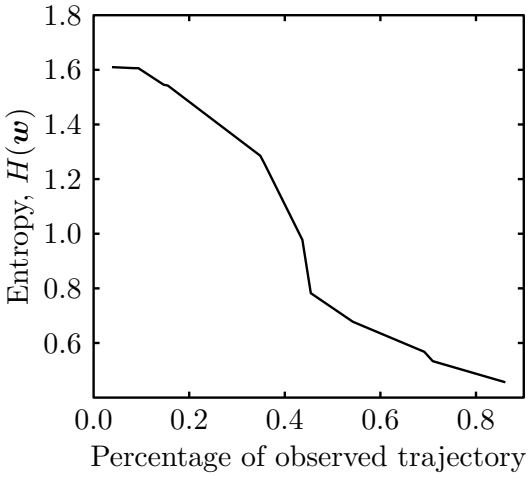
As in the previous section, our first experiment features the recognition process; the second experiment illustrates the ability of our method to recognize the components forming the observed trajectory, and how these are used to predict the endpoint of the trajectory. The third experiment illustrates the scalability of our approach when the library contains a large number of primitive movements. Finally, the fourth experiment showcases the application of our approach in a real world scenario featuring the Baxter robot.

Recognition

We start by illustrating the ability of our approach to perform recognition when the observed trajectory comprises segments from multiple primitive movements. Figure 5.12a shows the results observed when there are 6 DMPs in the library, and the observed trajectory is comprised of segments from the different trajectories. We note that, for some DMPs, the recognition weights may first decrease, as the system identifies another primitive as the most likely, and then increase, because the system decides



(a) Recognition weights for all DMP sequences at successive critical points.



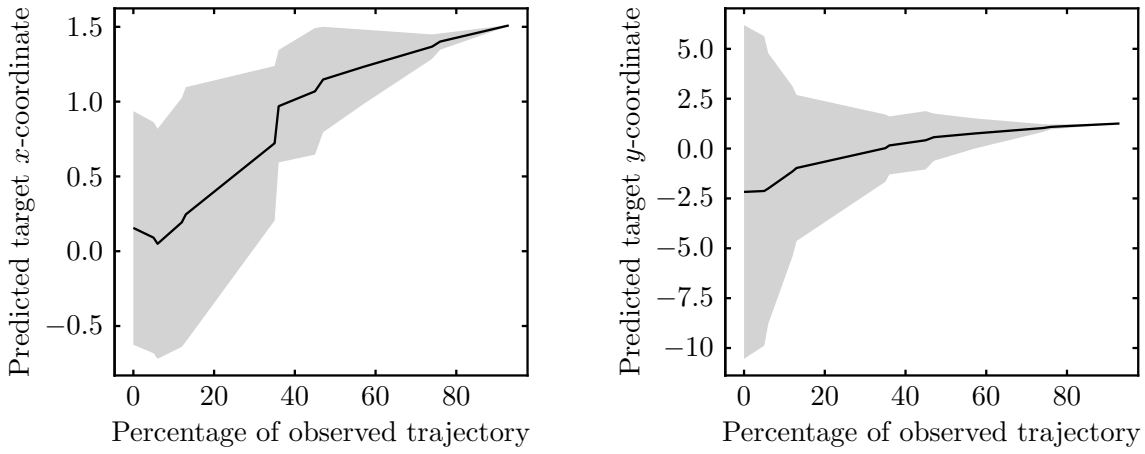
(b) Entropy of the prediction with the observation of new critical points.

Figure 5.12: Weight and entropy evolution during the recognition process, as larger portions of the trajectory are observed.

that better recognition is attained by switching to that particular primitive movement. The converse is also observed: the weight of some primitives first increases, as that primitive is the best to explain the observed movement, and then decrease, as switching to another primitive offers better recognition.

The algorithm updates the recognition weights every time a new critical point from the observed trajectory is discovered. Prediction can take place at any moment during the recognition phase; the best moment depends on the task, environment, and how much precision is required of the prediction.

To facilitate the analysis of the performance, we depict in Fig. 5.12b the evolution of the entropy in the recognition weights. We immediately note that the recognition process is slower if the observed trajectory is the composition of more than one movement, compared with a trajectory comprising a single movement. In fact, comparing the results in Fig. 5.12b with the results of the recognition using a single trajectory, in Fig. 5.4b, we note that the entropy decreases more slowly in Fig. 5.12b. This is due to the fact that, after the currently observed trajectory, we still consider the possibility that the current trajectory may not yet be over and, as such, there may be new primitive movements following the current trajectory. Hence, as we observe new critical points, there is always some level of uncertainty regarding the prediction that is only resolved as the trajectory concludes.



(a) Prediction of the target x coordinate.

(b) Prediction of the target y coordinate.

Figure 5.13: Uncertainty in the prediction of the target for the two dimensions of the trajectory. The solid line corresponds to the mean predicted target, $\hat{y}(\tau_{M+1})$, while the shaded area corresponds to the variance in the prediction.

Prediction

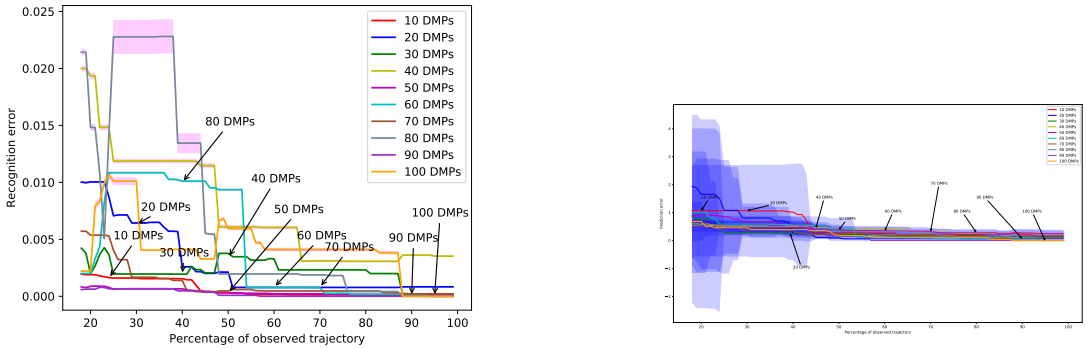
We also tested the ability of our approach to predict the target point of the observed trajectory. We depict in Fig. 5.13 the evolution of our prediction as more critical points are observed.

Note how the prediction approaches towards the target point, and the uncertainty decreases with the discovery of new critical points. Also, as seen in Fig. 5.12b, the entropy decreases slowly, which affects the prediction. The prediction process needs to observe the last part of the monitored trajectory to reach the last DMP in the chosen configuration and obtain a correct prediction, so slower convergence is to be expected.

Accuracy

We now test the ability of our approach to deal with large movement libraries. We compute the recognition error, given by $|\omega_{\max} - \omega_*|$, with $\omega_{\max}$ is the weight associated with the most likely DMP sequence, and ω_* is the weight for the *real* DMP sequence corresponding to the observed trajectory. We also compute the prediction error, given by $\|\hat{y}(\tau_{M+1}) - y(\tau_{M+1})\|$, where $\hat{y}(\tau_{M+1})$ is the predicted endpoint and $y(\tau_{M+1})$ is the actual endpoint.

In Fig. 5.14 we can see the average prediction and recognition errors averaged over 25 different trajectories, composed from segments of movements from a library of 85



(a) Recognition error.

(b) Prediction error.

Figure 5.14: The errors in recognition and prediction for the 25 observed trajectories, as a function of the observed trajectory. The solid lines depict the mean error values, while the shaded area depicts the variance in these errors.

primitive movements.

It is apparent from the figure that, while the prediction error decreases steadily (the trajectory approaches its endpoint, so the prediction errors tend to decrease), the recognition error fluctuates significantly, due to the fact that when a new segment is observed, the cost for changing the primitive motion generating a segment leads the search algorithm to, sometimes, "delay" the shift to a different primitive.

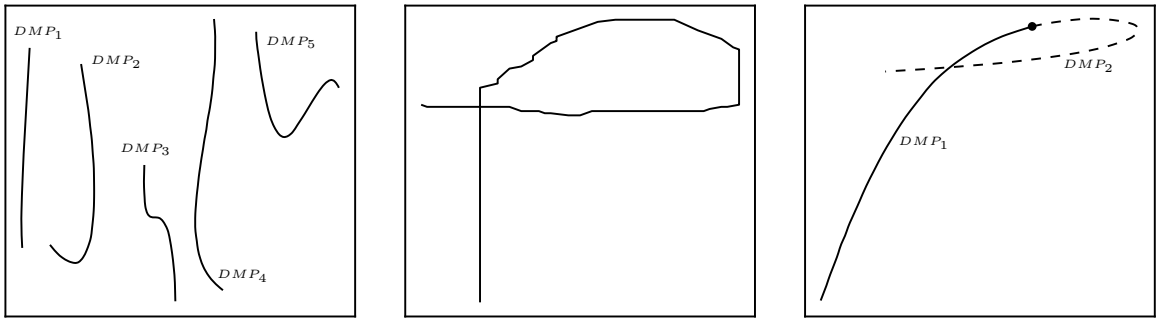
Improving the Knowledge of the Robot

We illustrate a practical application of our approach, where we recognize the letter "P" using a library of number trajectories (or parts thereof). Note, for example, that "P" can be formed by composing the movement for number "1" and part of the movement for number "3" (see Fig. 5.15).

Similarly, the output resulting from observing the trajectory for "B" would be three consecutive partial paths: "1" followed by half of "3" twice.

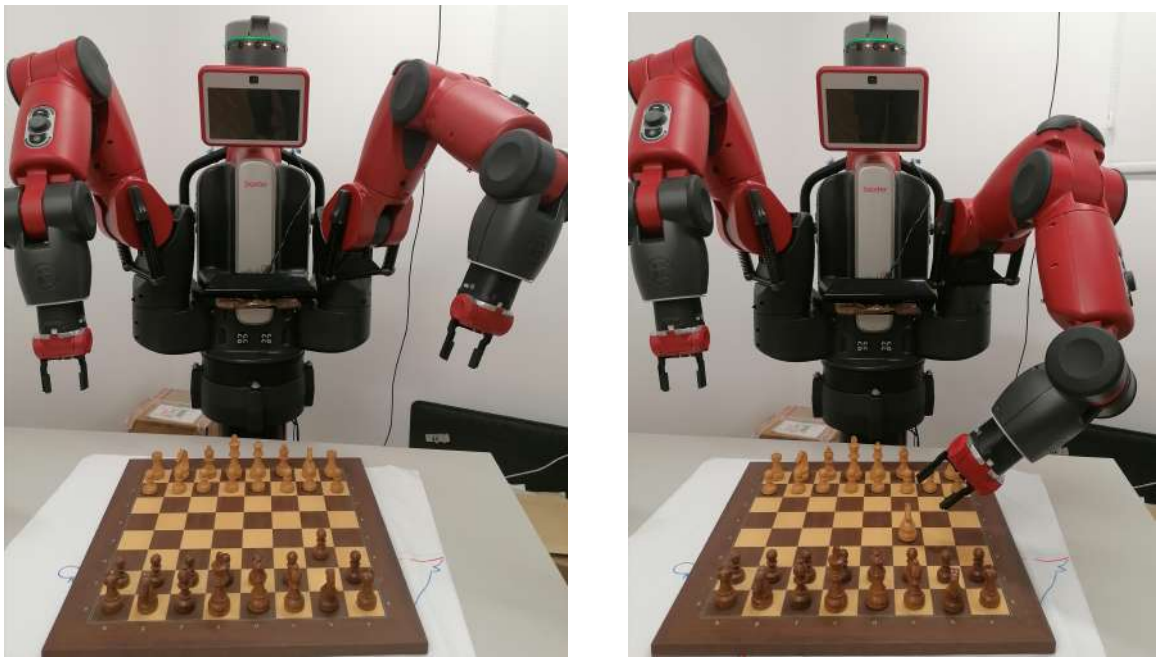
Experiment with Robot

In this experiment, Baxter is initially shown the movements of the Pawn (one step forward on the board) and the King (one step sideways on the board) in chess. Baxter then observes a human user performing the movement of the Knight (two steps forward and one step sideways). The Knight's movement can be seen as a composition of two consecutive movements from the Pawn, and Baxter successfully recognizes the



(a) DMPs library. (b) Observed trajectory. (c) Recognised DMP.

Figure 5.15: Recognizing the "P" letter using two learned DMPs in the DMPs library, which contains normalized trajectories, where the trajectory in 5.15b is constructed using the first DMP and the second one.



(a) (b)

Figure 5.16: Illustration of the human-interaction scenario considered in this section, where BAXTER succeed in anticipation to the user complex motion.

movement as the composition of the two simpler movements.

Baxter stores a Pawn's trajectory, presenting a one-step movement as a DMP. During the experiment, Baxter observes the human arm holding the Knight and discovers that the Knight's movement consists of two DMPs each one presents Pawn's movements. Consequently, Baxter can save a Knight's movement as a sequence of two Pawn movements by changing the second one's direction and target.

Baxter could now use the identified sequence of movements to replicate the Knight's movement using, for example, our framework.

5.5 Concluding Remarks

We present a simple yet robust method for recognizing and predicting the target of an observed trajectory. Our approach is applicable within the DMP framework without requiring an exhaustive comparison of all the trajectory points, considering only the critical points. Considering critical points alleviates the need for accurate time alignment and significantly decreases the computational burden associated with recognition and prediction. Moreover, as we noted in our discussion, often observing intermediate positions in the trajectory provides little additional information that can significantly help in recognition or prediction, as it does not add unique data that can make changes in the validation. In contrast, critical points provide unique data that contributes significantly to disambiguate between the different movements in the library, thus allowing efficient recognition and prediction.

We then extended our approach to consider also situations where the robot observes a trajectory, predicts its goal, and identifies the sub-trajectories that compose it. Our method works without training or prior knowledge of the observed trajectory and relies only on the movements in its library. The library serves as an initial knowledge base of the robot but can be improved as more trajectories are observed.

Our work is robust to changes in the observed trajectory compared to the original DMPs in the library, as it relies on comparing the ratios of change of velocity and angle, dealing efficiently with rotations, modulations, and scaling of the trajectory. Our approach uses critical points for efficient and effective trajectory identification, alleviating the need for time alignment. Our approach assumes that each segment in the observed trajectory is either the initial segment of a movement in its library or the continuation of the previous movement. Then, given a partial trajectory, our method is able to predict the most likely next target—i.e., the end-point of the movement currently being executed, if the latter is executed to the end. By using an effective search tree, our approach is able to run at execution time and provide an efficient way for action recognition and prediction.

Our statistical study illustrates the accuracy of this method in recognition and prediction—our results clearly illustrate the close relationship between the performance in terms of prediction and the recognition accuracy: correct recognition leads to accurate

prediction.

Finally, our approach is computationally efficient and can perform a recognition task in execution time, a key feature given its relevance in human-robot interaction tasks like that presented in our experiments.

Chapter 6

Coordinated Optimization of Multiple DMPs

This chapter addresses the last research challenge of the thesis—namely, how can a robot, endowed with a library of pre-learned primitive movements, generate multiple simultaneous *coordinated* robotic movements, adapting and optimizing those in the library, to complete one collaborative task? The work in this chapter can thus be seen as a follow-up to the work in Chapter 4 that now considers collaborative task and the existence of multiple robots/manipulators. Specifically, we extend the framework in Chapter 4 to accommodate coordinated execution of multiple DMPs in robots with multiple manipulators or—alternatively—multiple robots with a single manipulator. We investigate mechanisms to jointly optimize multiple DMPs to perform one task in a coordinated fashion. The joint trajectory is built from initial DMPs learned for a single manipulator, and its optimization must comply with task-specific constraints.

In previous chapters, we focused our discussion on scenarios in which the task to be executed consisted of a single movement. This was the case in the framework of Chapter 4 as well as the recognition/prediction approach in Chapter 5. But many practical tasks impose the need for collaborative tasks. For example, moving a big or awkward object may not be possible for a single robot, but it may be possible for two or more robots working in parallel, as long as they act in a coordinated fashion. In this chapter we consider this exact problem, of having multiple robots act in a coordinated fashion to solve a common task.

The consideration of multiple actors has the important advantage of significantly simplifying certain tasks that are complex to accomplish by a single robot, but which a group of robots can solve in a coordinated manner in a manner that is simpler, more accurate, more robust, and less prone to errors. The simpler the tasks, the higher, the more successful, and the more accessible and reliable the degree of cooperation will be.

It is important to note that the problems considered in this chapter do not include tasks that require breaking it down into sub-tasks that each robot can perform independently, as often happens in multi-robot settings [61]. Instead, we are concerned with tasks that require the execution of simultaneous movements in a coordinated and synchronous manner, while still taking into consideration the conditions and constraints of the environment. Such situations are common, for example, in robotics [87] and games [106].

Several approaches exist that address the problem of coordinating multiple motions.

For example, Stavridis and Doulgeri [121] address the problem of assembling two parts where—instead of using a manipulator to just hold one part steady and the second to assemble the second part—both manipulators move to render the assembly easier. The movement of the two manipulators is described using a dynamical system; the trajectories of the system are then adjusted to ensure that the task constraints are verified. Constraints include the desired position for the parts to be assembled, as well as the obstacles in the environment that should be avoided. There are two key differences from our work. First, an initial movement for both manipulators is provided upfront, which is then optimized using a *hierarchical quadratic programming approach*. In our work, we depart from a single-manipulator movement that is then used to construct the motion of *both* manipulators; additionally, following our approach in Chapter 4, we adopt a BBO instead of a quadratic programming optimizer.

Gams et al. [29] use *iterative learning control* (ILC) to learn a coupling term that limits two DMPs to enforce coordination. Given two DMPs, Gams et al. [29] define one DMP as the "leader" and the other as the "follower". The follower is constrained by the leader to ensure successful completion of the task. This requires that the leader DMP be already provided in such a way that successful task completion is possible. In contrast, our method optimizes both movements simultaneously to comply with the task constraints.

Nemec et al. [93] propose an approach for bimanual robots that engage in cooperative task with humans. The motion for the robot is taught by demonstration and stored as a (bi-manual) DMP. At execution time, the robot stiffness is initially set to allow the robot to comply with the human motion—i.e., the human "retains" control of the task. With successive executions, the robot gradually "takes control" from the human, by adjusting the stiffness of the movement execution. They propose *speed-scaled DMPs* as an extension to DMPs that allows for compliant execution. In the continuation, we present our own approach for collaborative tasks that need the coordinated movement of multiple robot arms. We again DMPs as the base representation for robot movements, and assume that our robot(s) have access to a library of pre-learned movements. Then, given a new environment configuration and a new task, our approach autonomously builds a set of optimized trajectories that consider and meet the constraints imposed by the task and the new environment. Our system can generate and optimize multiple

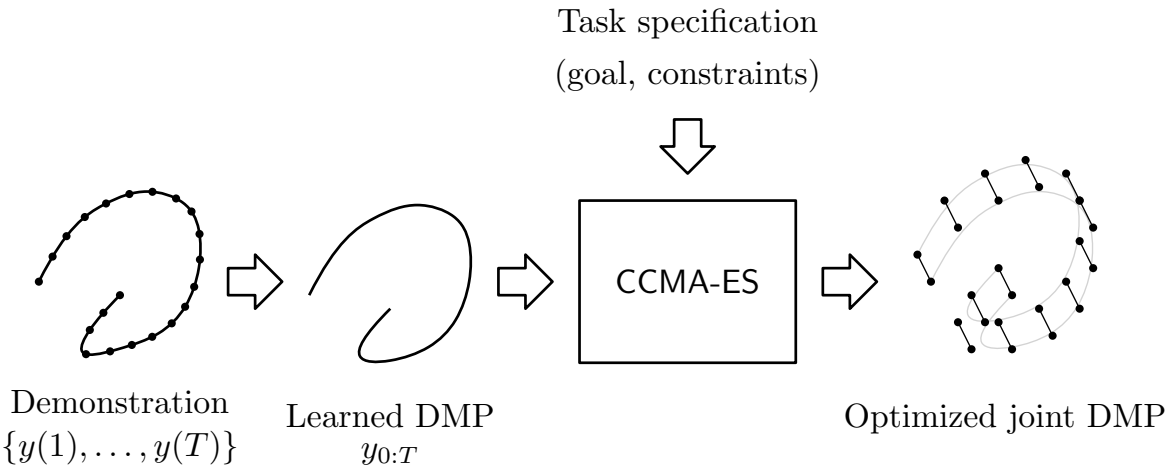


Figure 6.1: Pipeline for the proposed approach.

coordinated movements from a single movement learned from a demonstration in an environment different from the execution environment. We illustrate the application of our approach both in a simulated environment and in a (simulated) Baxter robot.

6.1 Coordinated Trajectory Optimization

The pipeline for the proposed approach is illustrated in Fig. 6.1. Following the general approach in Chapter 4, we collect a (single manipulator) demonstration from a human user that is then used to build a DMP (or a library thereof). Then, given the specification of a task (goal, constraints), we use the movements in the library to generate and optimize a *joint DMP*, providing the desired trajectory for the multiple manipulators. The environment in which the task is carried on (obstacles, environment layout) may differ from where the demonstration was originally provided, and we leave it to the optimizer to adjust the DMPs accordingly.

6.1.1 CCMA-ES for Coordinated Motion Generation

We adopt, as the optimizer, a variation of CMA-ES known as *constrained CMA-ES*, or CCMA-ES. This method is discussed in Chapter 2 for constrained optimization problems. We formulate the problem of coordinated motion as a constrained optimization problem, where the task and environment define the objective function (as in Chapter 4), and the constraints ensure coordination between the multiple DMPs being optimized. While different approaches exist for constrained optimization of DMPs [20], we follow up on

our approach from Chapter 4 and adopt CCMA-ES.

In the context of our work, CCMA-ES 2.2.1 is used to generate/optimize the weights of the force function describing the motion of the two manipulators. Given a specific task—typically described by a target configuration for the robots' movements—and a set of constraints that the motion of the two manipulators must verify—for example, the two end-effectors must maintain a constant distance—our approach proceeds by:

- Modulating the original DMP for each of the two manipulators target position. As a result, we obtain a single DMP $(y^1_{0:T}, y^2_{0:T})$ that describes the motion of the two manipulators, synchronized by a shared canonical system.

- Optimizing the joint DMP parameters (the weights of the corresponding force function) while ensuring that the constraints are verified.

For example, we can optimize two DMPs to allow a robot to move a rigid object of length D by solving the constrained optimization problem

$$\underset{w_1,\ldots,w_N}{\text{minimize}} \qquad J(y^1_{0:t}, y^2_{0:T}) \tag{6.1a}$$

$$\text{subject to} \qquad \|y^1(t) - y^2(t)\| = D, \quad t = 0, \ldots, T, \tag{6.1b}$$

where J is the objective function (4.1), used in Chapter 4 (see page 45). We note that obstacles in the environment can be incorporated either into the objective function J or as actual constraints. We note also that the DMP used to generate the original movement for the two manipulators can be obtained by combining several simpler DMPs, following Chapter 4. This means that the overall framework enables the robot to generate complex motions in different environments while verifying task-specific coordination constraints.

Moreover, for moving obstacles, we can also apply the method described in Chapter 4, applying the same modulation matrix on both DMPs—as depicted in Fig 6.2. When the moving obstacle is close to the coordinated DMPs, we build the modulation matrix M for the DMP closest to the obstacle and apply it to both DMPs at the same state.

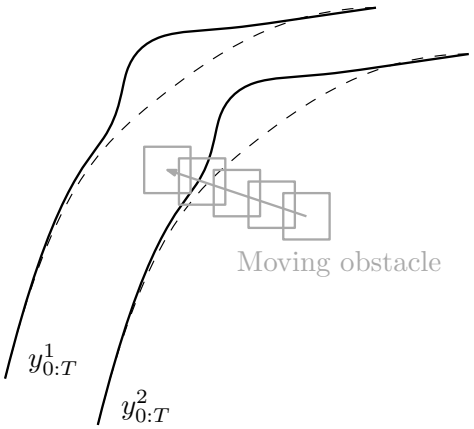


Figure 6.2: Application of the modulation matrix to the two DMPs, y^1 and y^2, to avoid a moving obstacle.

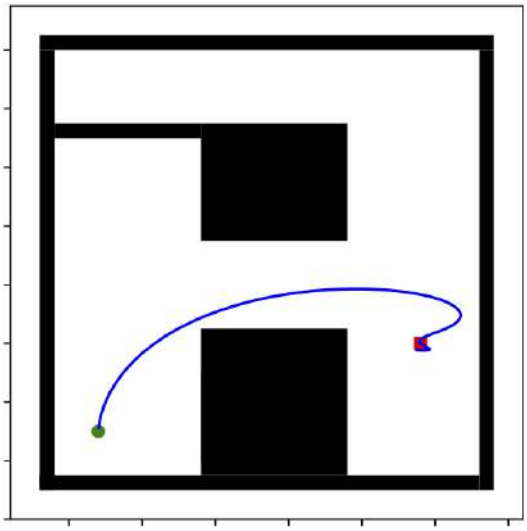


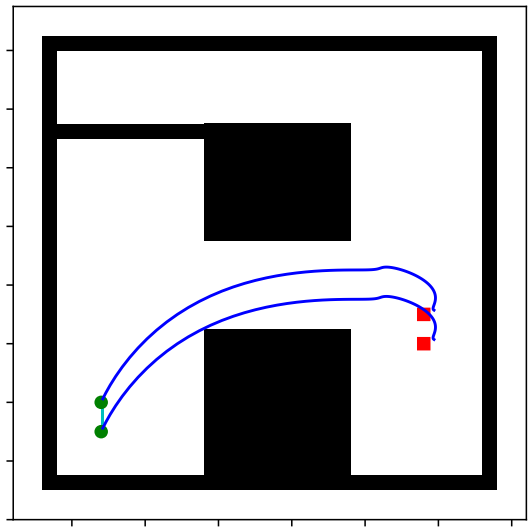
Figure 6.3: Initial DMP used in the experiments, generated using the framework from Chapter 4. The green circle and red square correspond to the start and endpoints.

6.2 Experiments and Results

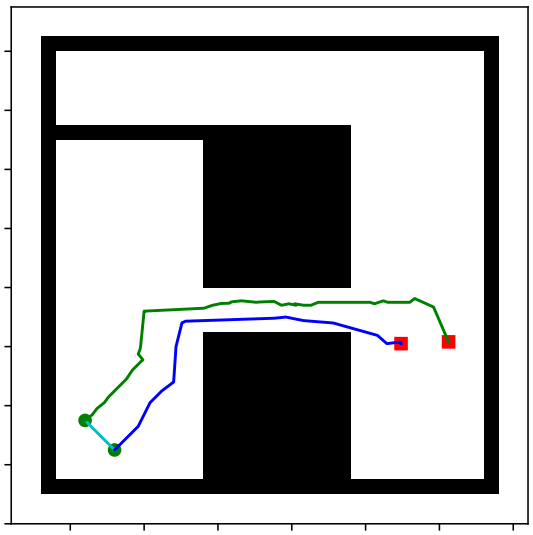
We now illustrate the application of our method both in simulation and using the Baxter robot. We start by presenting results in simulation, where we show, side-by-side the results obtained in a geometric simulator and in the more realistic Baxter simulator.

6.2.1 Simulation Results

We start by using the approach in Chapter 4 to generate an initial DMP that successfully navigates the environment. The resulting DMP is shown in Fig. 6.3. Note that the gap between the obstacles is relatively wide, so the optimization algorithm can easily find a



(a) Joint DMPs optimized to achieve a target configuration while maintaining the distance between the two manipulators constant. The corridor width is longer than the object length by double its length.



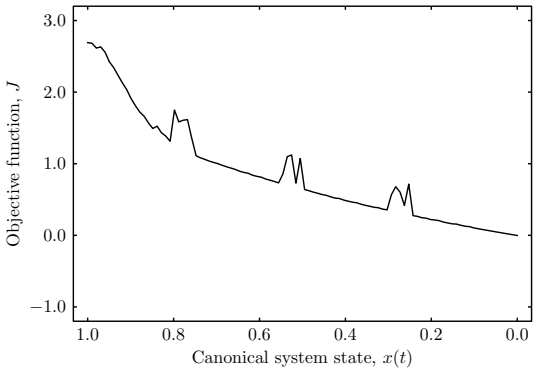
(b) Joint DMPs optimized to achieve a target configuration while maintaining the distance between the two manipulators constant in an environment with a narrower passage. The corridor width is smaller than the object length by 4 unit scales.

Figure 6.4: Optimized joint DMPs in two configurations of the environment—the original configuration and a configuration with a narrow passage. The green circles and red squares correspond to the start and endpoints.

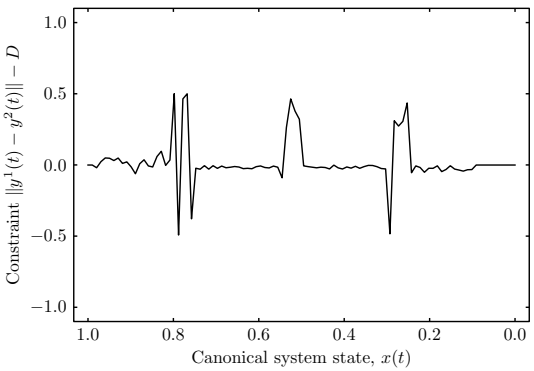
path that leads the movement from the initial position (the green circle) to the target position (the red square).

We then ran our proposed approach in the same environment, imposing a restriction where the distance between the two DMPs must be maintained constant. The task consists of moving a rigid object of length D from one place to another. To succeed in this task, the two manipulators must be coordinated to ensure that the object does not fall or hit any obstacle in the environment. Therefore, a constraint is placed on the optimization process to enforce a fixed distance between the two endpoints of the robotic arms. The environment still retains the two obstacles separated by a corridor. The goal is to optimize the two DMPs describing the motion of the two arms to allow for safe crossing of the corridor.

The resulting trajectories are depicted in Fig. 6.4a. As the plot shows, the optimizer is able to successfully drive both DMPs from the initial position (green circles) to the target position (red squares), maintaining the distance between the two DMPs and avoiding the obstacles. Figure 6.4b shows the result of our approach in a second environment, now featuring a narrower passage. In order to comply with the obstacles



(a) The value of the objective function steadily approaches 0.



(b) Constraint violation during execution.

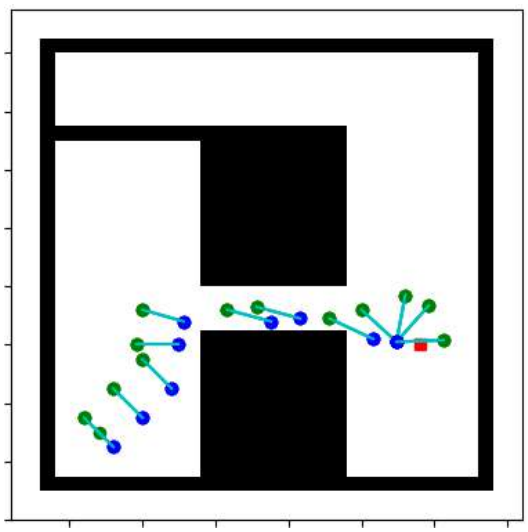
Figure 6.5: Value of the objective function and constraint during the execution of the movement in Fig. 6.4b.

in the environment and the constraints imposed by the task, the trajectories are much less smooth than the ones in Fig. 6.4a.

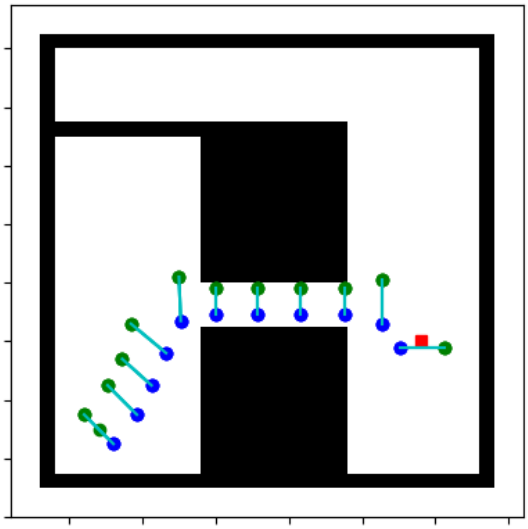
Figure 6.5 shows the value of the objective function and the constraint as the movement progresses, in the environment from Fig. 6.4b. We can see that the DMPs steadily approach the target (the value of the objective function steadily approaches 0). We can also see some jumps in the objective function, when the DMP approaches the obstacles in the middle of the environment. Similarly, the constraint remains approximately 0 throughout the trajectory except in those same points, where the constraints exhibit some small violations.

Finally, we also compare our approach with an approach where the DMPs are jointly taught but where constraints are not considered during the optimization [5]. The resulting trajectories are shown in Fig. 6.6, where we now explicitly showcase the different time steps, for easier inspection of whether the constraints are verified or not. When the learned DMPs are optimized to address environment restrictions (such as obstacles), but no task constraints are enforced to ensure coordination, the resulting DMPs may violate such constraints, even if the original joint DMP is learned by demonstration ensuring that the constraints are met—in the example, the original joint DMP was demonstrated in the environment with larger gap, so the constraints were easily verified in the demonstration. In contrast, our approach adapts to the new obstacle configuration while ensuring that the constraints are verified.

We also compared our work with the approach of Gams et al. [29], encountering similar difficulties—namely, the inability to include new environment information when



(a) Our method can enforce the task constraints to pass through the tiny corridor and achieve a collaborative task.



(b) The task constraints are not observed in the optimized movement, although they were present in the learned movement.

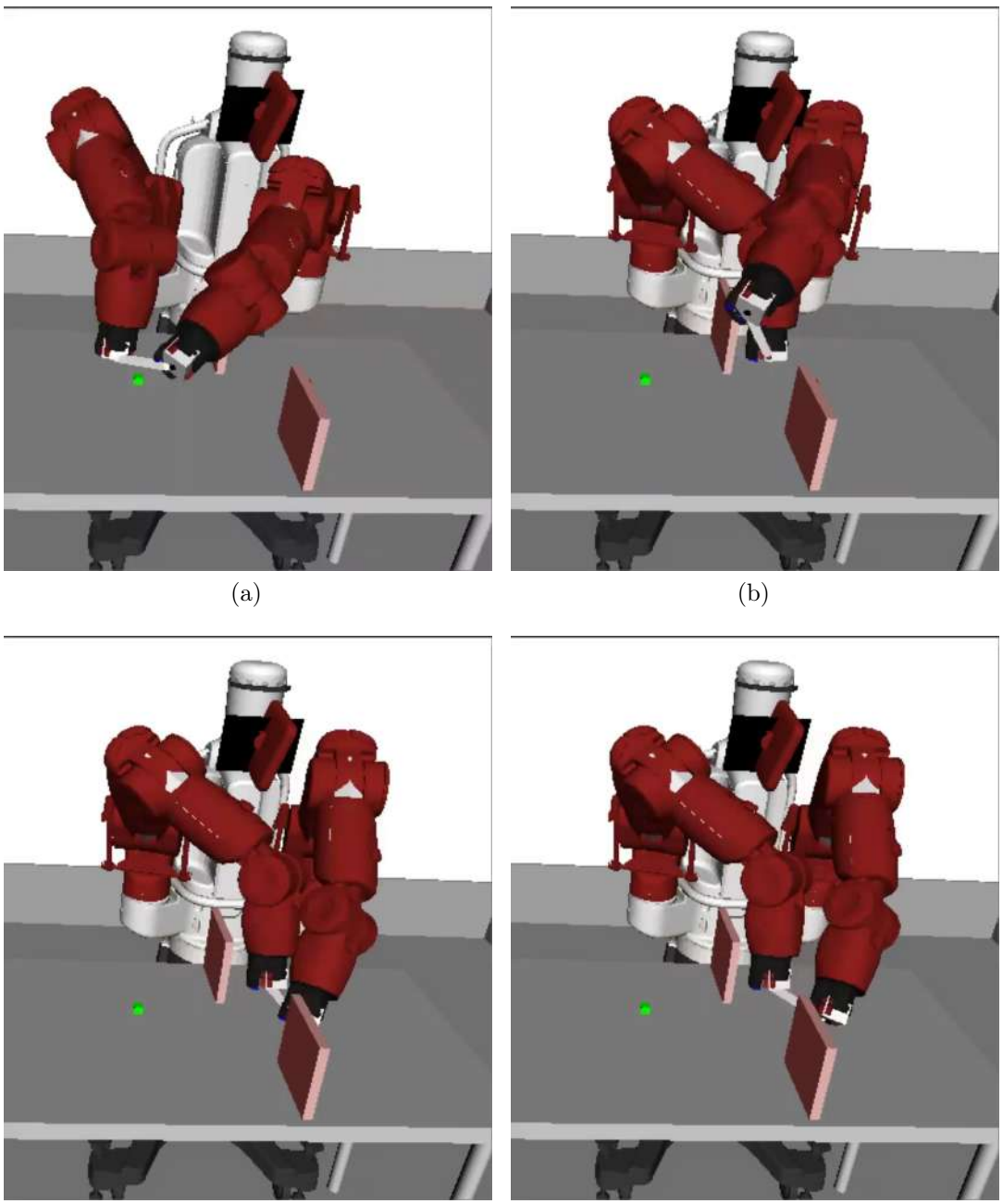
Figure 6.6: Comparison of our approach with an approach not enforcing constraints [5]. Green and blue circles represent different points along each DMP. The red squares correspond to the endpoint, and the naive bar between the two points represents the object that needs to be transferred using two optimized DMPs. The right figure shows the violation of the constraints on the distances between both corresponding states inside the path. While in the left figure, all corresponding states are following the constraints of the task under the environmental conditions. The corridor width is smaller than the object length by 4 unit scales.

adjusting the motion. Our approach includes the environmental information obstacles inside our cost function that evaluate the samples compatible with the task conditions and constraints. Also, the need for human annotation to determining the leader and follower DMPs restricts the outcome of this method, since it may prevent the algorithm from finding specific solutions.

We conclude by showcasing the same results now using the Baxter simulator. Figure 6.7 shows different snapshots of the resulting trajectory, where the vertical obstacles reproduce the layout of the environment in the previous geometric simulation. using the joint DMPs computed by our approach, the robot was able to complete the task and bypass the obstacles successfully.

6.2.2 Robot Experiments

We also applied our work in the physical Baxter robot, where the task was, once again, to move a solid object (a "stick") between two pre-specified positions using Baxter's two



(a) (b)

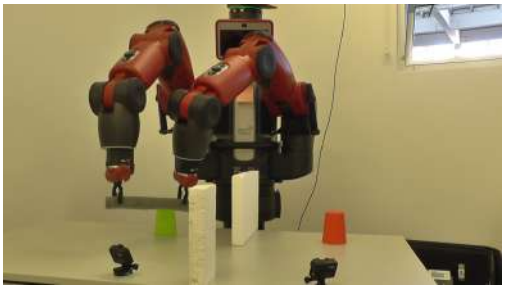
(c) (d)

Figure 6.7: Snapshots of the Baxter simulator executing the trajectories generated by our approach to accomplish a coordinated task. Our approach could successfully optimize two DMPs for Baxter's arms to achieve a collaborative task that imposed constraints on the trajectory. Baxter needs to transfer an object between two positions. The green circles and red squares correspond to the start and endpoints.

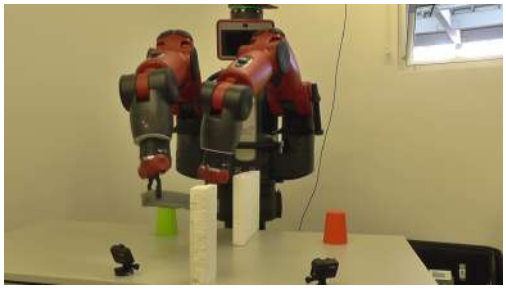
arms.

Figure 6.8 illustrates the path executed by Baxter, showcasing the movement executed by the robot when the gap between obstacles is broad, allowing the robot to successfully carry the stick across the pass without significant adjustment of the

(a)



(b)



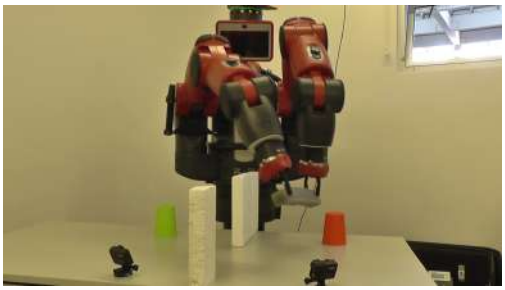
(c)

(d)

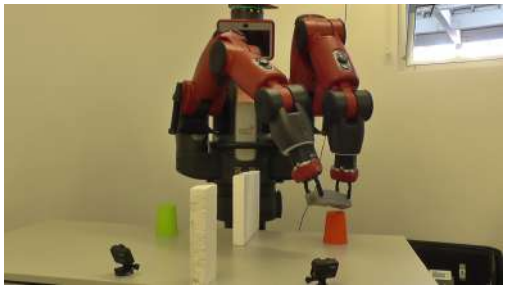
(e)

(f)

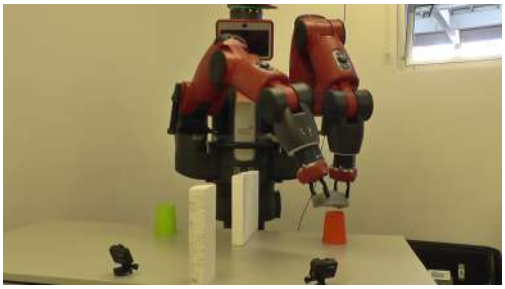
(g)



(h)



(i)



(j)

Figure 6.8: Baxter uses its two arms by coordinating its DMPs to do the task of transporting an object through a path that can accommodate this object's size. The corridor width is longer than the object length by 4 unit scales.

trajectories. The constraints allow the robot to go through the pass without colliding with the obstacles, while coordinating the two DMPs governing the individual motion of each arm.

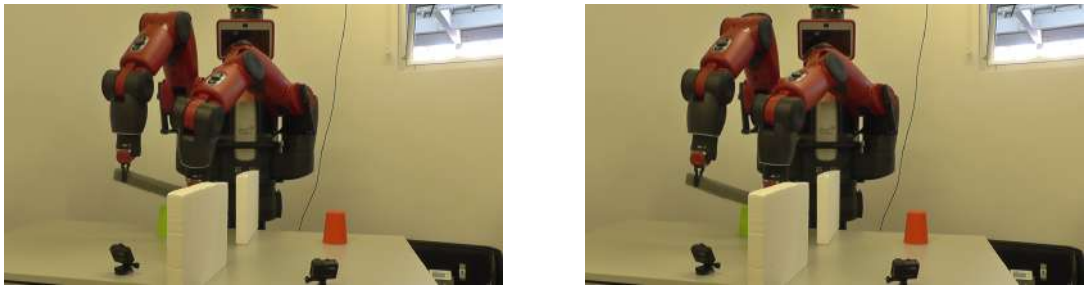
In Figs. 6.9 and 6.10 we showcase two views of the movement executed by Baxter when the passage is significantly reduced. With the narrow passage between the obstacles, Baxter must change the orientation of the stick in order to successfully pass through. The figures show two distinct views (captured with different cameras) of the same trajectory. We notice that—unlike in the first scenario, where the DMPs could be used almost unchanged—in this setting the robot required a more significant adjustment of the joint DMP in order to go through the path without colliding with the obstacles, while maintaining the distance between the two end effectors.

6.3 Conclusions

In this chapter, we contributed a novel approach for movement generation in collaborative tasks that require more than one robot arm to complete, using a single initial DMP that is then optimized to operate in the environment. Our approach builds on the framework of Chapter 4 to build an initial DMP already taking the environment into account; this DMP is then used to build the joint DMP that is then optimized to match the task-specific constraints. All DMPs are optimized together to accomplish the desired collaborative task and ensure successful coordination.

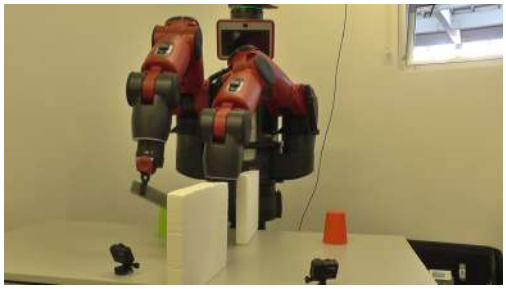
Our method is also amenable to further optimization, to overcome the increasing difficulty imposed by the obstacles while maintaining the synchronization of the individual DMPs, thus efficiently transferring a demonstrated movement to novel scenarios involving multiple robots/arms. Besides that, our results show the ability to adapt to both different environment changes and the task conditions.

Our research also opens exciting avenues for future work. For example, it would be interesting to find a way to optimize and coordinate multiple DMPs to bypass moving obstacles while maintaining collaborative task performance. The approach briefly discussed in Section 6.1.1 could be a starting point, but some experimental validations are necessary to assess whether the deformation imposed by the modulation matrix indeed maintains the trajectories within the constraints. Besides, the approach

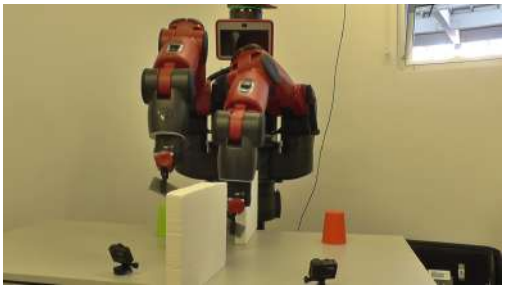


(a)

(b)



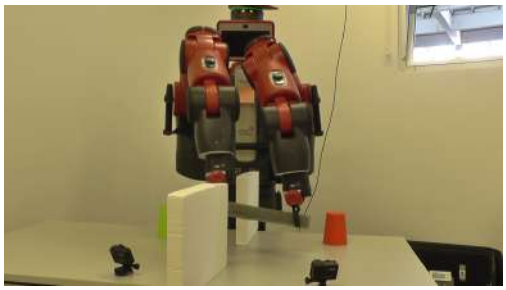
(c)



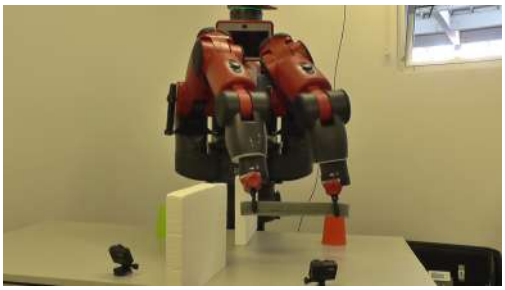
(d)

(e)

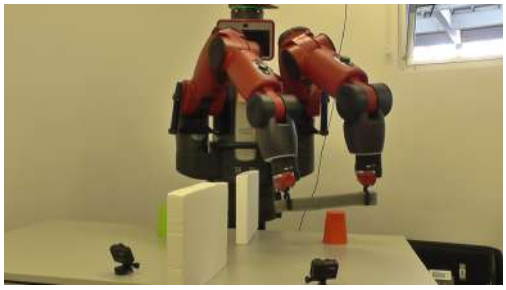
(f)



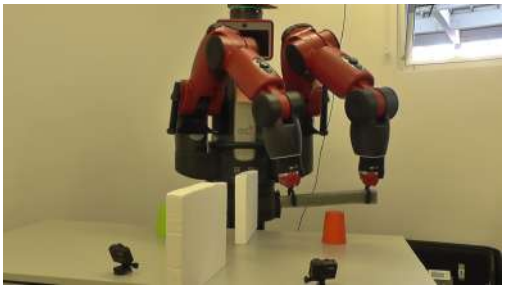
(g)



(h)



(i)



(j)

Figure 6.9: Baxter uses its two arms by coordinating its DMPs to do the task of transporting an object through a tiny path that cannot accommodate this object's size without hitting obstacles.The corridor width is smaller than the object length by 4 unit scales.

(a)

(b)

(c)

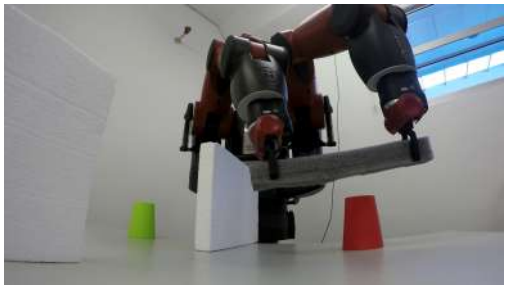
(d)

(e)

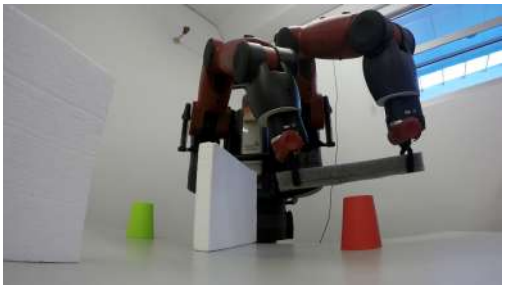
(f)

(g)

(h)



(i)



(j)

Figure 6.10: Another view of the Baxter experiment is when Baxter uses its two arms by coordinating its DMPs to transport an object through a tiny path that cannot accommodate its size without hitting obstacles. The corridor width is smaller than the object length by 4 unit scales.

proposed does not specifically consider the constraints, and it would be interesting to extend the modulation-based approach to indeed enforce those constraints.

Chapter 7

Conclusion and Future Work

This chapter concludes the thesis, bu summarizing the main contributions and highlighting some directions for future work.

7.1 Overview

We addressed the general problem of endowing a robot with the ability to execute complex manipulation tasks in collaboration with human users, building on a previously learned repertoire of primitive movements. Specifically, we addressed the above research problem from three different perspectives:

- First, we looked into the composition, adaptation, and optimization problem of primitive movements to yield complex movements. In this context, we contributed an end-to-end approach to learning and decomposing user demonstrations into DMPs and then optimizing and combining the resulting DMPs to perform complex motions. One important role of our proposed framework is to increase the acquired knowledge of the robot, namely by incorporating the process of acquisition of the repertoire of primitive movements into the overall framework of movement learning and generation.

  To this purpose, we look at the information provided by the trajectories *critical points* to break down demonstrations provided by humans into smaller, more homogeneous segments. It is the segments thus identified that are incorporated into the movement library.

- We then considered the problem of movement recognition and prediction, given a library of pre-learned primitive movements. In this context, we contributed a novel movement recognition method that, given a library of previously observed movements, identifies a partially observed trajectory from among those in the library, predicting its (unobserved) target point. The fact that our approach does not require the full trajectory to be observed is fundamental, as it can be used in run-time for actual human-robot interaction scenarios. We also extended this approach to accommodate the recognition of compound movements.

  Much like our first contribution, we again rely on the information provided by the trajectories critical points, comparing the observed trajectory and the known

trajectories precisely at these points. By considering the relative changes in the trajectory between critical points, we attain a process of recognition that is robust to scaling, rotation and modulation.

- Finally, we addressed the problem of optimizing multiple DMPs toward the coordinated execution of compound movements. In this context, we extended our framework for optimizing and combining pre-learned movements to now consider tasks requiring the execution of multipl, simultaneous trajectories by different manipulators in a coordinated fashion.

There are several key "ingredients" supporting the research in this thesis. The first—a core component in all our contributions—is the use of a library of pre-learned movements. The movements in the library can be both factory-programmed and movements learned by the robot from demonstrations provided by users, and are used to generate novel complex movements as well as to recognize movements observed by the robot. The use of this library as a core component of our contributions serves two important purposes in the wider vision of the thesis. On the one hand, the fact that the trajectories generated by our robot build on previously learned trajectories enables some degree of customization that may be harder to ensure using a pure planning approach. Specifically, the movements of the robot will—as much as the task and environment allow—follow the movements taught by the user. On the other hand, although the tasks considered herein all consist of movements from an initial configuration to a final configuration, the library of atomic actions can be seen as a discrete action space that can fit into a hierarchical architecture for planning and decision making in more complex scenarios.

A second key ingredient is the use of *dynamic movement primitives* to represent the motions performed by the robot. DMPs offer several important advantages that we abundantly rely on. The representation power, easy modulation, and robustness to perturbations are well-documented properties of DMPs [114]. However, for our purposes, DMPs are also amenable to optimization using black-box optimizers such as CMA-ES and CCMA-ES, and can be combined in a sequential manner, ensuring that the resulting movement is still smooth.

A third key ingredient is the use of *critical points*. We use critical points for segmenting demonstrations into smaller, more homogeneous movements that are then

incorporated into our movement library; however, even more fundamentally, we use critical points for movement recognition. The ability of a robot to recognize and predict observed movements is a core skill that any robot interacting with humans should possess; the ability to predict what a user is trying to do can greatly contribute for an effective and intelligent interaction between robots, or between robots and humans. The proposed approach leverages the information available at critical points to predict and make quick decisions during execution. In recognizing compound movements, we also make use of critical points to determine whether the current segment is the beginning of a new movement or the continuation of the previous movement.

It is worth mentioning that—although we do not do it—our recognition framework could easily be used to *prune* the DMP library for redundant movements, if the need for such pruning arises. Additionally, the ability of our approach to recognize compound movements can also be leveraged to replicate observed complex movements *without prior training*, simply by matching the sequence of observed movements to those in the library and composing them using the framework in Chapter 4.

7.2 Summary of Contributions

Summarizing, the main contributions of this thesis are threefold:

- An end-to-end approach that allows a robot to take as input user demonstrations and break these down into DMPs that are then stored in a library of "atomic actions". Then, when faced with a novel task, the proposed framework combines and optimizes the DMPs in the library to allow the robot to generate novel complex motions in possibly unseen scenarios. This contribution was published in Kordia and Melo [52].

- A movement recognition method that, given a library of known movements, identifies a partially observed trajectory as one of those in the library, using the outcome of the recognition process to then predict the target of the movement. We also contribute an extension of our recognition method to compound movements. These contributions were previously published in Kordia and Melo [54] and Kordia and Melo [53].

- An extension of our end-to-end framework that considers scenarios involving multiple simultaneous and coordinated movements. Given a library of individual trajectories, a coordinated trajectory is built and optimized to perform a coordinated task. This contribution is currently being prepared for submission at the 2023 IEEE International Conference on Robotics and Automation.

7.3 Thesis Publications

The work in this thesis led to the following publications:

1. A. Kordia and F. Melo. "An end-to-end approach for learning and generating complex robot motions from demonstration." In *Proc. 16th International Conference on Control, Automation, Robotics and Vision*, pp. 1008-1014, 2020.

2. A. Kordia and F. Melo. "Movement recognition and prediction using DMPs." In *Proc. 2021 IEEE International Conference on Robotics and Automation*, pp. 8544-8550, 2021.

3. A. Kordia and F. Melo. "Compound movement recognition using dynamic movement primitives." In *Proc. EPIA Conference on Artificial Intelligence*, pp. 456-468, 2021.

4. A. Kordia and F. Melo. "Optimizing and coordinating multiple DMPs under constraints for complex manipulation tasks." To be submitted to the 2023 IEEE International Conference on Intelligent Robots and Systems IROS.

7.4 Future Work

In this work, we relied on the representation of movement with DMPs. It is possible to take advantage of the features of other models to represent movement, such as the mix between two ProMPs movements to implement a part of a movement as in the framework of Chapter 4.

In Chapter 4, we teach a robot how to build a complex trajectory from several simpler trajectories in the library. However, although the robot may be able to devise a complete trajectory for the task at hand, parts of the execution may be more critical

than others, and we may want to take into consideration that the robot may be able to perform certain movements more reliably than others. It could be important for the robot to be able to autonomously evaluate the reliability of the execution of the trajectory as a whole and, identify parts that may require enhancements, be rebuilt, or further optimized. As such, an interesting avenue for future research could be the partial optimization/reconstruction of the trajectory.

Another important feature of the approach proposed in Chapter 4 is the fact that we build complex movements using a greedy approach. It could be interesting, on one hand, to analyze some of the theoretical properties of such choice; on the other hand, it could be interesting to consider alternative heuristics (for example, beam search) that may strike a compromise between optimality and computational efficiency.

Another important aspect that is worth exploring is related with the fact that—all throughout the thesis—we worked in Cartesian space. In other words, we considered the trajectories of the end-effector, to a great extent disregarding the kinematics of the robot. While most of our contributions can be naturally extended to work in joint space, there are a number of aspects that we do not consider and which are very relevant even for the scenarios considered in this thesis. For example, in the process of optimization, the kinematic limitations of the robot should be considered. An interesting avenue for future research would be to also consider—in the optimization process—the possibility of self-collisions. This is a particularly relevant problem when considering multiple manipulators, since each manipulator is, naturally, an obstacle that the other should take into account in the optimization process.

Overall, the contributions of the thesis provide a unified framework for learning, generating and recognizing complex movements building on prior knowledge of the robot, providing robots with a set of important skills that can open the door for wider deployment of robots in our everyday lives and richer interaction between robots and human users.

Bibliography

[1] A. Abdolmaleki. *Information theoretic stochastic search*. PhD thesis, University of Aveiro, 2018.

[2] S. Ahmad and S. Luo. Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in cartesian space. Technical Report TR-EE 88-14, School of Electrical Engineering, Pardue University, 1988.

[3] B. Akgün, D. Tunaoglu, and E. Sahin. Action recognition through an action generation mechanism. In *Proc. 10th Int. Conf. Epigenetic Robotics*, pages 3–10, 2010.

[4] P. Alves-Oliveira, S. Petisca, F. Correia, N. Maia, and A. Paiva. Social robots for older adults: Framework of activities for aging in place with robots. In *Proc. 7th Int. Conf. Social Robotics*, pages 11–20, 2015.

[5] H. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters. Generalization of human grasping for multi-fingered robot hands. In *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 2043–2050, 2012.

[6] H. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters. Interaction primitives for human-robot cooperation tasks. In *Proc. 2014 IEEE Int. Conf. Robotics and Automation*, pages 2831–2837, 2014.

[7] G. Arampatzis, D. Wälchli, P. Weber, H. Rästas, and P. Koumoutsakos. (μ, λ)-CCMA-ES for constrained optimization with an application in pharmacodynamics. In *Proc. Platform for Advanced Scientific Computing Conference*, pages 1–9, 2019.

[8] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 27:469–483, 2009.

[9] D. Arnold and N. Hansen. A (1+1)-CMA-ES for constrained optimisation. In *Proc. Genetic and Evolutionary Computation Conference*, pages 297–304, 2012.

[10] M. Asada, M. Ogino, S. Matsuyama, and J. Ooga. Imitation learning based on visuo-somatic mapping. In *Proc. 9th Int. Symp. Experimental Robotics*, pages 269–278, 2006.

[11] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[12] A. Auger and N. Hansen. Tutorial CMA-ES: Evolution strategies and covariance matrix adaptation. In *Proc. 14th Companion Conf. Genetic and Evolutionary Computation*, pages 827–848, 2012.

[13] K. Baraka, P. Alves-Oliveira, and T. Ribeiro. An extended framework for characterizing social robots. In *Human-Robot Interaction: Evaluation Methods and Their Standardization*, pages 21–64. Springer, 2020.

[14] A. Bauer, D. Wollherr, and M. Buss. Human-robot collaboration: A survey. *Int. J. Humanoid Robotics*, 5(01):47–66, 2008.

[15] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Incremental natural actor-critic algorithms. In *Adv. Neural Information Processing Systems 20*, 2007.

[16] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.

[17] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] M. Black and A. Jepson. Recognizing temporal trajectories using the condensation algorithm. In *Proc. 3rd IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 16–21, 1998.

[19] M. Black and A. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proc. 5th Eur. Conf. Computer Vision*, pages 909–924, 1998.

[20] C. Cardoso, L. Jamone, and A. Bernardino. A novel approach to dynamic movement imitation based on quadratic programming. In *Proc. 2015 IEEE Int. Conf. Robotics and Automation*, pages 906–911, 2015.

[21] B. Chandrasekaran and J. Conrad. Human-robot collaboration: A survey. In *Proc. IEEE SoutheastCon*, pages 1–8, 2015.

[22] J. Colgate, W. Wannasuphoprasit, and M. Peshkin. Cobots: Robots for collaboration with human operators. In *Proc. Int. Mechanical Engineering Congress and Exhibition*, pages 433–439, 1996.

[23] A. Colomé and C. Torras. Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes. *IEEE Trans. Robotics*, 34(3):602–615, 2018.

[24] O. Dermy, A. Paraschos, M. Ewerton, J. Peters, F. Charpillet, and S. Ivaldi. Prediction of intention during interaction with iCub with probabilistic movement primitives. *Frontiers in Robotics and AI*, 4:45, 2017.

[25] M. El-Shamouty, X. Wu, S. Yang, M. Albus, and M. Huber. Towards safe human-robot collaboration using deep reinforcement learning. In *Proc. 2020 IEEE Int. Conf. Robotics and Automation*, pages 4899–4905, 2020.

[26] Z. Feng and T. Cham. Video-based human action classification with ambiguous correspondences. In *Proc. 2005 IEEE Computer Society Confer. Computer Vision and Pattern Recognition—Workshops*, pages 82–82, 2005.

[27] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, 2003.

[28] R. Galin and R. Meshcheryakov. Human-robot interaction efficiency and human-robot collaboration. In *Robotics: Industry 4.0 Issues & New Intelligent Control Paradigms*, pages 55–63. Springer, 2020.

[29] A. Gams, B. Nemec, A. Ijspeert, and A. Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Trans. Robotics*, 30 (4):816–830, 2014.

[30] S. Green, M. Billinghurst, X. Chen, and J. Chase. Human-robot collaboration: A literature review and augmented reality approach in design. *Int. J. Advanced Robotic Systems*, 5(1):1, 2008.

[31] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proc. 1996 IEEE Int. Conf. Evolutionary Computation*, pages 312–317, 1996.

[32] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[33] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *Proc. 2nd Int. Symp. Intelligent Robotic Systems*, pages 198–204, 1994.

[34] V. Heidrich-Meisner and C. Igel. Evolution strategies for direct policy search. In *Proc. 10th Int. Conf. Parallel Problem Solving from Nature*, pages 428–437, 2008.

[35] H. Hoffmann, P. Pastor, D. Park, and S. Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In *Proc. 2009 IEEE Int. Conf. Robotics and Automation*, pages 2587–2592, 2009.

[36] A. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21:642–653, 2008.

[37] A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. 2002 IEEE Int. Conf. Robotics and Automation*, volume 2, pages 1398–1403, 2002.

[38] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Adv. Neural Information Processing Systems 16*, pages 1547–1554, 2003.

[39] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[40] M. Ito and J. Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12(2): 93–115, 2004.

[41] N. Jacobstein. NASA's Perseverance: Robot laboratory on Mars. *Science Robotics*, 6(52), 2021.

[42] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *J. Artificial Intelligence Research*, 4:237–285, 1996.

[43] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *Proc. 10th IEEE Int. Conf. Computer Vision*, pages 166–173, 2005.

[44] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *Proc. 11th IEEE Int. Conf. Computer Vision*, pages 1–8, 2007.

[45] J. Kelso. *Dynamic Patterns: The Self-Organization of Brain and Behavior*. MIT Press, 1995.

[46] S. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32(4):433–454, 2012.

[47] S. Khansari-Zadeh and A. Billard. Real-time avoidance of fast moving objects A dynamical system-based approach. In *Proc. IROS Workshop on Robot Motion Planning Online, Reactive, and in Real-Time*, 2012.

[48] M. Kherallah, L. Haddad, A. Alimi, and A. Mitiche. On-line handwritten digit recognition based on trajectory and velocity modeling. *Pattern Recognition Letters*, 29(5):580–594, 2008.

[49] J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84:171–203, 2011.

[50] J. Kober, K. Mülling, O. Krömer, C. Lampert, B. Schölkopf, and J. Peters. Movement templates for learning of hitting and batting. In *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, pages 853–858, 2010.

[51] A. Kolbeinsson, E. Lagerstedt, and J. Lindblom. Foundation for a classification of collaboration levels for human-robot cooperation in manufacturing. *Production & Manufacturing Research*, 7(1):448–471, 2019.

[52] A. Kordia and F. Melo. An end-to-end approach for learning and generating complex robot motions from demonstration. In *Proc. 16th Int. Conf. Control, Automation, Robotics and Vision*, pages 1008–1014, 2020.

[53] A. Kordia and F. Melo. Compound movement recognition using dynamic movement primitives. In *Proc. 20th EPIA Conf. Artificial Intelligence*, pages 456–468, 2021.

[54] A. Kordia and F. Melo. Movement recognition and prediction using DMPs. In *Proc. 2021 IEEE Int. Conf. Robotics and Automation*, 2021.

[55] P. Kormushev, S. Calinon, and D. Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *Proc. 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3232–3237, 2010.

[56] P. Kormushev, S. Calinon, and D. Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *Proc. 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3232–3237, 2010.

[57] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Wörgötter. Modified dynamic movement primitives for joining movement sequences. In *Proc. 2011 IEEE Int. Conf. Robotics and Automation*, pages 2275–2280, 2011.

[58] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Trans. Robotics*, 28(1):145–157, 2012.

[59] Y. Kume, Y. Hirata, and K. Kosuge. Coordinated motion control of multiple mobile manipulators handling a single object without using force/torque sensors. In *Proc. 2007 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 4077–4082, 2007.

[60] D. Lee, C. Ott, and Y. Nakamura. Mimetic communication model with compliant physical contact in human-humanoid interaction. *Int. J. Robotics Research*, 29 (13):1684–1704, 2010.

[61] S. Liemhetcharat and M. Veloso. Modeling and learning synergy for team formation with heterogeneous agents. In *Proc. 11th Int. Conf. Autonomous Agents and Multiagent Systems*, pages 365–374, 2012.

[62] B. Lim, S. Ra, and F. Park. Movement primitives, principal component analysis, and the efficient generation of natural motions. In *Proc. 2005 IEEE Int. Conf. Robotics and Automation*, pages 4630–4635, 2005.

[63] H. Liu and L. Wang. Gesture recognition for human-robot collaboration: A review. *Int. J. Industrial Ergonomics*, 68:355–367, 2018.

[64] T. Liu, Y. Lei, L. Han, W. Xu, and H. Zou. Coordinated resolved motion control of dual-arm manipulators with closed chain. *Int. J. Advanced Robotic Systems*, 13(3), 2016.

[65] W. Lohmiller and J. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[66] M. Lopes, F. Melo, L. Montesano, and J. Santos-Victor. Abstraction levels for robotic imitation: Overview and computational approaches. In O. Sigaud and J. Peters, editors, *From Motor Learning to Interaction Learning in Robots*, pages 313–355. Springer, 2010.

[67] I. Loshchilov. A computationally efficient limited memory CMA-ES for large scale optimization. In *Proc. 2014 Annual Conf. Genetic and Evolutionary Computation*, pages 397–404, 2014.

[68] R. Luo, R. Hayne, and D. Berenson. Unsupervised early prediction of human reaching for human-robot collaboration in shared workspaces. *Autonomous Robots*, 42(3):631–648, 2018.

[69] F. Lv and R. Nevatia. Recognition and segmentation of 3-D human action using HMM and multi-class AdaBoost. In *Proc. European Conf. Computer Vision*, pages 234–249, 2006.

[70] S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *Proc. IEEE Int. Conf. Computer Vision*, pages 2744–2751, 2013.

[71] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters. Probabilistic movement primitives for coordination of multiple human-robot collaborative tasks. *Autonomous Robots*, 41(3):593–612, 2017.

[72] G. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, and J. Peters. A probabilistic framework for semi-autonomous robots based on interaction primitives with phase estimation. In *Robotics Research*, pages 253–268. Springer, 2018.

[73] S. Maeso, M. Reza, J. Mayol, J. Blasco, M. Guerra, E. Andradas, and M. Plana. Efficacy of the Da Vinci surgical system in abdominal surgery compared with that of laparoscopy: A systematic review and meta-analysis. *Annals of Surgery*, 252(2):254–262, 2010.

[74] J. Mainprice and D. Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Proc. 2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 299–306, 2013.

[75] J. Mainprice, R. Hayne, and D. Berenson. Goal set inverse optimal control and iterative re-planning for predicting human reaching motions in shared workspaces. *IEEE Trans. Robotics*, 32(4):897–908, 2016.

[76] G. Maistros, Y. Marom, and G. Hayes. Perception-action coupling via imitation and attention. In *Proc. AAAI Fall Symp. Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, 2001.

[77] S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robotics and Autonomous Systems*, 74:97–107, 2015.

[78] E. Marder and D. Bucher. Central pattern generators and the control of rhythmic movements. *Current Biology*, 11:R986–R996, 2001.

[79] C. Marques, J.C̃ristov ao, P. Lima, J. Frazão, M.I. Ribeiro, and R. Ventura. RAPOSA: Semi-autonomous robot for rescue operations. In *Proc. 2006 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3988–3993, 2006.

[80] D. Marr and L. Vaina. Representation and recognition of the movements of shapes. *Philosophical Transactions of the Royal Society of London B*, 214:501–524, 1982.

[81] M. Matarić. Sensory-motor primitives as a basis for learning by imitation Linking perception to action and biology to robotics. In *Imitation in Animals and Artifacts*, 2002.

[82] F. Meier, E. Theodorou, F. Stulp, and S. Schaal. Movement segmentation using a primitive library. In *Proc. 2011 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3407–3412, 2011.

[83] F. Meier, E. Theodorou, and S. Schaal. Movement segmentation and recognition for imitation learning. In *Proc. 15th Int. Conf. Artificial Intelligence and Statistics*, pages 761–769, 2012.

[84] K. Merckaert, B. Convens, C. Wu, A. Roncone, M. Nicotra, and B. Vanderborght. Real-time motion control of robotic manipulators for safe human-robot coexistence. *Robotics and Computer-Integrated Manufacturing*, 73:102223, 2022.

[85] X. Miao, H. Lee, and B. Kang. Multi-cleaning robots using cleaning distribution method based on map decomposition in large environments. *IEEE Access*, 8: 97873–97889, 2020.

[86] J. Miura, K. Iwase, and Y. Shirai. Interactive teaching of a mobile robot. In *Proc. 2005 IEEE Int. Conf. Robotics and Automation*, pages 3378–3383, 2005.

[87] P. Morasso, V. Mohan, G. Metta, and G. Sandini. Motion planning and bimanual coordination in humanoid robots. *Frontiers in Artificial Intelligence and Applications*, 196:169–185, 2009.

[88] L. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *Proc. 2007 IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[89] K. Mulling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. In *Proc. 2010 IEEE-RAS Int. Conf. Humanoid Robots*, pages 411–416, 2010.

[90] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, pages 237–242, 1991.

[91] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79–91, 2004.

[92] B. Nemec, M. Tamosiunaite, F. Woergoetter, and A. Ude. Task adaptation through exploration and action sequencing. In *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, pages 610–616, 2009.

[93] B. Nemec, N. Likar, A. Gams, and A. Ude. Bimanual human robot cooperation with adaptive stiffness control. In *Proc. 16th IEEE-RAS Int. Conf. Humanoid Robots*, pages 607–613, 2016.

[94] S. Niekum, S. Osentoski, G. Konidaris, and A. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 5239–5246, 2012.

[95] O. Ogorodnikova. Human weaknesses and strengths in collaboration with robots. *Periodica Polytechnica Mechanical Engineering*, 52(1):25–33, 2008.

[96] N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[97] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. *Adv. Neural Information Processing Systems 26*, pages 2616–2624, 2013.

[98] D. Pardo and C. Angulo. Collaborative control in a humanoid dynamic task. In *Proc. 4th Int. Conf. Informatics in Control, Automation and Robotics—Robotics and Automation*, pages 174–180, 2007.

[99] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Proc. 2009 IEEE Int. Conf. Robotics and Automation*, pages 763–768, 2009.

[100] J. Pepito and R. Locsin. Can nurses remain relevant in a technologically advanced future? *Int. J. Nursing Sciences*, 6(1):106–110, 2019.

[101] C. Pérez-D'Arpino and J. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In *Proc. 2015 IEEE Int. Conf. Robotics and Automation*, pages 6175–6182, 2015.

[102] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–97, 2008.

[103] J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *Proc. 16th Eur. Conf. Machine Learning*, pages 280–291, 2005.

[104] M. Pittelkau. Adaptive load-sharing force control for two-arm manipulators. In *Proc. 1988 IEEE Int. Conf. Robotics and Automation*, pages 498–503, 1988.

[105] M. Prada, A. Remazeilles, A. Koene, and S. Endo. Dynamic movement primitives for human-robot interaction: Comparison with human behavioral observation. In *Proc. 2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 1168–1175, 2013.

[106] R. Prada, P. Lopes, J. Catarino, J. Quitério, and F. Melo. The geometry friends game AI competition. In *Proc. 2015 IEEE Conf. Computational Intelligence and Games*, pages 431–438, 2015.

[107] F. Pratticò and F. Lamberti. Mixed-reality robotic games: Design guidelines for effective entertainment with consumer robots. *IEEE Consumer Electronics Magazine*, 10(1):6–16, 2021.

[108] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[109] R. Rao, A. Shon, and A. Meltzoff. A Bayesian model of imitation in infants and robots. In *Imitation and Social Learning in Robots, Humans, and Animals*. Cambridge University Press, 2007.

[110] L. Roveda, J. Maskani, P. Franceschi, A. Abdi, F. Braghin, L. Tosatti, and N. Pedrocchi. Model-based reinforcement learning variable impedance control for human-robot collaboration. *J. Intelligent & Robotic Systems*, 100(2):417–433, 2020.

[111] J. Rubin and W. Richards. Boundaries of visual motion. Technical report, Massachusetts Institute of Technology, 1985.

[112] T. Rückstiess, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn J. Behavioral Robotics*, 1:14–24, 2010.

[113] R. Samant, L. Behera, and G. Pandey. Adaptive learning of dynamic movement primitives through demonstration. In *Proc. 2016 Int. Joint Conf. Neural Networks*, pages 1068–1075, 2016.

[114] M. Saveriano, F. Abu-Dakka, A. Kramberger, and L. Peternel. Dynamic movement primitives in robotics: A tutorial survey. *CoRR*, abs/2102.03861, 2021.

[115] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[116] S. Schaal, C. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[117] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *IROS Workshop on Bilateral Paradigms on Humans and Humanoids*, pages 1–21, 2003.

[118] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Proc. 11th Int. Symp. Robotics Research*, pages 561–572, 2005.

[119] F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[120] E. Shahriari, A. Kramberger, A. Gams, A. Ude, and S. Haddadin. Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement

primitives. In *Proc. 17th IEEE/RAS Int. Conf. Humanoid Robotics*, pages 136–142, 2017.

[121] S. Stavridis and Z. Doulgeri. Bimanual assembly of two parts with relative motion generation and task related optimization. In *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 7131–7136, 2018.

[122] F. Stulp and O. Sigaud. Path integral policy improvement with covariance matrix adaptation. In *Proc. 29th Int. Conf. Machine Learning*, pages 1547–1554, 2012.

[123] F. Stulp and O. Sigaud. Robot skill learning: From reinforcement learning to evolution strategies. *Paladyn J. Behavioral Robotics*, 4(1):49–61, 2013.

[124] Freek Stulp and Olivier Sigaud. Policy improvement methods: Between black-box optimization and episodic reinforcement learning. 2012.

[125] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient natural evolution strategies. In *Proc. 11th Annual Conf. Genetic and Evolutionary Computation*, pages 539–546, 2009.

[126] Markku Suomalainen, Yiannis Karayiannidis, and Ville Kyrki. A survey of robot manipulation in contact. *Robotics and Autonomous Systems*, 156:104224, 2022.

[127] U. Tan, O. Rabaste, C. Adnet, and J. Ovarlez. On the eclipsing phenomenon with phase codes. In *Proc. 2019 IEEE Int. Conf. Radar*, pages 1–5, 2019.

[128] Y. Tanaka, J. Kinugawa, Y. Sugahara, and K. Kosuge. Motion planning with worker's trajectory prediction for assembly task partner robot. In *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 1525–1532, 2012.

[129] P. Tavares, J. Lima, P. Costa, and A. Moreira. Multiple manipulators path planning using double A*. *Industrial Robot*, 43(6):657–664, 2016.

[130] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *J. Machine Learning Research*, 11:3137–3181, 2010.

[131] V. Villani, F. Pini, F. Leali, and C. Secchi. Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018.

[132] Y. Wang, M. Huber, V. Papudesi, and D. Cook. User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 424–429, 2003.

[133] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

[134] H. Yoon, J. Soh, Y. Bae, and H. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501, 2001.

[135] H. Zadeh, M. Menhaj, and H. Talebi. Decentralized force and motion control of multiple cooperative manipulators. *Automatika*, 62(1):98–108, 2021.

[136] Peng Zhang and Junxia Zhang. Motion generation for walking exoskeleton robot using multiple dynamic movement primitives sequences combined with reinforcement learning. *Robotica*, 40(8):2732–2747, 2022.