Biologically inspired motor skill learning in robotics through probabilistic inference

by

Elmar Rückert

DISSERTATION

to obtain the title of

Doctor of Technical Sciences

Graz University of Technology

Institute for Theoretical Computer Science

Thesis Advisor: Univ.-Prof. DI Dr. Wolfgang MAASS
defended on February 4, 2014

Jury:

Advisor:	Univ.-Prof. DI Dr. Wolfgang MAASS	- TU Graz
Reviewer:	Univ.-Prof. Dr. Jan PETERS	- TU Darmstadt
Dean of Studies:	Assc.-Prof. DI Dr. Denis HELIC	- TU Graz

Abstract

Modern humanoid robots are impressive complex machines with many degrees-of-freedom and sensors. Some of these robots are equipped with compliant joints that smoothly react to external forces. This feature allows for a natural interaction between humans and the robot, and one hopes that they also facilitate learning of motor skills through compensating disturbances. Classical engineering approaches that try to exactly model the compliant joints, the environment, or the possible movements are likely to fail.

Inspired by recent findings in biological motor control, I have investigated alternative approaches based on probabilistic inference for representing and learning complex motor skills in continuous action spaces. In particular, I have shown how movement plans can be generated in the probabilistic inference framework, how complex movements can be composed of simpler elementary movement primitives, how such movement representations can be learned using reinforcement learning, and which salient features of biological motor control can be reproduced in this manner.

More generally, these studies are first steps towards an autonomous learning system that encapsulates machine learning disciplines like “transfer learning”, “reinforcement learning”, “structure learning”, and “imitation learning” in the context of robotics through the framework of probabilistic inference. Such a probabilistic inference approach has the potential to close the gap between learning abilities of current robots and biological learning systems in the near future.

Keywords. Motor skill learning, probabilistic inference, biological features of motor control, reinforcement learning, policy search, movement primitives

Kurzfassung

Moderne humanoide Roboter sind eindrucksvoll komplexe Maschinen die über eine Vielzahl von Freiheitsgraden und Sensoren verfügen. Einige dieser Roboter verfügen über nachgiebige Gelenke, die durch Einwurkung externe Kräfte flexibel werden. Diese Eigenschaft ermöglicht eine natürlichere Interaktion zwischen Mensch und Machine und man erhofft sich, dass das Lernen von Bewegungen durch eine automatische Kompensation von Störungen erleichtert wird. Klassische maschinelle Lernmethoden, die versuchen ein möglichst genaues Model der Gelenke, der Umgebung und der möglichen Bewegungsabläufe zu erstellen scheinen hier an ihre Grenzen zu stoßen.

Inspiriert durch neue Erkenntnisse biologischer Bewegungsforschung bin ich in dieser Arbeit alternative Wege gegangen und habe probabilistische Inferenzmethoden zum Beschreiben und zum Lernen komplexer Bewegungen in hochdimensionalen kontinuierlichen Räumen untersucht. Im Speziellen habe ich gezeigt wie Bewegungspläne durch probabilistische Inferenz generiert werden können, wie sich komplexe Bewegungsabläufe aus einfacheren elementaren Bewegungen bilden lassen, wie solche Repräsentationen mit "Reinforcement learning" gelernt werden könnnen und welche charakteristischen Eigenschaften biologischer Bewegungskontrolle in diesem Zusammenhang modelliert werden können.

Allgemeiner betrachtet sind diese Arbeiten erste Schritte in Richtung eines autonomen Lernsystems das maschinelle Lerndisziplinien wie "transfer learning", "reinforcement learning", "structure learning" und "imitation learning" in der Robotik durch probabilistische Inferenz miteinander in Verbindung setzen kann. Solche inferenz Methoden haben das Potential die Kluft zwischen den aktuellen Fähigkeiten moderner Roboter und biologischer Lernsysteme in naher Zukunft zu schließen.

Acknowledgments

I would like to thank my advisor Wolfgang Maass for his guidance, for the opportunity in participating in exciting research, and above all for his inspiring ideas and visions. Special thanks go to Jan Peters for taking time out from his very busy schedule to review my thesis and to attend my defense.

I am also very grateful for the enjoyable and fruitful collaborations with my co-authors and colleagues Gerhard Neumann, Marc Toussaint and Andrea d'Avella. It was a pleasure to work with you.

My deepest thanks go to to my current and former colleagues during my time at the Institute for Theoretical Computer Science, especially to Gerhard Neumann, David Kappel, Dejan Pecevski, Oliver Friedl, Potzinger Daniela and Regina Heidinger.

I also want to acknowledge the financial support from Graz University of Technology and the research program of the European Union, including the AMARSi project. My gratitude also goes to my research colleagues in the AMARSi consortium, Juan Pablo Carbajal, Enrico Chiovetto, Marting Giese, Yuri Ivanenko, William Land, Albert Mukovskiy, Thomas Schack, and Benjamin Schrauwen for the nice and inspiring discussions.

Last but not least, I would like to thank my family and friends, in particular my parents Gerhard and Karoline, for their endless patience and support. Special thanks also go the my brothers and sisters, Claudia, Genoveva, Katharina, and Paul for their encouragement and friendship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Motor skill learning in autonomous robots with many degrees-of-freedoms and compliant actuators is challenging. Classical movement planning approaches typically fail in such systems due to the high-dimensional control signals and the stochasticity of the executed motor commands. Despite these challenges, autonomous robots have to cope with complex problems like changing environments, learning from sparse teacher signals, complex, but often structured task settings, and noisy demonstrations or observations. For these challenges in robotics specialized learning theories have been developed, such as transfer learning, reinforcement learning, structure learning, and imitation learning. Probabilistic inference is a promising approach to implement motor skill learning in high-dimensional autonomous systems that links these learning theories. As a first step toward an integrative learning system, I have applied this new perspective on probabilistic information processing to implement several biologically inspired motor skill learning approaches.

1.1 An introduction to probabilistic inference for planning

Probabilistic inference provides a mathematical theory for anticipatory behavior. To illustrate the key idea behind the approach, we consider a simple movement planning example.

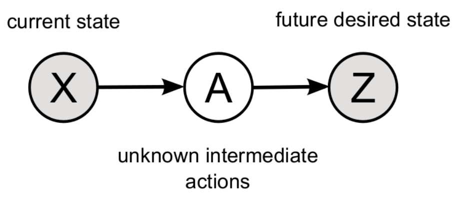


Figure 1.1: Conceptual idea of probabilistic inference for planning. Assume we know our current state X (“where we are”) and our future desired state Z (“where we want to be”). We can now determine the probabilities of intermediate actions A.

Assume we know our current state X (“where we are”) and our future desired state Z (“where we want to be”). We can now calculate the probabilities of intermediate actions A, which are necessary to reach the desired state Z from the current state X by “observing” of (or conditioning on) X and Z. Thus, the desired future Z is seen as “mental observation”.

The capital letters X, A, Z denote random variables (RVs). Lower case letters denote values, where e.g., $x = \mathrm{dom}(X)$ represents a particular value x in the domain X. Its probability is denoted by $p(X = x)$, where typically $p(x)$ is used as shorthand.

Using these definitions we can give a mathematical description of the simple planning example. In general this two part process involves *modelling* the interaction between random variables and *inference*. For the modelling, we define a joint distribution over all involved RVs, $p(x, a, z) = p(x)p(a|x)p(z|a)$. These dependencies can be visualized in the graphical model in Figure 1.1. RVs are represented by nodes, where shaded circles denote knowns and white circles unknowns. The links between these nodes encode the dependencies between RVs, i.e., the conditional distributions.

All questions of interest are answered by performing inference in this graphical model, e.g., the probability of a particular action a is $p(a|x, z) = 1/N\, p(x)\, p(a|x)\, p(z|a)$. The normalization constant $N = \sum_{\tilde{a} \in A} p(x)p(\tilde{a}|x)p(z|\tilde{a})$ denotes the marginal distribution over all actions. Note that for this result only the rule for conditional probabilities $p(X|Y) = p(X, Y)/P(Y)$ needed to be applied.

This simple example can be easily extended to more complex multi-step planning problems in high-dimensional non-linear systems. For instance, the action a is replaced by a sequence of states $\langle \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T \rangle$ of length T. Such a sequence of states and the corresponding control commands in non-linear dynamical systems are inferred in Chapter 2 and in Chapter 3.

The advantage of probabilistic inference for planning is that there are no distinctions between sensor and motor, perception and action. We can include a multitude of random variables, some of which might represent features of the state, some might represent goals or constraints in the future. These additional random variables can also be used to model prior knowledge, which facilitates the learning of motor skills and can be exploited in transfer learning, i.e., the acquisition of new skills based on previously learned priors (see Chapter 3).

For solving the inference problem many sophisticated inference techniques can be used, e.g., message passing, variational inference methods or Markov Chain Monte Carlo (MCMC) sampling approaches. In Chapter 2 and Chapter 3, message passing with Gaussians is used as inference method. In a recent study on movement planning through networks of spiking neurons probabilistic inference is implemented using a MCMC sampling approach. This study is currently in progress of writing and not presented in this thesis.

1.2 Motor skill learning with movement primitives

A common strategy to control high-dimensional complex robots is to decompose complex movements into simpler elementary movement primitives (MPs). These MPs are parametrized functions that encode movement trajectories or control signals. The key principle of MPs is that complex time-varying trajectories can be generated using a trajectory generator based on few parameters (see Section 3.2.1 for a discussion). In this thesis, two new types of MPs are proposed that comply with salient features of biological motor control. In Chapter 3, the probabilistic inference for planning framework is used to formulate movement primitives. This movement representation models salient features of human motor control such as motor variability and learning efficiency. In Chapter 4, a biologically inspired movement primitive model is proposed that is based on muscle synergies, i.e., the coherent activations of groups of muscles. This movement primitive framework is used for learning multiple related tasks exploiting learned shared task knowledge.

During a learning phase the parameters of such MPs are adapted to achieve a particular goal (or even multiple goals). In this thesis, I have focused on the reinforcement learning paradigm (in contrast to imitation learning or kinesthetic teaching), where a typically sparse teacher signal is used to improve the current parametrization of a movement primitive. These parameters can be interpreted as policy — a strategy to solve a task. For learning or optimizing the policy parameters a variety of policy search algorithms exist in the motor control literature (see Section 3.2.2 or Section 4.2.4 for a discussion).

In this thesis, a stochastic search method is used for policy search and the main focus of the presented motor skill learning approaches is on modelling salient features of biological motor control.

1.3 Salient features of biological motor skill learning

A good understanding of salient features of biological motor control can facilitate the development of new and powerful motor skill learning approaches in robotics. In this thesis, the following features are modeled in particular:

- **Biological organisms developed specialized morphologies throughout evolution.** In morphological computation one follows a similar goal, where different robot designs are evaluated to improve the learning speed, the flexibility, or the robustness of the hardware. This time-consuming process could be advanced by comparing simulated robots with different morphologies. However, the learned control strategy and the morphology interact and cannot be investigated in isolation. In Chapter 2, I have demonstrated how different morphologies of simulated robots can be compared by always applying the optimal control strategy, which is generated using probabilistic inference for planning.

- **Complex movements are composed of simpler elementary movements.** The musculoskeletal apparatus of a biological motor system is a high-dimensional redundant stochastic system and has many more degrees-of-freedom than needed to perform a specific action. A classical hypothesis is that such redundancy is resolved by a combination of only a small number of functional units, namely movement primitives. In Chapter 3, a novel movement primitive representation based on probabilistic inference in learned graphical models is presented.

- **If humans perform the same task several times, the resulting movement trajectories vary considerably.** Probabilistic inference is an excellent computational theory of this effect named motor variability. If the task constraints (which could for example be the requirement to reach a target) are not violated, suppressing the inherent noise in the stochastic system will lead to inefficiencies. This effect is modeled in Chapter 3.

- **Transfer of knowledge to new situations facilitates motor skill learning.** A salient feature of human motor skill learning is the ability to exploit similarities across related tasks. In Chapter 4, I have shown that these similarities can be represented by learned basis functions or muscle synergies. Multiple movement tasks can be solved by applying individual combinations of the shared basis functions.

In a recent study on an implementation of probabilistic planning through networks of spiking neurons two additional interesting features of biological motor control were modeled. They are only briefly discussed here as an outlook on ongoing work that is not essential for this thesis. First, such neural networks can encode multiple movement trajectories that compete, and the agent (in this study a simulated rodent) can switch between multiple strategies. This strategy switching behavior is also observed when a *real* rodent has learned that multiple paths (with equal length) in a maze lead to the same reward [Pfeiffer and Foster, 2013]. Second, these trajectories can be modulated through reinforcement learning, where in particular a linear ramping reward signal is used for learning to navigate in mazes with obstacles. Similar ramping dopamine signals were recently observed in rodents [Howe et al., 2013], which cannot be explained by traditional temporal difference reinforcement learning methods. However, the proposed spiking neural network model that implements probabilistic planning provides a functional role for the suprising finding of ramping dopamine signals in rodents.

1.4 Organization of the thesis

The framework of probabilistic inference for planning is first used in Chapter 2 to investigate the power of morphological computation. In this thesis, multi-step movement planning problems are studied with simulated humanoid robots with different morphologies.

In Chapter 3, this approach is extended, where the graphical model for probabilistic planning is learned from data and not assumed to be known. This reproduces these salient features of biological motor control: its modular organization in movement primitives, its characteristics of stochastic optimality under perturbations, and its learning efficiency.

Another salient feature of human motor skill learning is the ability to exploit similarities across related tasks. Such similarities can be interpreted as structural knowledge, which facilitates the learning of related tasks. A modelling approach of this feature using shared parametrized basis functions (muscle synergies) is discussed in Chapter 4.

Chapter 2

A study of morphological computation using probabilistic inference

Contents

A key idea behind morphological computation is that many difficulties of a control problem can be absorbed by the morphology of a robot. The performance of the controlled system naturally depends on the control architecture and on the morphology of the robot. Due to this strong coupling, most impressive applications in morphological computation typically apply minimalistic control architectures. Ideally, adapting the morphology of the plant and optimizing the control law interact such that finally, optimal physical properties of the system and optimal control laws emerge. As a first step towards this vision we apply optimal control methods for investigating the power of morphological computation. We use a probabilistic optimal control method to acquire control laws given the current morphology. We show that by changing the morphology of our robot, control problems can be simplified, resulting in optimal controllers with reduced complexity and higher performance. This concept is evaluated on a compliant 4-link model of a humanoid robot during balance control in the presence of external pushes.

2.1 Introduction

The control of compliant robots is inherently difficult due to their nonlinear stochastic dynamics. Morphological computation poses the vision that parts of the complexity of a control or learning task can be outsourced to the morphology of the robot [Pfeifer and Bongard, 2006, Pfeifer et al., 2007]. This computational power of the morphology has been illustrated in many impressive biologically inspired robotic applications. For example, Tedrake et al. [2005] showed that the complexity of learning to walk is drastically reduced by exploiting the dynamics of a passive biped walker. Iida and Pfeifer [2006] demonstrated that the stability of a simple four-legged robot exhibits a surprisingly robust behavior using a mixture of active and passive joints. For swimming [Ziegler et al., 2006] or flying [Wood, 2007] investigations of robot morphologies yield rich behavioral diversity using only a single actuator exploiting the dynamics of the environment.

These approaches typically used minimalistic control architectures, like open loop actuation with sinusoidal drives [Iida and Pfeifer, 2006, Ziegler et al., 2006, Wood, 2007] or a simple linear combination of state dependent features [Tedrake et al., 2005]. The morphology is assumed to absorb much of the computation needed for controlling the robot, and, therefore, less emphasis is placed on the controller. More sophisticated control architectures were omitted as these are typically more difficult to fine-tune for a new morphology. Thus, finding a good control law defines an individual optimization problem for each morphology, which renders the joint optimization of the morphology and the controller challenging. However, an autonomous system has to encompass for both aspects — the morphology and the control architecture — for efficient learning of complex motor skills.

One remarkable exception to the typically used minimalistic control architectures is the theoretical study of the morphological contribution for compliant bodies in Hauser et al. [2011]. The authors have demonstrated that by outsourcing the computation to the physical body, the difficult problem of learning to control a complex body, could be reduced to a simple and perspicuous linear learning task. This concept was tested in simulations for simple but generic nonlinear models of physical bodies that are based on mass-spring systems. The linear learning task less likely get stuck in local minima of an error function. However, learning the optimal morphology and the optimal control law remains an open problem for many motor control tasks using real robots.

As a first step towards fulfilling this vision, we propose to directly use optimal control methods for eliminating the dependency between the morphology of the robot and the control architecture. We show that for balancing control tasks with simple humanoid robot models this dependency can be truly eliminated and the effect of different morphologies can quantified. Thus, for evaluating the computational power of the morphology, we will always use the optimal control law connected to this morphology. For more complex tasks or robot models the optimal control law needs to be learned, which is not discussed in this manuscript.

For determining our optimal control law for a given morphology, we can use one of the many tools provided by Stochastic Optimal Control (SOC) methods such as Bertsekas and Shreve [1978], von Stryk and Bulirsch [1992], Bertsekas et al. [1995], Todorov and Li [2005], Kappen [2007], Toussaint [2009]. These methods have been shown to be powerful approaches for movement planning in high-dimensional robotic systems. We will use Approximate Inference Control (AICO) [Toussaint, 2009], as it is a state of the art planning method for stochastic optimal control tasks. AICO is based on probabilistic inference for motor control. The beauty of this approach is that there is no distinction between sensor and motor, perception and action. We can include a multitude of variables, some of which might represent some features of the state, some of which might represent goals, constraints or motivations in the future and some of which might represent future actions or motor signals.

As all other stochastic optimal control methods, AICO minimizes a cost function, which is in our case given by the quadratic distance to a target state and the used energy of the movement. In order to apply the AICO method to torque constraint dynamical models we will briefly explain how to extend the algorithm to systems with control limits in Section 2.2.

Quantifying how much computation is done by the morphology of the robot and how much is done by the controller is often a difficult problem. Despite of the advancements for theoretical models of morphological computation [Hauser et al., 2011], where the computational power of the plant can be assessed in a principled manner, this remains an open problem for many motor control tasks using real robots. Yet, by the use of optimal control laws the computation done by the morphology can be quantified by investigating the complexity of the controller. Thus, if the controller needs to do a lot of computation less computation is provided by the morphology in order to fulfill a task.

AICO also provides us with a time-varying linear feedback controller as policy to generate movement trajectories. We will use the variance of the control gains as complexity measure of the controller. If the control gains are almost constant in time, the control law is close to linear and does not perform much computation. However, if the control gains are highly varying, the controller needs to do a lot of computation and therefore, less computation is provided by the morphology.

As different complexity measures are possible we additionally use the final costs and the total jerk of a movement trajectory for a comparison. We illustrate the power of morphological computation combined with optimal control on a dynamical model of a humanoid robot (70kg, 2m). The robot is modelled by a 4-link pendulum, which has to keep balance in the presence of external pushes. We will show in Section 2.3 that by changing the morphology of a robot like the joint friction, the link lengths or the spring constants, the resulting optimal controllers have reduced complexity. As we will demonstrate there are optimal values for the physical properties for a given control task, which can only be found by the use of optimal control laws. With naive control architectures, different, sub-optimal morphologies would be chosen.

2.1.1 Related work

We propose to use optimal control methods to eliminate the dependency between the control architecture and the morphology of the robot. These stochastic optimal control (SOC) methods such as Todorov and Li [2005], Kappen [2007], Toussaint [2009] have been shown to be powerful methods for controlling high-dimensional robotic systems. For example the incremental Linear Quadratic Gaussian (iLQG) [Todorov and Li, 2005] algorithm is one of the most commonly used SOC methods. It uses Taylor expansions of the system dynamics and cost function to convert the nonlinear control problem in a Linear dynamics, Quadratic costs and Gaussian noise system (LQG). The algorithm is iterative - the Taylor expansions are recalculated at the newly estimated optimal trajectory for the LQG system.

A SOC problem can be reformulated as inference problem in a graphical model [Toussaint, 2009, H.J. et al., 2012], which has the nice property that there is no distinction between sensor and motor, perception and action. The unknown quantities, i.e., the states and actions can be efficiently inferred using for example the Approximate Inference Control (AICO) algorithm [Toussaint, 2009]. The graphical model is given by a simple dynamic Bayesian network with states $\mathbf{x}_t$, actions $\mathbf{u}_t$ and task variables $\mathbf{z}_t$ (representing the costs) as nodes, see Figure 2.1. In this graphical model observations are denoted by shaded nodes in contrast to the unknown random variables, which are indicated by circles. If beliefs in the graphical model are approximated as Gaussian the resulting algorithm is very similar to iLQG. Gaussian message passing iteratively re-approximates local costs and transitions as LQG around the current mode of the belief within a time slice. A difference to iLQG is that AICO uses forward messages instead of a forward roll-out to determine the point of a local LQG approximation. Thus, AICO allows to iterate belief re-approximation within a time slice until convergence. This may lead to faster overall convergence [Toussaint, 2009].

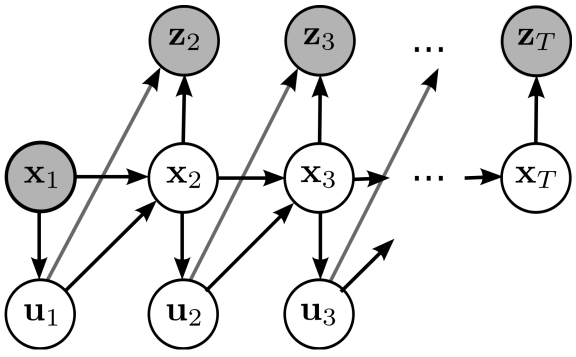


Figure 2.1: This figure illustrates the graphical model for probabilistic planning, where we consider finite horizon tasks with T time steps. The state variable $\mathbf{x}_t$ denotes for example the joint angles and joint velocities of a robot. Controls are labelled by $\mathbf{u}_t$. The beauty of probabilistic inference for motor control is that we can include a multitude of variables, some of which might represent some features of the state, some of which might represent goals, constraints or motivations in the future and some of which might represent future actions or motor signals. These constraints are expressed in the model by the task variables $\mathbf{z}_t$.

The dynamic Bayesian network shown in Figure 2.1 is fully specified by the conditional distributions encoded by the cost function and by the state transition model. Like most SOC methods [Todorov and Li, 2005, Kappen, 2007, Toussaint, 2009] we assume that the cost function and the state transition model are known. However, for model learning many types of function approximators can be applied [Vijayakumar et al., 2005, Nguyen-Tuong et al., 2008a,b]. For learning the cost function inverse reinforcement learning methods such as Abbeel and Ng [2004], Boularias et al. [2011] could be used. Since we demonstrate in this paper how morphological computation and optimal control can benefit from each other in general, an extension to the more complex learning problems remains for future research.

The original formulation of the AICO method [Toussaint, 2009] does not consider torque limits, which are important for many robotic experiments as well for the dynamic balancing experiments we consider in this paper. Therefore we extended the algorithm, which is discussed in the next section.

2.2 Probabilistic inference for motor planning

We use the probabilistic planning method Approximate Inference Control (AICO) [Toussaint, 2009] as optimal control method. Most applications of AICO are in the kinematic planning domain. Here, we want to apply AICO to fully dynamic, torque controlled robot simulations. Therefore we had to extend the AICO framework with control or torque limits, which is explained in the next subsections.

2.2.1 Approximate inference control

We will briefly clarify the notation for our discussion. Let $\mathbf{x}_t$ denote the state and $\mathbf{u}_t$ the control vector at time step t. A trajectory τ is defined as sequence of state control pairs, $\tau = \langle \mathbf{x}_{1:T}, \mathbf{u}_{1:T-1} \rangle$, where T is the length of the trajectory. Each trajectory has associated costs

$$L(\tau) := \sum_{t=1}^{T} c_t(\mathbf{x}_t, \mathbf{u}_t), \tag{2.1}$$

where $c_t(\mathbf{x}_t, \mathbf{u}_t)$ represents the cost function for a single time step, which is in our case given by the quadratic distance to a target state and the used energy of the movement. Solely the cost function $c_{1:T}(\cdot)$ and the initial state $\mathbf{x}_1$ are known to the optimal control algorithm. The unknown trajectory τ is the result of the inference process.

AICO uses message passing in graphical models to infer the trajectory τ. In order to transform the minimization of $L(\tau)$ into an inference problem, for each time step an individual binary random variable z_t is introduced. This random variable indicates a reward event. Its probability is given by

$$P(z_t = 1 | \mathbf{x}_t, \mathbf{u}_t, t) \propto \exp(-c_t(\mathbf{x}_t, \mathbf{u}_t)).$$

AICO now assumes that a reward event $z_t = 1$ is observed at every time step, see Figure 2.1. Given that evidence, AICO calculates the posterior distribution $P(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | z_{1:T} = 1)$ over trajectories.

We will use the simplest version of AICO, where an extended Kalman smoothing approach is used to estimate the posterior $P(\mathbf{x}_{1:T}, \mathbf{u}_{1:T} | z_{1:T} = 1)$. The extended Kalman smoothing approach uses Taylor expansions to linearize the system and subsequently uses

Gaussian messages for belief propagation in a graphical model. Gaussian message passing iteratively re-approximates local costs and transitions as a Linear dynamics, Quadratic costs and Gaussian noise system (LQG) around the current mode of the belief within a time slice. Concise derivations of the messages for AICO are given in the Appendix, as they are used to further extend the algorithm to implement control constraints.

AICO provides us with a linear feedback controller for each time slice of the form

$$\mathbf{u}_t = \mathbf{O}_t \mathbf{x}_t + \mathbf{o}_t,$$

where $\mathbf{O}_t$ is the inferred feedback control gain matrix and $\mathbf{o}_t$ denotes the linear feedback controller term. For our evaluations we also use a complexity measure which is proportional to the variance of this feedback control law:

$$\mathrm{Var}(\mathbf{O}_{1:T-1}, \mathbf{o}_{1:T-1}) = \sum_{i=1}^{D^x} \sum_{j=1}^{D^u} \mathrm{var}(\mathbf{O}_{i,j,1:T-1}) + \sum_{j=1}^{D^u} \mathrm{var}(\mathbf{o}_{j,1:T-1}), \tag{2.2}$$

where the dimension of the states is denoted by D^x and the dimension of the controls is denoted by D^u.

AICO is only a local optimization method and we have to provide an initial solution which is used for the first linearization. We will simply use the first state as initialization $\mathbf{x}_{2:T} = \mathbf{x}_1$.

If we use AICO with a constant cost and dynamic model for each time step, the algorithm reduces to calculating a Linear Quadratic Regulator (LQR), which is often used in optimal control. A LQR is the optimal linear feedback controller for a given linear system. In contrast, AICO uses a time-varying linear feedback controller, which may be different for each time step. In our experiments we will compare both approaches on a dynamic nonlinear balancing task. Thus we evaluate the benefit of using AICO (time-varying linear feedback control) and a LQR (constant linear feedback control).

2.2.2 Cost function for constrained systems

In order to apply the AICO algorithm to torque controlled robots, we have to extend the framework to incorporate control limits as the available motor torque is typically limited. This extension is done by adding a control dependent punishment term $c_t^u(\mathbf{u}_t)$ to the cost function in Equation 2.1. Thus for a trajectory τ we specify the costs

$$L(\tau) := \sum_{t=1}^{T} c_t(\mathbf{x}_t, \mathbf{u}_t) + c_t^u(\mathbf{u}_t).$$

We use this additional term to punish controls $\mathbf{u}_t$ that exceed a given bound $\mathbf{u}_{B_t}$ at time t:

$$c_t^u(\mathbf{u}_t) = (\mathbf{u}_t - \mathbf{u}_{B_t})^T \mathbf{H}_{B_t} (\mathbf{u}_t - \mathbf{u}_{B_t}).$$

As a consequence, the resulting Gaussian distributions which are used to represent the costs c_t change. This distributions have typically zero mean in the control space due to the typically used quadratic control costs. In the case of control limits the mean of the distribution is *non-zero*. Consequently, also the message passing update equations used for the AICO algorithm changes. The exact message passing equations of AICO with control limits are presented in the Appendix.

2.3 Experiments

We investigate morphological computation combined with optimal control on dynamic nonlinear balancing tasks [Atkeson and Stephens, 2007], where a robot gets pushed with a specific force and has to move such that it maintains balance. The optimal strategy is a nonlinear control law which returns the robot to the upright position. For our evaluations different initial postures and multiple forces are used as sketched in Figure 2.2. We study the morphological computation by changing different physical properties of a 4-link model of a humanoid robot. In particular we investigate the computation done by different friction coefficients and the link lengths. Inspired by the work of Hauser et al. [2011] we will also incorporate springs to our model for analyzing different spring constants.

2.3.1 Setting

We use a 4-link robot as a simplistic model of a humanoid (70kg, 2m) [Atkeson and Stephens, 2007]. The 8-dimensional state $\mathbf{x}_t$ is composed of the ankle, the knee, the hip and the arm positions and their velocities. Table B.1 in Appendix B.2 shows the initial velocities (resulting from the force F which always acts at the shoulder of the robot) and the valid joint angle range for the task. Additionally to the joint limits, the controls are limited to the intervals $[\pm 70, \pm 500, \pm 500, \pm 250]$Ns (ankle, knee, hip and arm). For more details we refer to Atkeson and Stephens [2007].

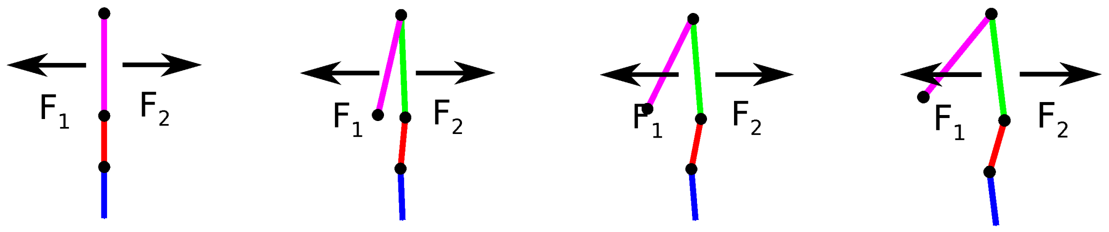


Figure 2.2: This figure illustrates different initial postures of a 4-link pendulum modelling a humanoid robot (70kg, 2m). The robot gets pushed with specific forces, i.e., F_1 and F_2 and has to move such that it maintains balance. At the end of the movement the robot should stabilize at the upright position.

We use a quadratic cost function given by

$$c_t(\mathbf{x}_t, \mathbf{u}_t) = (\mathbf{x}_t - \mathbf{x}^*)^T \hat{\mathbf{R}}_t (\mathbf{x}_t - \mathbf{x}^*) + \mathbf{u}_t^T \hat{\mathbf{H}}_t \mathbf{u}_t,$$

where the final state is denoted by $\mathbf{x}^* = \mathbf{0}$. The precision matrix $\hat{\mathbf{R}}_t$ determines how costly it is not to reach $\mathbf{x}^*$. The diagonal elements of $\hat{\mathbf{R}}_{1:T}$ are set to $2 \cdot 10^3$ for joint angles and to 0.2 for joint velocities. Controls are punished by $\hat{\mathbf{H}}_{1:T-1} = 0.05\mathbf{I}$.

The movement trajectories are simulated with a time-step of 0.5ms. For the AICO algorithm we use a planning time step of $\Delta t = 5$ms and a horizon of $T = 500$, which results in a movement duration of 2.5s. We use a torque dependent noise model — the controls are multiplied by Gaussian noise ϵ with mean 1 and a standard deviation of 0.25, i.e., we use 25% torque dependent noise. The noise affects the system dynamics $\dot{\mathbf{x}} = f(\mathbf{x}_t, \mathbf{u}_t + \epsilon)$ while simulating a trajectory, where $\epsilon \sim \mathcal{N}(\epsilon|0, 0.25^2 \cdot \text{abs}(\mathbf{u}_t))$. The experiments are performed for multiple forces sampled from $F \sim \mathcal{U}[-15, +15]$Ns, as shown in Figure 2.2.

2.3.2 Complexity measures

We evaluate the average values of the final costs $L(\tau)$ in Equation 2.1, the variance of the time-varying controller gains returned by the AICO approach in Equation 2.2 and the total jerk $J(\tau) = \Delta t \sum_{t=0}^{T} \dot{\mathbf{u}}_t^T \dot{\mathbf{u}}_t$ of the trajectory (proportional to the squared derivative of the torque). Different complexity measures would be possible. However, the chosen ones are plausible since the complexity of controlling the robot is reflected by how well the costs $L(\tau)$ are optimized. As the jerk of a movement tracks the derivative of the torques, this also seems to be a reliable complexity measure. Finally the variance of the time-varying controller gains quantifies the complexity of the control law in comparison to linear controllers (no variance).

2.3.3 Friction induces morphological computation

In this experiment we evaluate the influence of the friction coefficient γ on the quality of the optimal controller for the 4-link model. The friction coefficient directly modulates the acceleration of the joints, i.e., $\ddot{\phi}_\gamma = -\gamma\dot{\phi}$.

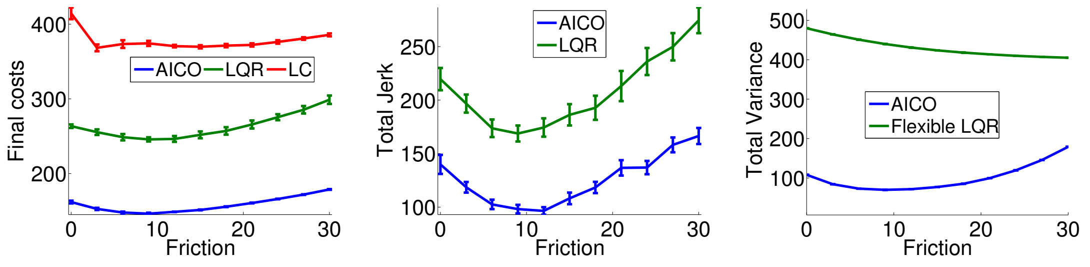
Figure 2.3 (a) shows the resulting final costs for different friction coefficients. In order to illustrate the need for sophisticated control architectures, we compare the optimal control method AICO to a LQR controller and a simple constant linear feedback controller. While AICO calculates the optimal control law for the nonlinear system, the LQR controller linearizes the system at the upright position and subsequently calculates the closed form solution of the optimal controller for the resulting LQG system. The constant linear feedback controller does not adapt to the morphology of the robot. As we can see from Figure 2.3 (a), we cannot determine the optimal morphology with the simple linear controller, while we can recognize a clear minimum for the AICO and the LQR controller.

AICO could find the most simple control law according to the jerk criteria in Figure 2.3 (b) and was also able to find control laws with considerably decreased costs. In Figure 2.3 (c), we illustrated the variance of the time-varying controller gains. The complexity of the resulting controllers is reduced by changing the friction coefficient. The minimum lies in the same range as the minimum of the cost function. Thus, in this case a simpler controller resulted in better performance because the morphology (the friction) of the robot has simplified the control task. Still, we needed a more complex optimal control algorithm to discover this simpler control law if we compare to the LQR.

In Figure 2.4 we depict the joint and torque trajectories for the hip and the arm joint, where we applied the friction coefficients $\gamma = 0$, $\gamma = 12$ and $\gamma = 30$. With $\gamma = 0$ the trajectories overshoot as shown in Figure 2.4 (a), whereas in Figure 2.4 (c) more energy is needed for stabilizing the robot with $\gamma = 30$. The trajectories for the optimal friction coefficient $\gamma = 12$ (according to the jerk complexity measure) are shown in Figure 2.4 (b).

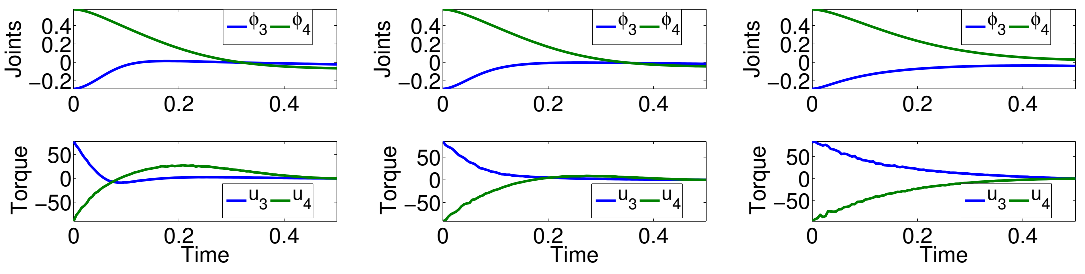
2.3.4 The influence of the link lengths

To show the effects of morphological computation on a more complex variation of morphologies, we consider in this experiment the variation of the shank, the thigh, the trunk, and the arm length $\mathbf{l} = [l_1, l_2, l_3, l_4]$. Initially the link lengths are $\mathbf{l}_0 = [0.5, 0.5, 1, 1]$m and the weights of these links are specified by $\mathbf{m}_0 = [17.5, 17.5, 27.5, 7.5]$kg.



(a) Friction vs. Final Costs (b) Friction vs. Jerk (c) Controller Variance

Figure 2.3: This figure shows the influence of the friction coefficient on controlling the 4-link model using the optimal control methods AICO and LQR. Illustrated are the mean and the standard error over 50 trajectories. As complexity measures of the morphology we used the final costs (a), the total jerk (b) and the controller variance (c). For the final costs we additionally compare to a simple linear feedback controller denoted by LC. For (c) we compare to a LQR controller with time varying costs in contrast to the standard LQR method which uses constant costs for all time steps.



(a) $\gamma = 0$ (b) $\gamma = 12$ (c) $\gamma = 30$

Figure 2.4: The plots show the mean trajectories of the hip (ϕ_3) and the arm (ϕ_4) joints over 100 runs for the first 0.5s, where we used the AICO approach for the friction coefficients $\gamma = 0$, $\gamma = 12$ and $\gamma = 30$.

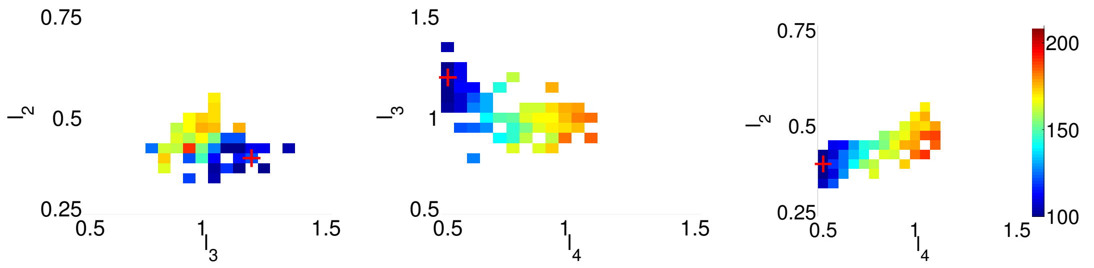
We consider variations of $\mathbf{l}$ of up to 50% from their initial configuration. Different link lengths of the 4-link robot result in different initial velocities [Atkeson and Stephens, 2007], which influences the complexity of the motor control problem. For this reason, we consider constant inertias for a fair comparison. Thus, for each link-length $\mathbf{l}$ the masses of the links are given by $\mathbf{m} = \mathbf{m}_0\mathbf{l}_0{}^2/\mathbf{l}^2$. To match the physical properties of humans we assume equal lengths for the shank and the thigh link $l_1 = l_2$.

In addition to the link lengths of the thigh, the trunk and the arm we still optimize the friction coefficient, resulting into 4 parameters (l_2, l_3, l_4, and γ) of our morphology. A naive search for optimal parameters like in previous experiment is in the multi-dimensional case intractable. Therefore we apply the nonlinear stochastic optimizer Hansen et al. [2003] as a local search method.

The stochastic optimizer locally explores the parameter space (l_2, l_3 and l_4) until convergence. For example, the resulting final cost values for different link lengths are illustrated in Figure 2.5 when using AICO as optimal control law. For this experiment we used a friction coefficient of $\gamma = 12$, where we averaged over multiple initial states and multiple forces. The optimizer converged to link lengths of $l_2 = 0.39$, $l_3 = 1.18$ and $l_4 = 0.5$ for the thigh, the torso and the arm.

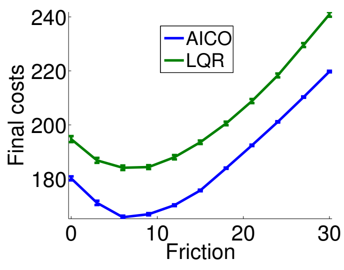
For multiple friction coefficients an evaluation of the presented complexity measures is shown in Figure 2.6. According to the final costs in Figure 2.6 (a) the best controller was found with a friction coefficient of $\gamma = 9$. For the jerk criteria in Figure 2.6 (b) and the variance as complexity measure shown in Figure 2.6 (c), the minimum of the complexity measures lies in the same range.

The morphologies found by the optimizer for the AICO and the LQR controller are sketched in Figure 2.7 for the friction coefficients $\gamma = 0$, $\gamma = 9$ and $\gamma = 30$. The initial configuration of the robot is shown in Figure 2.7 (a) and in Figure 2.7 (e). Interestingly with increasing values of the friction coefficients the link lengths change to compensate for the larger motor controls. This change is shown in Figure 2.7 (b-d) for the AICO controller and in Figure 2.7 (f-h) for the LQR controller. AICO could again find significantly different morphologies as the LQR controller. As is illustrated in the captions of Figure 2.7, the morphologies found by AICO could produce lower costs as the morphologies found by LQR.

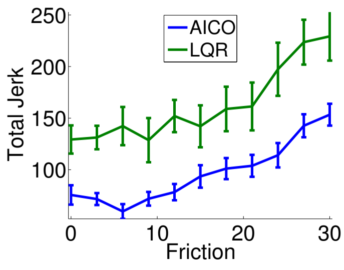


(a) l_2 vs. l_3 (b) l_3 vs. l_4 (c) l_2 vs. l_4

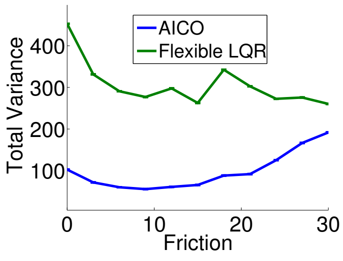
Figure 2.5: This figure illustrates the explored link lengths for the thigh, the torso and the arm (l_2, l_3 and l_4) using AICO with the friction coefficient $\gamma = 12$. For this illustration we discretized the parameter space to visualize the final cost values, which are denoted by the color of the dots. Darker dots correspond to lower cost values as specified by the colorbar on the right. The optimal values of the link lengths are denoted by the large crosses ($l_2 = 0.39$, $l_3 = 1.18$ and $l_4 = 0.5$).



(a) Friction vs. Final Costs

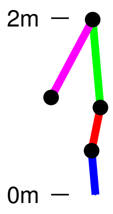


(b) Friction vs. Jerk

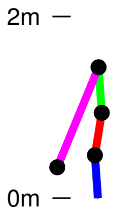


(c) Controller Variance

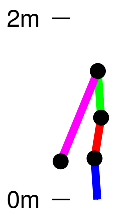
Figure 2.6: This figure shows the results for optimizing the link lengths and the friction coefficient of the 4-link model. Like before we used the final costs (a), the total jerk (b) and the variance of the time-varying feedback controller (c) as complexity measure.



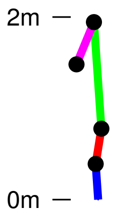
(a) initial configuration



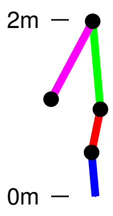
(b) AICO with $\gamma = 0$, $L(\tau) = 180.2$



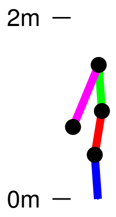
(c) AICO with $\gamma = 9$, $L(\tau) = 166.9$



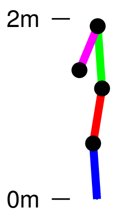
(d) AICO with $\gamma = 30$, $L(\tau) = 219.7$



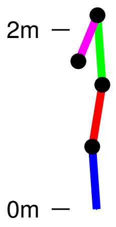
(e) initial configuration



(f) LQR with $\gamma = 0$, $L(\tau) = 194.7$



(g) LQR with $\gamma = 9$, $L(\tau) = 184.4$



(h) LQR with $\gamma = 30$, $L(\tau) = 240.9$

Figure 2.7: This figure shows the learned link lengths for multiple friction coefficients $\gamma = 0$, $\gamma = 9$ and $\gamma = 30$ for the AICO (b-d) and the LQR (f-h) controller. The link lengths are optimized with a stochastic optimizer and could vary up to 50% from the initial configuration shown in (a) and (e). With increasing friction coefficients the link lengths change to compensate for the larger control costs. The final costs $L(\tau)$ for these morphologies are given in the captions.

2.3.5 Robot model with linear springs

Inspired by the theoretical analysis of Hauser et al. [2011], we also evaluate the computational power of linear springs in our model. The springs are mounted on the 4-link model as illustrated in Figure 2.8. These springs support the control of the 4-link robot as they push the robot back into its upright position. Thus, the torques applied to the joints are given by the sum of the controls provided by the feedback controller and the spring forces $\mathbf{u}_t^* = \mathbf{u}_t + \text{diag}(\mathbf{k})\mathbf{\Delta l_s}$. The vector $\mathbf{\Delta l_s}$ denotes the displacements of the four springs and $\mathbf{k}$ is a vector containing the 4 spring constants. Note that due to the calculation of the displacements, the springs act nonlinearly on the joint torques.

The adjustable parameters of the morphology include the friction coefficient and the four spring constants $\mathbf{k} = [k_1, k_2, k_3, k_4]$ for the ankle, the knee, the hip, and the arm joint. Initially the spring constants are set to $\mathbf{k}_0 = [1, 10^3, 10^3, 1]$. For the optimization a lower bound of $0.1\mathbf{k}_0$ and an upper bound of $100\mathbf{k}_0$ are used.

As in the previous experiment, for different friction coefficients γ, we optimize the spring constants using the stochastic search method Hansen et al. [2003]. The results for the final costs, the total jerk and the variance of the feedback controller as complexity measures are shown in Figure 2.9. The benefit of the supporting springs is best illustrated by the final costs (AICO: 109.5 and LQR: 122.2) in Figure 2.9 (a), which are drastically reduced compared to the final costs using the 4-link model without springs (AICO: 146.9 and LQR: 246.9) shown in Figure 2.3 (a).

The optimal spring constants strongly depend on the used friction coefficient. This effect of the friction coefficient is illustrated in Figure 2.10, where we compare the learned spring constants using AICO and LQR as control method. With an increasing friction also the spring constants increase to compensate for larger control costs.

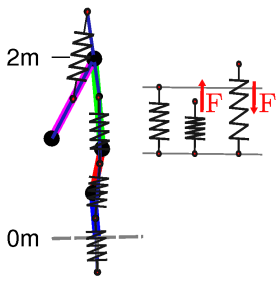
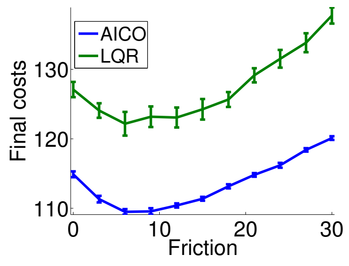
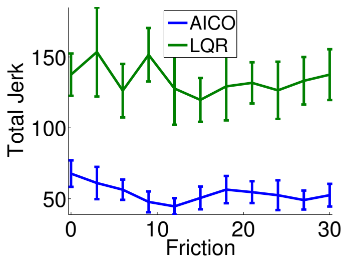


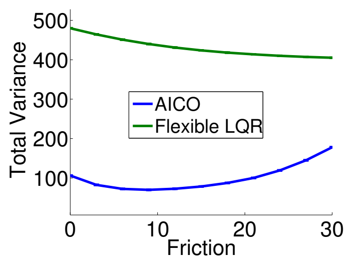
Figure 2.8: This figure illustrates the 4-link robot model endowed with linear springs which are mounted at the midpoints of the joints. These springs support the control of the 4-link robot as they push the robot back into its upright position. Due to the calculation of the spring length displacements, the springs act nonlinearly on the joint torques.



(a) Friction vs. Final Costs

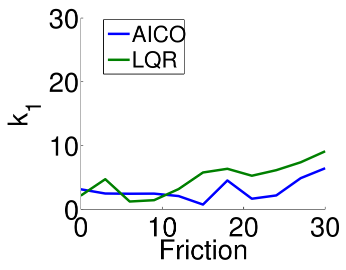


(b) Friction vs. Jerk

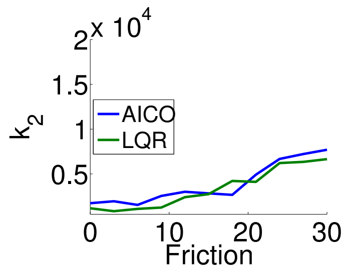


(c) Controller Variance

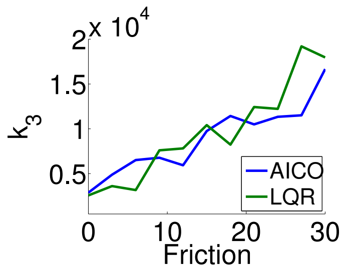
Figure 2.9: This figure shows the results for optimizing the spring constants and the friction coefficient of the 4-link model. We used the final costs (a), the total jerk (b) and the variance of the time-varying feedback controller (c) as complexity measure.



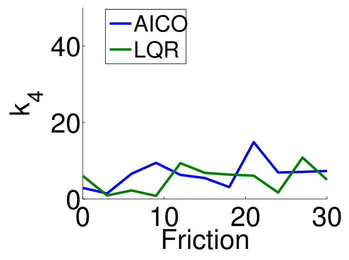
(a) shank



(b) thigh



(c) trunk



(d) arm

Figure 2.10: This figure shows the optimal spring constants for different friction coefficients using AICO and LQR.

2.3.6 Concluding summary

We have shown that by using optimal control methods an optimal morphology could be found for the 4-link robot model. This approach was evaluated on three tasks with increasing complexity, i.e., optimizing only the friction coefficient, optimizing the friction coefficient and the link lengths, and finally, optimizing the friction coefficient and the four spring constants. We can now also ask how beneficial it was to optimize for example the link lengths compared to optimizing the spring constants. This comparison is best reflected by the achieved final costs in Figure 2.3 (a), Figure 2.6 (a), and Figure 2.9 (a), where the most complex task also performs best.

2.4 Conclusion

In this paper we have shown that optimal control and morphological computation are two complementary approaches which can benefit from each other. The search for an optimal morphology is simplified if we can calculate an optimal controller for a given morphology. This calculation can be done by new approaches from probabilistic inference for motor control, i.e., the approximate inference control algorithm [Toussaint, 2009]. By the use of optimal control methods, we have shown that for dynamic nonlinear balancing tasks, an appropriate setting of the friction, the link lengths or the spring constants of the incorporated springs of the compliant robot can simplify the control problem.

We have demonstrated that there are optimal values for the physical properties for a given control task, which can only be found by the use of optimal control laws. With naive control architectures such as the evaluated constant linear feedback controller which does not adapt to the morphology, different, sub-optimal morphologies would be chosen.

In our studies we assumed that we can always compute an optimal controller given the current morphology. If this requirement is fulfilled then the morphological computation can be quantified using the proposed method. However, for computing optimal control laws, using e.g., AICO, we need to know the costs at each time-step, as well as the dynamics model. For more complex tasks these costs and the model are usually unknown. However, as argued in the related work Section 2.1.1, for model learning many types of function approximators can be applied [Vijayakumar et al., 2005, Nguyen-Tuong et al., 2008a,b]. For learning the cost function inverse reinforcement learning methods such as Abbeel and Ng [2004], Boularias et al. [2011] could be used.

In the future, we plan to investigate more complex and more nonlinear tasks. In this case the benefit of AICO in comparison to LQR controllers should be even more prominent. In the end we are planning to simultaneously evolve walking controllers based on AICO and the morphology of bipedal robots like a model of a planar walker.

2.5 Acknowledgment

This chapter is based on the paper Rückert and Neumann [2012] written by Elmar Rückert (ER) and Gerhard Neumann (GN). Together, the two authors developed the basic ideas, the experimental design, and did the writting. The algorithms and experiments were implemented by ER.

Chapter 3

Learned graphical models for probabilistic planning

Contents

Biological movement generation combines three interesting aspects: its modular organization in movement primitives, its characteristics of stochastic optimality under perturbations, and its efficiency in terms of learning. A common approach to motor skill learning is to endow the primitives with dynamical systems. Here, the parameters of the primitive indirectly define the shape of a reference trajectory. We propose an alternative movement primitive representation based on probabilistic inference in learned graphical models with new and interesting properties that complies with salient features of biological movement control.

Instead of endowing the primitives with dynamical systems, we propose to endow movement primitives with an intrinsic probabilistic planning system, integrating the power of stochastic optimal control methods within a movement primitive. The parametrization of the primitive is a graphical model that represents the dynamics and intrinsic cost function such that inference in this graphical model yields the control policy. We parametrize the intrinsic cost function using task-relevant features, such as the importance of passing through certain via-points. The system dynamics as well as intrinsic cost function parameters are learned in a reinforcement learning setting. We evaluate our approach on a complex 4-link balancing task. Our experiments show that our movement representation facilitates learning significantly and leads to better generalization to new task settings without re-learning.

3.1 Introduction

Efficient motor skill learning in redundant stochastic systems is of fundamental interest for both, understanding biological motor systems as well as applications in robotics.

Let us first discuss three aspects of human and animal movement generation the combination of which is the motivation for our approach: (1) its modular organization in terms of movement primitives, (2) its variability and behavior under perturbations, and (3) the efficiency in *learning* such movement strategies.

First, concerning the movement primitives (MPs) in biological motor systems, the musculoskeletal apparatus is a high-dimensional redundant stochastic system and has many more degrees-of-freedom than needed to perform a specific action [Bernstein, 1967]. A classical hypothesis is that such redundancy is resolved by a combination of only a small number of functional units, namely movement primitives [d'Avella et al., 2003, Bizzi et al., 2008, d'Avella and Pai, 2010]. In other terms, MPs can be understood as compact parametrizations of elementary movements which allows for an efficient abstraction of the high-dimensional continuous action spaces. This abstraction has been shown to facilitate learning of complex movement skills [Schaal et al., 2003, Neumann et al., 2009, d'Avella et al., 2003].

A second important aspect about biological movement are the characteristics of motor variability under perturbations or stochasticity. If humans perform the same task several times, the resulting movement trajectories vary considerably. Stochastic optimal control (SOC), besides its high relevance in engineering problems, has proven itself as an excellent computational theory of this effect [Todorov and Jordan, 2002, Trommershauser et al., 2005]. An implication of SOC, the *minimum intervention principle*, states that we should only intervene in the system if it is necessary to fulfill the given task. If the task constraints are not violated it is inefficient to suppress the inherent noise in the stochastic system. The fact that biological movements account for such principles suggests that SOC principles are involved on the lowest level of movement generation.

These biological perspectives suggest that the third aspect, efficient motor skill learning, is facilitated by this combination of MPs with low level SOC principles. While existing MP methods have demonstated efficient learning of complex movement skills [d'Avella et al., 2003, Schaal et al., 2003, Neumann et al., 2009] they lack an integration of SOC principles *within* MPs. Instead, in current approaches the parameters of the MP compactly determine the shape of the desired trajectory either directly or indirectly. This trajectory is then followed by feedback control laws. An example for an indirect trajectory parametrization are the widely used Dynamical Movement Primitives (DMPs) [Schaal et al., 2003], which use parametrized dynamical systems to determine a movement trajectory. The idea of DMPs to endowing MPs with an intrinsic dynamical system has several benefits: They provide a linear policy parametrization which can be used for imitation learning and policy search [Kober and Peters, 2011]. The complexity of the trajectory can be scaled by the number of parameters [Schaal et al., 2003] and one can adapt meta-parameters of the movement such as the movement speed or the goal state of the movement [Kober et al., 2010, Pastor et al., 2009]. Further, the dynamical system of a DMP is to some degree also reactive to perturbations by adapting the time progression of the canonical system depending on joint errors and thereby de- or accelerating the movement execution as needed [Ijspeert and Schaal, 2003, Schaal et al., 2003]. However, the trajectory shape itself is fixed and non-reactive to the environment.

In our approach we aim to go beyond MPs that parametrize a fixed reference trajectory and instead truly integrate SOC principles within the MP. The general idea is to endow MPs with an intrinsic probabilistic planning system instead of an intrinsic dynamical system. Such a *Planning Movement Primitive* (PMP) can react to the environment by optimizing the trajectory for the specific current situation. The intrinsic probabilistic planning system is described as a graphical model that represents the SOC problem [Toussaint, 2009, Kappen et al., 2009]. Training such a MP therefore amounts to learning a graphical model such that inference in the learned graphical model will generate an appropriate policy. This has several implications. First, this approach implies a different level of generalization compared to a dynamical system that generates a fixed (temporally flexible) reference trajectory. For instance, if the end effector target changes between training and testing phase, an intrinsic planning system will generalize to a new target without retraining. A system that directly encodes a trajectory would either have to be retrained or use heuristics to be adapted [Pastor et al., 2009]. Second, this approach truly integrates SOC principles within the MP. The resulting policy follows the minimum intervention principle and is compliant compared to a feedback controller that aims to follow a reference trajectory.

As with DMPs, a PMP is trained in a standard reinforcement learning (RL) setting. Instead of parametrizing the shape of the trajectory directly, a PMP has parameters that determine the intrinsic cost function of the intrinsic SOC system. While the reward function (typically) gives a single scalar reward for a whole movement, the learned intrinsic cost function is in the standard SOC form and defines task and control costs for every time-step of the movement. In other terms, training a PMP means to learn from a sparse reward signal an intrinsic cost function such that the SOC system will, with high probability, generate rewarded movements. Parallel to this learning of an intrinsic cost function, a PMP also exploits the data to learn an approximate model of the system dynamics. This approximate dynamics model is used by the intrinsic SOC system. Therefore, PMP learning combines model-based and model-free RL: it learns a model of the system dynamics while at the same time training PMP parameters based on the reward signal. It does not learn an approximate model of the reward function itself. We can exploit supervised learning methods such as Vijayakumar et al. [2005], Nguyen-Tuong et al. [2008a,b] for learning the system dynamics and at the same time use policy search methods to adapt the PMP parameters that determine the intrinsic cost function. This two-fold learning strategy has the promising property of fully exploiting the data by also estimating the system dynamics instead of only adapting policy parameters.

As mentioned above, our approach is to represent the intrinsic SOC system as a graphical model, building on the recent work on Approximate Inference Control (AICO), [Toussaint, 2009]. AICO generates the movement by performing inference in the graphical model that is defined by the system dynamics and the intrinsic cost function. Since we learn both from experience, all conditional probability distributions of this graphical model are learned in the RL setting. The output of the planner is a linear feedback controller for each time slice.

Our experiments show that by the use of task relevant features, we can significantly facilitate learning and generalization of complex movement skills. Moreover, due to the intrinsic SOC planner, our movement primitive representation implements the principles of optimal control, which allows to learn solutions of high quality which are not representable with traditional trajectory-based methods.

In the following section we review in more detail related previous work and the background on which our methods build. Section 3.3 then introduces the proposed Planning

Movement Primitives. In Section 3.4 we evaluate the system on a one-dimensional via-point task and a complex dynamic humanoid balancing task and compare to DMPs. We conclude with a discussion in Section 3.5.

3.2 Related work and background

We review here the related work based on parametrized movement policies, policy search methods and stochastic optimal control.

3.2.1 Parametrized movement policies

Movement primitives (MPs) are a parametric description of elementary movements [d'Avella et al., 2003, Schaal et al., 2003, Neumann et al., 2009]. We will denote the parameter vector of a MP by $\boldsymbol{\theta}$ and the possibly stochastic policy of the primitive as $\pi(\mathbf{u}|\mathbf{x}, t; \boldsymbol{\theta})$, where $\mathbf{u}$ is the applied action and $\mathbf{x}$ denotes the state. The key idea of the term 'primitive' is that several of these elementary movements can be combined not only sequentially but also simultaneously in time. However, in this paper, we want to concentrate on the parametrization of a single MP. Thus we only learn a single elementary movement. Using several MPs simultaneously is part of future work for our approach as well as for existing approaches such as Schaal et al. [2003], Neumann et al. [2009].

Many types of MPs can be found in the literature. The currently most widely used movement representation for robot control are the Dynamic Movement Primitives (DMPs) [Schaal et al., 2003]. DMPs evaluate parametrized dynamical systems to generate trajectories. The dynamical system is constructed such that the system is stable. In order to do so, a linear dynamical system is used which is modulated by a learnable non-linear function f. A great advantage of the DMP approach is that the function f depends linearly on the parameters $\boldsymbol{\theta}$ of the MP: $f(s) = \boldsymbol{\Phi}(s)^T\boldsymbol{\theta}$, where s is the time or phase variable. As a result, imitation learning for DMPs is straightforward, as this can simply be done by performing a linear regression [Schaal et al., 2003]. Furthermore, it also allows the use of many well-established reinforcement learning methods such as policy gradient methods [Peters and Schaal, 2008] or Policy Improvements by Path Integrals PI^2 [Theodorou et al., 2010]. The complexity of the trajectory can be scaled by the number of features used for modelling f. We can also adapt meta-parameters of the movement such as the movement speed or the goal state of the movement [Kober et al., 2010, Pastor et al., 2009]. However, as the features $\mathbf{\Phi}(s)$ are fixed, the ability of the approach to extract task-relevant features is limited. Yet, the change of the desired trajectory due to the change of the meta-parameters is based on heuristics and does not consider task-relevant constraints. While the dynamical system of a DMP is to some degree reactive to the environment—namely by adapting the time progression of the canonical system depending on joint errors and thereby de- or accelerating the movement execution as needed [Ijspeert and Schaal, 2003, Schaal et al., 2003]—the trajectory shape itself is fixed and non-reactive to the environment. As the DMPs are the most common movement representation, we will use it as a baseline in our experiments. A more detailed discussion of the DMP approach can be found in the appendix.

Another type of movement representation was introduced in Neumann et al. [2009] by the movement template framework. Movement templates are temporally extended, parametrized actions, such as sigmoidal torque, velocity or joint position profiles, which can be sequenced in time. This approach uses a more complex parametrization as the DMPs. For example, it also incorporates the duration of different phases, like an acceleration or deceleration phase. The division of a movement into single phases allows the use of reinforcement learning methods to learn how to sequence these primitives. However, as the approach still directly specifies the shape of the trajectory, defining complex movements for high dimensional systems is still complicated, which has restricted the use of movement templates to rather simple applications.

An interesting movement representation arizing from analysis of biological data are muscle synergies [d'Avella et al., 2003, Bizzi et al., 2008]. They have been used to provide a compact representation of electromyographic muscle activation patterns. The key idea of this approach is that muscle activation patterns are linear sums of simpler, elemental patterns, called muscle synergies. Each muscle synergy can be shifted in time and scaled with a linear factor to construct the whole activation pattern. While the synergy approach has promising properties such as the linear superposition and the ability to share synergies between tasks, except for some smaller applications [Chhabra and Jacobs, 2006], these MPs have only been used for data analysis, and not for robot control.

All the so far presented MPs are inherently local approaches. The specified trajectory and hence the resulting policy are only valid for a local (typically small) neighborhood of our initial state. If we are in a new situation, it is likely that we need to re-estimate the parameters of the MP. The generation of the reference trajectory for these approaches is often an offline process and does not incorporate knowledge of the system dynamics, proprioceptive or other sensory feedback. Because the reference trajectory itself is usually created without any knowledge of the system model, the desired trajectory might not be applicable, and thus, the real trajectory of the robot might differ considerably from the specified trajectory.

There are only few movement representations which can also be used globally, i.e., for many different initial states of the systems. One such methods is the Stable Estimator of Dynamical Systems [Khansari-Zadeh and Billard, 2011] approach. However, this method has so far only been applied to imitation learning, using the approach for learning or improving new movement skills is not straight forward. We will therefore restrict our discussion to local movement representations.

Our Planning Movement Primitive approach is, similar as the DMPs, a local approach. In a different situation, different abstract goals and features might be necessary to achieve a given task. However, as we extract task-relevant features and use them as parameters, the same parameters can be used in different situations as long as the task relevant features do not change. As we will show, the valid region where the local movement primitives can still be applied is much larger for the given control tasks in comparison to trajectory based methods.

3.2.2 Policy search for movement primitives

Let $\mathbf{x}$ denote the state and $\mathbf{u}$ the control vector. A trajectory τ is defined as sequence of state control pairs, $\tau = \langle \mathbf{x}_{1:T}, \mathbf{u}_{1:T-1} \rangle$, where T is the length of the trajectory. Each trajectory has associated costs $C(\tau)$ (denoted as extrinsic cost), which can be an arbitrary function of the trajectory. It can, but need not be composed of the sum of intermediate costs during the trajectory. For example, it could be based on the minimum distance to a given point throughout the trajectory. We want to find a movement primitive's parameter vector $\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ which minimizes the expected costs $J(\boldsymbol{\theta}) = \mathbb{E}\left[C(\tau)|\boldsymbol{\theta}\right]$. We assume that we can evaluate the expected costs $J(\boldsymbol{\theta})$ for a given parameter vector $\boldsymbol{\theta}$ by performing roll-outs on the real system.

In order to find $\boldsymbol{\theta}^*$ we can apply policy search methods. Here a huge variety of possible methods exists. Policy search methods can be coarsely divided into step-based exploration and episode-based exploration approaches. Step-based exploration approaches such as Theodorou et al. [2010], Peters and Schaal [2008], Kober and Peters [2011] apply an exploration noise to the action of the agent at each time-step of the episode. Subsequently, the policy is updated such that the (noisy) trajectories with higher reward are more likely to be repeated. In order to do this update, step-based exploration techniques strictly rely on a policy which is linear in its parameters. This requirement is fulfilled for the Dynamic Movement Primitives (DMPs) [Schaal et al., 2003]. Currently, the most common policy search methods are step-based approaches, including the REINFORCE [Williams, 1992], the episodic Natural Actor Critic [Peters and Schaal, 2008], the PoWER [Kober and Peters, 2011] or the PI^2 [Theodorou et al., 2010] algorithm. This also explains partially the popularity of the DMP approach for motor skill learning because DMPs are, from those introduced above, the only representation which can be used for these step-based exploration methods (apart from very simple ones like linear controllers).

However, recent research has also intensified on episode-based exploration techniques that make no assumptions on a specific form of a policy [Sehnke et al., 2010, Wierstra et al., 2008, Hansen et al., 2003]. These methods directly perturb the policy parameters $\boldsymbol{\theta}$ and then estimate the performance of the perturbed $\boldsymbol{\theta}$ parameters by performing roll-outs on the real system. During the episode no additional exploration is applied (i.e., a deterministic policy is used). The policy parameters are then updated in the estimated direction of increasing performance. Thus, these exploring methods do not depend on a specific form of parametrization of the policy. In addition, they allow the use of second order stochastic search methods that estimate correlations between policy parameters [Hansen et al., 2003, Heidrich-Meisner and Igel, 2009a, Wierstra et al., 2008]. This ability to apply correlated exploration in parameter-space is often beneficial in comparison to the uncorrelated exploration techniques applied by all step-based exploration methods, as we will demonstrate in the experimental section.

3.2.3 Stochastic optimal control and probabilistic inference for planning

Stochastic optimal control (SOC) methods such as Todorov and Li [2005], Kappen [2007], Toussaint [2009] have been shown to be powerful methods for movement planning in high-dimensional robotic systems. The incremental Linear Quadratic Gaussian (iLQG) [Todorov and Li, 2005] algorithm is one of the most commonly used SOC algorithms. It uses Taylor expansions of the system dynamics and cost function to convert the non-linear control problem in a Linear dynamics, Quadratic costs and Gaussian noise system (LQG). The algorithm is iterative - the Taylor expansions are recalculated at the newly estimated optimal trajectory for the LQG system.

In Toussaint [2009], the SOC problem has been reformulated as inference problem in a graphical model, resulting in the Approximate Inference Control (AICO) algorithm. The graphical model is given by a simple dynamic Bayesian network with states $\mathbf{x}_t$, actions $\mathbf{u}_t$ and task variables $\mathbf{g}^{[i]}$ (representing the costs) as nodes, see Figure 3.1. The dynamic Bayesian network is fully specified by conditional distributions encoded by the cost function and by the state transition model. If beliefs in the graphical model are approximated as Gaussian the resulting algorithm is very similar to iLQG. Gaussian message passing iteratively re-approximates local costs and transitions as LQG around the current mode of the belief within a time slice. A difference to iLQG is that AICO uses forward messages instead of a forward roll-out to determine the point of local LQG approximation and can iterate belief re-approximation with in a time slice until convergence, which may lead to faster overall convergence. For a more detailed discussion of the AICO algorithm with Gaussian message passing see Section 3.3.5.

Local planners have the advantage that they can be applied to high-dimensional dynamical systems, but the disadvantage of requiring a suitable initialization. Global planning [Kuffner and LaValle, 2000] on the other hand does not require an initial solution, however, they have much higher computational demands. Our motivation for using only a local planner as component of a Planning Movement Primitive is related to the learning of an intrinsic cost function.

Existing planning approaches for robotics typically use hand-crafted intrinsic cost functions and the dynamic model is either analytically given or learned from data [Mitrovic et al., 2010]. Planning Movement Primitives (PMPs) use reinforcement learning (RL) to train an intrinsic cost function for planning instead of trying to learn a model of the extrinsic reward directly. The reason is that a *local* planner often fails to directly solve realistically complex tasks by optimizing directly the extrinsic cost functions. From this perspective, PMPs learn to translate complex tasks to a simpler intrinsic cost function that can efficiently be optimized by a local planner. This learning is done by trial-and-error in the RL setting: the PMP essentially learns from experience which intrinsic cost function the local planner can cope with and uses it to generate good trajectories. Thereby, the reinforcement learning of the intrinsic cost function can compensate the limitedness of the local planner.

3.3 Materials and methods

In this section we introduce the proposed Planning Movement Primitives (PMPs), in particular the parametrization of the intrinsic cost function. The overall system will combine three components: (1) a regression method for learning the system dynamics,

(2) a policy search method for finding the PMP parameters, and (3) a SOC planner for generating movements with the learned model and PMP parameters.

3.3.1 Problem definition

We consider an unknown dynamical system of the form

$$\mathbf{x}_{t+1} = f_{\mathrm{Dyn}}(\mathbf{u}_t, \mathbf{x}_t) + \epsilon_t, \tag{3.1}$$

with state variable $\mathbf{x}_t$, controls $\mathbf{u}_t$ and Gaussian noise $\epsilon_t \sim \mathcal{N}(0, \sigma)$. The agent has to realize a control policy $\pi : \mathbf{x}_t \mapsto \mathbf{u}_t$, which in our case will be a linear feedback controller for each time slice. The problem is to find a policy that minimizes the expected costs of a finite-horizon episodic task. That is, we assume there exists a cost function $C(\tau)$, where $\tau = (\mathbf{x}_{1:T}, \mathbf{u}_{1:T-1})$ is a roll-out of the agent controlling the system. We do not assume that the cost function $C(\tau)$ is analytically known or can be written as sum over individual costs for each time-step, i.e., $C(\tau) = \sum_t h_t(\mathbf{x}_t, \mathbf{u}_t)$. This would imply an enormous credit assignment problem with separate costs at each time-step. Thus more generally, we only get a single scalar reward $C(\tau)$ for the whole trajectory. The problem is to find $\mathrm{argmin}_\pi \langle C(\tau)\rangle_\pi$.

The system dynamics f_{Dyn} as well as the cost function $C(\tau)$ are analytically unknown. Concerning the system dynamics we can estimate an approximate model of the systems dynamics from a set of roll-outs—as standard in model-based reinforcement learning (RL). However, concerning costs, we only receive the single scalar cost $C(\tau)$ after a roll-out indicating the quality or success of a movement. Note that $C(\tau)$ is a function of the whole trajectory, not only of the final state. Learning C from data would be an enormous task, more complex than learning an immediate reward function $\mathbf{x}_t \mapsto \mathbf{r}_t$ as in standard model-based RL where $\mathbf{r}_t$ denotes the reward at time t.

Generally, approaches to learn $C(\tau)$ directly in a form useful for applying SOC methods seems an overly complex task and violates the maxim "never try to solve a problem more complex than the original". Therefore, our approach will not try to learn $C(\tau)$ from data but to employ reinforcement learning to learn *some* intrinsic cost function that can efficiently be optimized by SOC methods and generates control policies that, by empiricism, minimizes $C(\tau)$.

3.3.2 Parametrization of PMP's intrinsic cost function

In PMPs the parameters $\boldsymbol{\theta}$ encode task-relevant abstract goals or features of the movement, which specify an intrinsic cost function

$$L(\tau;\boldsymbol{\theta}) \quad := \quad \sum_{t=0}^{T} l(\mathbf{x}_t,\mathbf{u}_t,t;\boldsymbol{\theta}) + c_p(\mathbf{x}_t,\mathbf{u}_t), \tag{3.2}$$

where l denotes the intermediate intrinsic cost function for every time-step and $c_p(\mathbf{x}_t,\mathbf{u}_t)$ is used to represent basic known task constraints, such as torque or joint limits. We will assume that such basic task constraints are part of our prior knowledge, thus c_p is given and not included in our parametrization. For the description of PMPs we will neglect the constraints c_p for simplicity. We will use a via-point representation for the intermediate intrinsic cost function $l(\mathbf{x}_t,\mathbf{u}_t,t;\boldsymbol{\theta})$. Therefore, parameter learning corresponds to extracting goals which are required to achieve a given task, such as passing through a via-point at a given time. As pointed out in the previous section, $L(\tau;\boldsymbol{\theta})$ is *not* meant to approximate $C(\tau)$. It provides a feasible cost function that empirically generates policies that minimize $C(\tau)$.

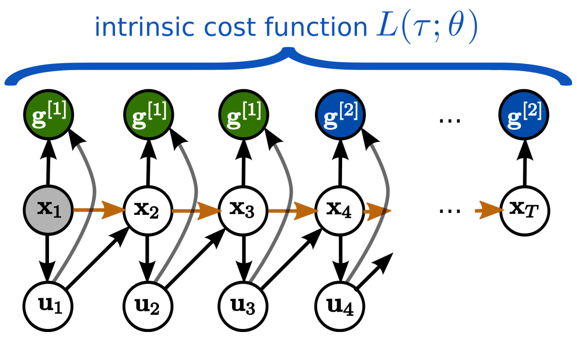


Figure 3.1: Planning Movement Primitives are endowed with an intrinsic planning system, which performs inference in a *learned* graphical model. States are denoted by $\mathbf{x}_t$, controls by $\mathbf{u}_t$, and the time horizon is fixed to T time-steps. In this example the graphical model is used to infer the movement by conditioning on two abstract goals $\mathbf{g}^{[1]}$ and $\mathbf{g}^{[2]}$, which are specified in the *learned* intrinsic cost function $L(\tau;\boldsymbol{\theta})$.

There are many ways to parametrize the intermediate intrinsic cost function l. In this paper we choose a simple via-point approach. However, in an ongoing study we additionally implemented a desired energy state of a pendulum on a cart, which simplifies the learning problem. The movement is decomposed in N shorter phases with duration $d^{[i]}$, $i = 1,..,N$. In each phase the cost function is assumed to be quadratic in the state and control vectors. In the ith phase ($t < d^{[1]}$ for $i = 1$ and $\sum_{j=1}^{i-1} d^{[i]} < t \leq \sum_{j=1}^{i} d^{[i]}$ for $i > 1$) we assume the intrinsic cost has the form:

$$l(\mathbf{x}_t, \mathbf{u}_t, t; \boldsymbol{\theta}) \quad = \quad (\mathbf{x}_t - \mathbf{g}^{[i]})^T \mathbf{R}^{[i]} (\mathbf{x}_t - \mathbf{g}^{[i]}) + \mathbf{u}_t^T \mathbf{H}^{[i]} \mathbf{u}_t. \tag{3.3}$$

It is parametrized by the via-point $\mathbf{g}^{[i]}$ in state space; by the precision vector $\mathbf{r}^{[i]}$ which determines $\mathbf{R}^{[i]} = \text{diag}(\exp \mathbf{r}^{[i]})$ and therefore how steep the potential is along each state dimension; and by the parameters $\mathbf{h}^{[i]}$ which determine $\mathbf{H}^{[i]} = \text{diag}(\exp \mathbf{h}^{[i]})$ and therefore the control costs along each control dimension. We represent the importance factors $\mathbf{r}^{[i]}$ and $\mathbf{h}^{[i]}$ both in log space as we are only interested in the relationship of these factors. At the end of each phase (at the via-point), we multiply the quadratic state costs by the factor $1/\Delta t$ where Δt is the time step used for planning. This ensures that at the end of the phase the via-point is reached, while during the phase the movement is less constraint. With this representation, the parameters $\boldsymbol{\theta}$ of our PMPs are given by

$$\boldsymbol{\theta} = [d^{[1]}, \mathbf{g}^{[1]}, \mathbf{r}^{[1]}, \mathbf{h}^{[1]} \ \dots \ d^{[N]}, \mathbf{g}^{[N]}, \mathbf{r}^{[N]}, \mathbf{h}^{[N]}]. \tag{3.4}$$

Cost functions of this type are commonly used—and hand-crafted—in control problems. They allow to specify a via-point, but also to determine whether only certain dimensions of the state need to be controlled to the via-point, and how this trades off with control cost. Instead of hand-designing such cost functions, our method will use a policy search method to learn these parameters of the intrinsic cost function. As for the DMPs we will assume that the desired final state at time index T is known, and thus $\mathbf{g}^{[N]}$ is fixed and not included in the parameters. Furthermore, since we consider finite-horizon episodic tasks the duration of the last phase is also fixed: $d^{[N]} = T - \sum_{i=1}^{N-1} d^{[i]}$. Still, the algorithm can choose the importance factors $\mathbf{r}^{[N]}$ and $\mathbf{h}^{[N]}$ of the final phase.

3.3.3 Dynamic model learning

Planning Movement Primitives are endowed with an intrinsic planning system. For planning we need to learn a model of the system dynamics f_{Dyn} in Equation 3.1. The planning algorithm cannot interact with the real environment, it solely has to rely on the learned model. Only after the planning algorithm is finished, the resulting policy is executed on the real system and new data points $\langle [\mathbf{x}_t, \mathbf{u}_t], \dot{\mathbf{x}}_t \rangle$ are collected for learning the model.

Many types of function approximators can be applied in this context [Vijayakumar et al., 2005, Nguyen-Tuong et al., 2008a,b]. We use the lazy learning technique Locally

Weighted Regression (LWR) [Atkeson et al., 1997] as it is a very simple and effective approach. LWR is a memory-based, non-parametric approach, which fits a local linear model to the locally-weighted set of data points. For our experiments, the size of the data set was limited to 10^5 points implemented as a first-in-first-out queue buffer, because the computational demands of LWR drastically increase with the size of the data set. In particular we used a Gaussian kernel as distance function with the bandwidth parameters $h_\phi = 0.25$ for joint angles, $h_{\dot{\phi}} = 0.5$ for velocities and $h_u = 0$ for controls*. For more details we refer to Chapter 4 in Atkeson et al. [1997].

3.3.4 Policy search for PMP's intrinsic cost function

Model learning takes place simultaneously to learning the parameters $\boldsymbol{\theta}$ of the movement primitive (MP). In general this could lead to some instability. However, while the distribution $P(\mathbf{x}_t)$ depends on the policy and the data for model learning is certainly non-stationary, the conditional distribution $P(\mathbf{x}_{t+1}|\mathbf{u}_t, \mathbf{x}_t)$ is stationary. A local learning scheme as LWR behaves rather robust under such type of non-stationarity of the input distribution only. On the other hand, from the perspective of $\boldsymbol{\theta}$ optimization, the resulting policies may change and lead to different payoffs $C(\tau)$ even for the same parameters $\boldsymbol{\theta}$ due to the adaption of the learned system dynamics.

Since the resulting control policies of our PMPs depend non-linearly on the parameters $\boldsymbol{\theta}$, step-based exploration techniques cannot be used in our setup. Hence, we will use the second order stochastic search method CMA (Covariance Matrix Adaptation, Hansen et al. 2003) which makes no assumptions on the parametrization of the movement primitive.

We employ the second order stochastic search method CMA to optimize the parameters $\boldsymbol{\theta}$ w.r.t. $C(\tau)$. The parameter space is approximated using a multivariate Gaussian distribution. Roughly, CMA is an iterative procedure that, from the current Gaussian distribution, generates a number of samples, evaluates the samples, computes second order statistics of those samples that reduced $C(\tau)$ and uses these to update the Gaussian search distribution. In each iteration, all parameter samples $\boldsymbol{\theta}$ use the same learned dynamic model to evaluate $C(\tau)$. Further, CMA includes an implicit forgetting in its update of the Gaussian distribution and therefore behaves robust under the non-stationarity introduced by adaptation of the system dynamics model.

We will compare our PMP approach to both, DMPs learned with CMA policy search and DMPs learned with the state of the art step-based method PI^2 [Theodorou et al., 2010].

*The bandwidth parameter for controls is set to zero as the matrizes $\mathbf{A}$ and $\mathbf{B}$ in the linearized model, i.e., $\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}(\mathbf{x})\mathbf{u}$ are independent on the controls $\mathbf{u}$.

However, we focus in this work on the characteristics of the movement representation and place less emphasis on a specific policy search method.

Note that even if the learned model is only a rough approximation of the true dynamics, the policy search for parameters of the intrinsic cost function can compensate for an imprecise dynamics model: The policy search approach finds parameters $\boldsymbol{\theta}$ of the intrinsic cost function such that—even with a mediocre model—the resulting controller will lead to low extrinsic costs in the real system.

3.3.5 Probabilistic planning algorithm

We use the probabilistic planning method Approximate Inference Control (AICO) [Toussaint, 2009] as intrinsic planning algorithm. It offers the interpretation that a movement primitive can be represented as graphical model and the movement itself is generated by inference in this graphical model.

The graphical model is fully determined by the learned system dynamics and the learned intrinsic cost function, see Figure 3.1. In order to transform the minimization of $L(\tau;\theta)$ into an inference problem, for each time-step an individual binary random variable z_t is introduced. This random variable indicates a reward event. Its probability is given by

$$P(z_t = 1|\mathbf{x}_t, \mathbf{u}_t, t) \propto \exp(-l(\mathbf{x}_t, \mathbf{u}_t, t; \boldsymbol{\theta})),$$

where $l(\mathbf{x}_t, \mathbf{u}_t, t; \boldsymbol{\theta})$ denotes the cost function for time-step t defined in Equation 3.3. AICO now assumes that a reward event $z_t = 1$ is observed at every time-step. Given that evidence, AICO calculates the posterior distribution $P(\mathbf{x}_{1:T}, \mathbf{u}_{1:T-1}|z_{1:T} = 1)$ over trajectories.

We will use the simplest version of AICO [Toussaint, 2009], where an extended Kalman smoothing approach is used to estimate the posterior distribution $P(\mathbf{x}_{1:T}, \mathbf{u}_{1:T-1}|z_{1:T} = 1)$. The extended Kalman smoothing approach uses Taylor expansions to linearize the system and subsequently uses Gaussian messages for belief propagation in a graphical model. Gaussian message passing iteratively re-approximates local costs and transitions as a Linear dynamics, Quadratic costs and Gaussian noise system (LQG) around the current mode of the belief within a time slice. For more details we refer to [Toussaint, 2009].

AICO is only a local optimization method and we have to provide an initial solution which is used for the first linearization. We will use the direct path (or the straight line) to the via-points $\mathbf{g}^{[i]}$ in Equation 3.3 as initial solution. Before learning the via-points $\mathbf{g}^{[i]}$

with $i = 1..N-1$ are set to the initial state $\mathbf{x}_1$. The final via-point is fixed and set to the desired final state $\mathbf{g}^{[N]} = \mathbf{x}_T$.

AICO provides us with a linear feedback controller for each time slice of the form

$$\mathbf{u}_t = \mathbf{O}_t \mathbf{x}_t + \mathbf{o}_t, \tag{3.5}$$

where $\mathbf{O}_t$ is the inferred feedback control gain matrix and $\mathbf{o}_t$ denotes the linear feedback controller term. This feedback control law is used as policy of the movement primitive and is evaluated on a simulated or a real robot.

The original formulation of the AICO method [Toussaint, 2009] does not consider torque limits, which are important for many robotic experiments as well for the dynamic balancing experiments we consider in this paper. Therefore we extended the algorithm. This extension yields not only a modified form of the immediate cost function but also results in different update equations for the messages and finally different equations of the optimal feedback controller. A complete derivation of the extension including the resulting messages and the corresponding feedback controller is given in Rückert and Neumann [2012]. Also the algorithm is listed in that work.

On overview of the interactions between policy search of the PMP's intrinsic cost function and the planning process using AICO is sketched in Figure 3.2. The learning framework is organized the following: Given the parameters $\boldsymbol{\theta}$ from the policy search method CMA, AICO is initialized with an initial solution which is the direct path to the via-points. AICO is then used to optimize the parametrized intrinsic cost function $L(\tau;\theta)$ to estimate a linear feedback controller for each time-step, see Equation 3.5. The feedback controller is subsequently executed on the simulated or the real robot and the extrinsic cost $C(\tau)$ is evaluated. Based on this evidence CMA will update its distribution over the policy search space and computes a new parameter vector. Simultaneously we collect samples of the system dynamics $\langle[\mathbf{x}_t, \mathbf{u}_t], \dot{\mathbf{x}}_t\rangle$ while executing the movement primitive. These samples are used to improve our learned dynamics model, which is used for planning.

3.4 Results

We start our evaluation of the proposed Planning Movement Primitive (PMP) approach on a one-dimensional via-point task to illustrate basic characteristics. In order to demonstrate our approach on a more challenging dynamic robot task we choose a complex 4-link humanoid balancing task. At the end of this section we discuss an important issue: the

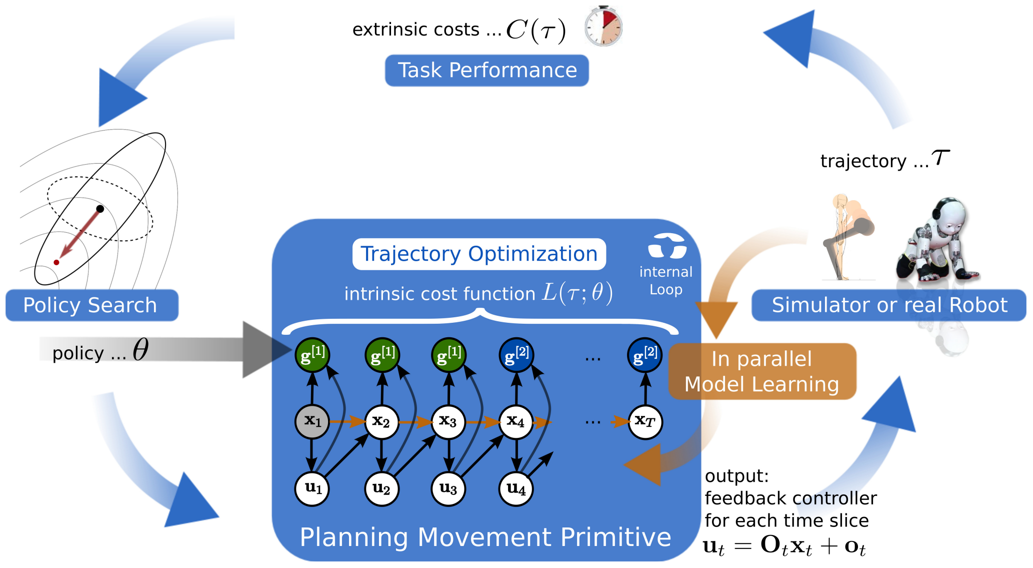


Figure 3.2: We decompose motor skill learning into two different learning problems. At the highest level we learn parameters $\boldsymbol{\theta}$ of an intrinsic cost function $L(\tau;\boldsymbol{\theta})$ using policy search (model-free RL). Given parameters $\boldsymbol{\theta}$ the probabilistic planner at the lower level uses the intrinsic cost function $L(\tau;\boldsymbol{\theta})$ and the learned dynamics model to estimate a linear feedback controller for each time-step (model-based RL). The feedback controller is subsequently executed on the simulated or the real robot and the extrinsic cost $C(\tau)$ is evaluated. Based on this evidence the policy search method computes a new parameter vector. Simultaneously we collect samples of the system dynamics $\langle[\mathbf{x}_t,\mathbf{u}_t],\dot{\mathbf{x}}_t\rangle$ while executing the movement primitive. These samples are used to improve our learned dynamics model which is used for planning.

computational time of PMPs for simulated and real world tasks.

In our experiments, we focus on investigating the optimality of the solution, the robustness to noise for learning, and the generalizability to different initial or final states. For the 4-link task we additionally demonstrate how model learning influences the learning performance.

For a comparison we take the commonly used DMPs as a baseline where we use the newest version of the DMPs [Pastor et al., 2009] as discussed in detail in Appendix C.1. As described above we use 2nd order stochastic search to learn the PMP and DMP parameters. In order to compare to a more commonly used policy search algorithm we additionally test the PI^2 algorithm [Theodorou et al., 2010] for learning the DMP parameters. For all experiments we empirically evaluate the optimal settings of the algorithms (such as the exploration rate of CMA and PI^2, the number of centers for the DMPs, or the number of via-points for the PMPs), which are listed in Appendix C.2.

3.4.1 One-dimensional via-point task

In this task the agent has to control a one dimensional point mass of 1kg. The state at time t is denoted by $\mathbf{x}_t = [\phi_t, \dot{\phi}_t]^T$ and we directly control the acceleration u_t. The time horizon was limited to $T = 50$ time-steps, which corresponds to a simulation time of 0.5 seconds with a time-step of $\Delta t = 10$ms. Starting at $\mathbf{x}_1 = [0, 0]^T$ the agent has to pass through a given via-point $g_v = -0.2$ at $t_v = 30$. The velocity of the point mass at the via-point is not specified and can have any value. The final target g_T was set to 1. The movement is shown in Figure 3.3. For this task we define the extrinsic cost function:

$$C(\tau) = 10^4(\dot{\phi}_T^2 + 10(g_T - \phi_T)^2) + 10^5(g_v - \phi_{t_{30}})^2 + 5 \cdot 10^{-3} \sum_{t=1}^{T} u_t^2.$$

The first two terms punish deviations from the target g_T and the via-point g_v, where $\phi_{t_{30}}$ denotes the first dimension of the state $\mathbf{x}_t = [\phi_t, \dot{\phi}_t]^T$ at time index 30. The target should be reached with zero velocity at $T = 50$. The last term punishes high energy consumption where u_t denotes the applied acceleration. The control action is noisy, we always add a Gaussian noise term with a standard deviation of $\sigma = 20$ to the control action. As this experiment is a very simple task, we use it just to show different characteristics of the DMPs (using 10 Gaussians for that representation was optimal) and PMPs (apparently 2 via-points are sufficient for this task).

A quite similar task has been used in Todorov and Jordan [2002] to study human movement control. The experiments showed that humans were able to reach the given via-points with high accuracy, however, in between the via-points, the trial-to-trial variability was rather high. This adaptive motor variability is a well known concept from optimal control, called the *minimum intervention principle*, showing also that human movement control follows basic rules of optimal control.

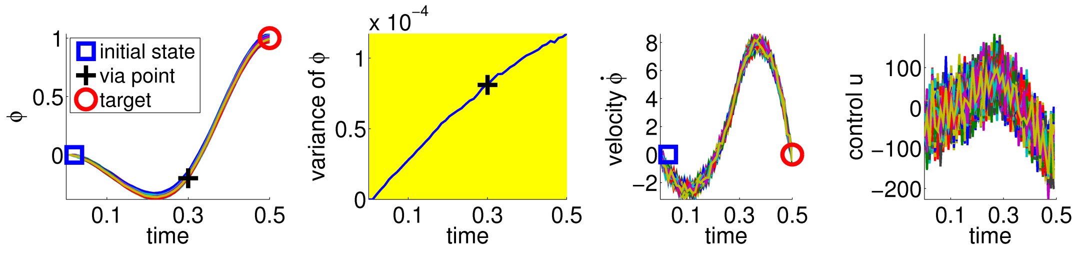
Optimality of the solutions We first estimate the quality of the *best available* policy with the DMP and the PMP approach. We therefore use the PMPs with two via-points and set the parameters $\boldsymbol{\theta}$ per hand. As we are using a linear system model and a simple extrinsic cost function, the PMP parameters can be directly obtained by looking at the extrinsic costs. As the PMPs use the AICO algorithm, which always produces optimal policies for LQG systems, the PMP solution is the optimal solution. We subsequently use the mean trajectory returned by AICO and use imitation learning to fit the DMP parameters. We also optimized the feedback controllers used for the DMPs[†]. In Figure 3.3 we plotted 100 roll-outs of the DMP and PMP approach using this optimal policies. The second column illustrates the trial-to-trial variability of the trajectories. The optimal solution has minimum variance at the via-point and the target. As expected this solution is reproduced with the PMP approach, because the parameters of the PMPs are able to reflect the importance of passing through the via-point. The DMPs could not adapt the variance during the movement because the used (optimized) feedback controller uses constant controller gains. As we can see, the variance of the DMP trajectory is simply increasing with time.

Comparing the optimal solutions we find that PMPs, in contrast to DMPs, can naturally deal with the inherent noise in the system. This feature is also reflected by the average cost values over 1000 trajectories, 1286 ± 556 for the DMPs and 1173 ± 596 for the PMPs. The $\pm$ symbol always denotes the standard deviation. PMPs perform significantly better than DMPs (t-test: $p < 10^{-3}$).

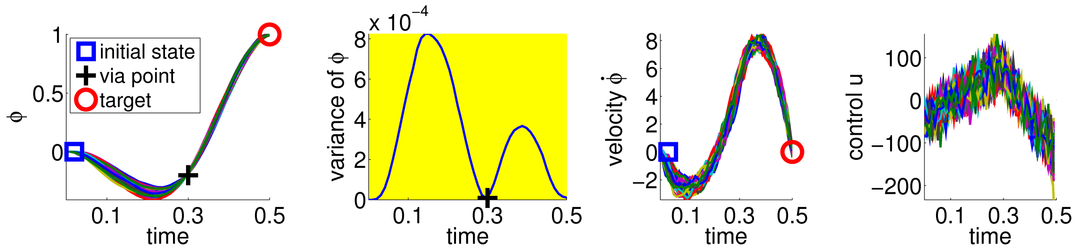
Robustness to noise for learning This advantage would not be very useful if we were not able to learn the optimal PMP parameters from experience. Next we test using CMA policy search to learn the parameters for the DMPs and the PMPs. In addition, in order to compare to a more commonly used policy search method, we also compare to the PI^2 approach [Theodorou et al., 2010] which we could only evaluate for the DMP approach. We evaluated the learning performance in the case of no control noise, Figure 3.4 (a), and in the case of control noise $\sigma = 20$, Figure 3.4(b) performing 15 runs.

Without control noise the quality of the learned policy found by 2nd order search is similar for the DMPs (657.5$\pm$0.18) and the PMPs (639.6$\pm$0.01). PI^2 could not find as good solutions as the stochastic search approach. The reason for this worse performance is that PI^2 could not find the very large weight values which are needed for the last few centers of the DMPs in order to have exactly zero velocity at the final state (note that the weights of the DMPs are multiplied by the phase variable s which almost vanishes in the end of the movement and therefore these weight values have to be very high). Because CMA policy search uses second order information, such large parameter values are easily found. This comparison clearly shows that using 2nd order search for policy search is justified. If we compare the learning speed in terms of required episodes or roll-outs between DMPs and PMPs, we find an advantage for PMPs which could be learned an order of magnitude

[†]The control gains, i.e., the two scalars k_{pos} and k_{vel} of the linear feedback controller in Equation C.1 in the Appendix are learned using a 2nd order stochastic search method.



(a) Optimal policy learned with DMPs (average costs over 1000 trajectories: 1286 ± 556)



(b) Optimal policy learned with PMPs (average costs over 1000 trajectories: 1173 ± 596)

Figure 3.3: This figure illustrates the best available policies for the DMPs and the PMPs for the via-point task. From left—to—right shown are the point mass trajectories, the variance of these trajectories, the velocity of the point mass, and the applied accelerations. The agent has to pass the via-point at $t_v = 0.3$s and deal with the stochasticity of the system (Gaussian control noise with a variance of 20^2). The plots show 100 trajectories reproduced with the optimal parameters for the DMPs (a) and 100 trajectories with the (hand-crafted) optimal parameters for PMPs (b). The PMP approach is able to reduce the variance of the movement if it is relevant for the task, while the DMPs can only suppress the noise in the system throughout the trajectory in order to get an acceptable score. This advantage is also reflected by the average costs over 1000 trajectories. The DMP solution achieved cost values of 1286 ± 556 whereas the PMP result was 1173 ± 596.

faster than the DMPs.

The second experiment (with control noise of $\sigma = 20$) was considerably harder to learn. Here, we needed to average each performance evaluation over 20 roll-outs. The use of more sophisticated extensions of CMA [Heidrich-Meisner and Igel, 2009b] which can deal with noisy performance evaluations and hence improve the learning speed of CMA policy search in the noisy setup is part of future work. In Figure 3.4 (b) we find that the PMPs could be learned an order of magnitude faster than the DMPs. As expected from the earlier experiment, the PMPs could find clearly better solutions as the DMPs as they can adapt the variance of the trajectory to the task constraints. Again, PI^2 showed a worse performance than 2nd order search. Illustrated are mean values and standard deviations over 15 runs of learning (1034 ± 1.46 for the PMPs and 1876 ± 131 for the DMPs using CMA). To compare these results to the optimal costs we evaluated the best learned policies of both approaches and generated 1000 trajectories. The learned solution for the PMPs was similar to the hand-coded optimal solution, 1190 ± 584 versus costs of 1173 ± 596 for the optimal solution. DMPs achieved costs of 1478 ± 837, illustrating that, eventhough the DMPs are able to represent much better solutions with costs of 1286 ± 556 (see Figure 3.3), it is very hard to find this solution.

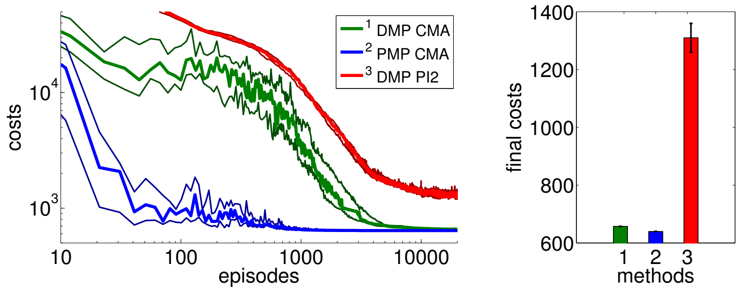
In Table 3.1, we show the mean and variance of the found parameters averaged over 15 runs for the first via-point in comparison to the optimal PMP parameters. We can see that the found parameters matched the optimal ones. Interestingly, in the experiment with no noise, the found parameters had a larger deviation from the optimal ones, especially for the first via-point $g^{[1]}$ in Table 3.1. The reason for this larger deviation is the simple observation that without noise, we can choose many via-points which results in the same trajectory, whereas with noise we have to choose the correct via-point in order to reduce the variance of the trajectory at this point in time.

scenario	$d^{[1]}$	$g^{[1]}$	$\log(\mathbf{r}^{[1]})$	$\log(\mathbf{h}^{[1]})$
optimal	0.3	$-\mathbf{0.2}$	$[5, 0]$	-2.3
no noise	0.29 ± 0.01	$-\mathbf{0.27} \pm 0.03$	$[4.08 \pm 4.18, -0.8 \pm -0.77]$	-3.05 ± -4
with noise	0.29 ± 0.01	$-\mathbf{0.23} \pm 0.05$	$[4.93 \pm 5.29, -0.31 \pm -0.12]$	-2.85 ± -3

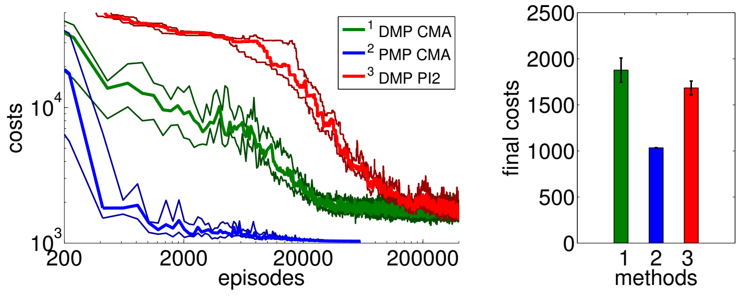
Table 3.1: Learned parameters using PMPs for the via-point task (1st via-point). The symbol $\pm$ denotes the standard deviation.

Generalizability to different task settings Next, we investigate the ability of both approaches to adapt to different situations or to adapt to different priors. In the previous task, Figure 3.3 the initial and the target state were assumed as prior knowledge. The movement was learned for the initial state $\phi_1 = 0$ and for the target state $\phi_T = 1$. We want to investigate if the same learned parameters can be re-used to generate different movements, e.g., used for different initial or target states.

For PMPs we use the new initial or final states denoted by $\mathbf{x}_1$ and $\mathbf{g}^{[N]}$ in the graphical model in Figure 3.1 and re-plan the movement (using the same learned parameters). The change of the initial state or the target state is also allowed by the DMP framework.



(a) Learning performance without noise

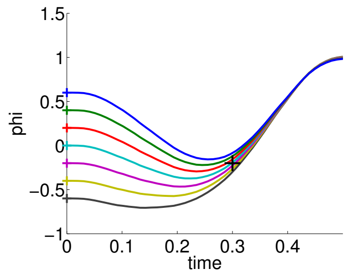


(b) Learning performance with noise

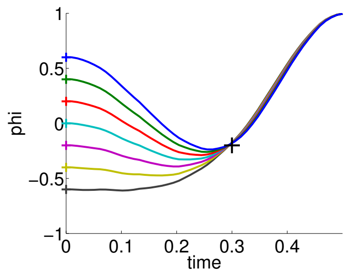
Figure 3.4: This figure illustrates the learning performance of the two movement representations, DMPs and PMPs, for the one-dimensional via-point task. Illustrated are mean values and standard deviations over 15 runs after CMA policy search. In addition, we also compare to the PI^2 approach [Theodorou et al., 2010] which we could only evaluate for the DMP approach. Without noise the final costs of the two representations are similar if CMA policy search is used (a). In the second example (b) we use zero-mean Gaussian noise with $\sigma = 20$ for the controls. In this setup we needed to average each performance evaluation for CMA over 20 roll-outs. For both setups the PMPs could considerably outperform the DMPs in terms of learning speed. For the noisy setup the PMPs could additionally produce policies of much higher quality as they can adapt the variance of the trajectories to the task constraints. PI^2 could not find as good solutions as the CMA policy search approach in both setups.

However, how the movement is generalized to these new situations is based on heuristics [Pastor et al., 2009] and does not consider any task constraints (in this example to pass through the via-point).

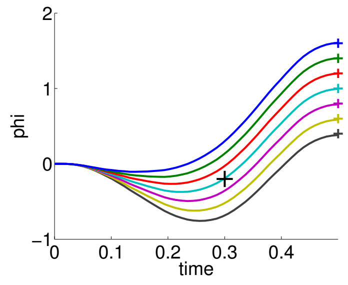
In Figure 3.5 the learned policies are applied to reach different initial states $\phi_1 \in \{-0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6\}$ (a-b) and different goal states $\phi_T \in \{1.5, 1.25, 1, 0.75, 0.5\}$ (c-d). All plots show the mean trajectory. In order to change the initial or the target state of the movement we have to change the point attractor of the DMPs, which changes the complete trajectory. Due to this heuristic, the resulting DMP trajectories shown in Figure 3.5 (a,c) do not pass through the via-point any more. Note that we use a modified version of the DMPs [Pastor et al., 2009] which has already been built for generalization to different initial or target points. The PMPs on the other hand still navigate through the learned via-point when changing the initial or the goal state as shown in Figure 3.5 (b,d).



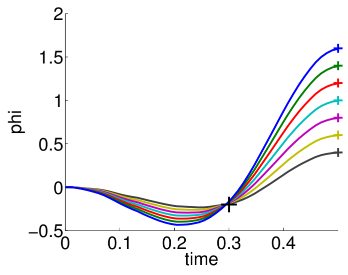
(a) DMPs - varying ϕ_1



(b) PMPs - varying ϕ_1



(c) DMPs - varying ϕ_T



(d) PMPs - varying ϕ_T

Figure 3.5: For the previous task illustrated in Figure 3.3 the movement was learned for a single initial state $\phi_1 = 0$ and a single target state $\phi_T = 1$. The initial and the target state were assumed as prior knowledge. In this experiment we evaluated the generalization of the learned policies to different initial states $\phi_1 \in \{-0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6\}$ (a-b) and different target states $\phi_T \in \{1.5, 1.25, 1, 0.75, 0.5\}$ (c-d). Always the same parameters $\boldsymbol{\theta}$ have been used, i.e., the parameters were not re-learned. Illustrated are the mean trajectories. The DMPs (a,c) are not aware of task-relevant features and hence do not pass through the via-point any more. (b,d) PMPs can adapt to varying initial or target states with small effects on passing through the learned via-point.

Concluding summary As we have seen the PMPs implement all principles of optimal control, which allows to learn solutions for stochastic systems of a quality which is not representable with traditional trajectory based methods such as the DMPs. The optimal movement trajectory could be learned from scratch up to one order of magnitude faster compared to DMPs. This difference was even more visible in the stochastic case, where the DMPs needed more than 30000 episodes to find satisfactory solutions. In the setting with control noise the learned parameters matches the optimal ones because only by the use of noise the parameters are uniquely determined. Finally the PMPs could extract the task-relevant feature, the via-point. Even if the task changes — e.g., the initial or the final state are changed, the movement trajectory still passes through the learned via-point. The DMPs on the other hand heuristically scale the trajectory which offers no control for fulfilling task relevant constraints.

With DMPs 12 parameters were learned, where we used 10 Gaussian kernels and optimized 2 control gains. For the PMPs 2 via-points were sufficient, where the last one was fixed. However, for both via-points we could specify 3 importance weights. Thus, in total $8 = 2 + 3 + 3$ parameters were learned.

3.4.2 Dynamic humanoid balancing task

In order to assess the PMPs on a more complex task, we evaluate the PMP and DMP approach[‡] on a dynamic non-linear balancing task [Atkeson and Stephens, 2007]. The robot gets pushed with a specific force F and has to keep balance. The push results in an immediate change of the joint velocities. The motor torques are limited, which makes direct counter-balancing of the force unfeasible. The optimal strategy is therefore to perform a fast bending movement and subsequently return to the upright position, see Figure 3.6. This task is a very non-linear control problem, applying any type of (linear) balancing control or local optimal control algorithm such as AICO with the extrinsic cost function fails. Thus, we have to use a parametric movement representation. Like in the previous experiment, we take the Dynamic Movement Primitive (DMP) [Schaal et al., 2003] approach as a baseline.

We use a 4-link robot as a simplistic model of a humanoid (70kg, 2m) [Atkeson and Stephens, 2007]. The 8-dimensional state $\mathbf{x}_t$ is composed of the arm, the hip, the knee and the ankle positions and their velocities. Table 1 in [Rückert and Neumann, 2012] shows the initial velocities (resulting from the force F which always acts at the shoulder of the robot) and the valid joint angle range for the task. In all experiments the applied force was $F = 25$Ns. If one of the joints leaves the valid range the robot is considered to be fallen. Additionally to the joint limits, the controls are limited to the intervals $[\pm 250, \pm 500, \pm 500, \pm 70]$Nm (arm, hip, knee and ankle). For more details we refer to Atkeson and Stephens [2007].

[‡]For PMPs again 2 via-points were optimal. DMPs performed best when using 10 Gaussian kernels per dimension.

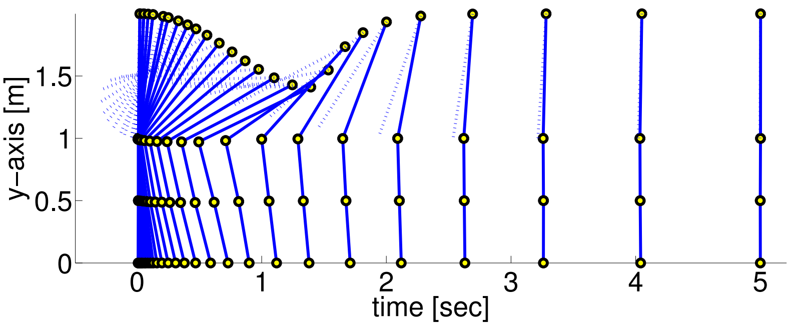


Figure 3.6: This figure illustrates a dynamic balancing movement learned using the proposed Planning Movement Primitives. The 4-link robot modelling a humanoid (70kg, 2m) gets pushed from behind with a specific force ($F = 25$Ns) and has to move such that it maintains balance. The optimal policy is to perform a fast bending movement and subsequently return to the upright robot posture. The circles denote the ankle, the knee, the hip and the shoulder joint. The arm link is drawn as dotted line to differentiate it from the rest of the body.

Let t_s be the last time index where the robot has not fallen and let $\mathbf{x}_{t_s}$ be the last valid state. The final state or resting state (upright position with zero velocity) is denoted by $\mathbf{x}_r$. The movement was simulated for 5 seconds with a $\Delta t = 10$ms resulting in $T = 500$ time-steps. As extrinsic cost function $C(\tau)$ we use:

$$C(\tau) = 20(t_s - T)^2 + (\mathbf{x}_{t_s} - \mathbf{x}_r)^T \mathbf{R}_E (\mathbf{x}_{t_s} - \mathbf{x}_r) + \sum_{t=1}^{t_s} \mathbf{u}_t^T \mathbf{H}_E \mathbf{u}_t \ . \tag{3.6}$$

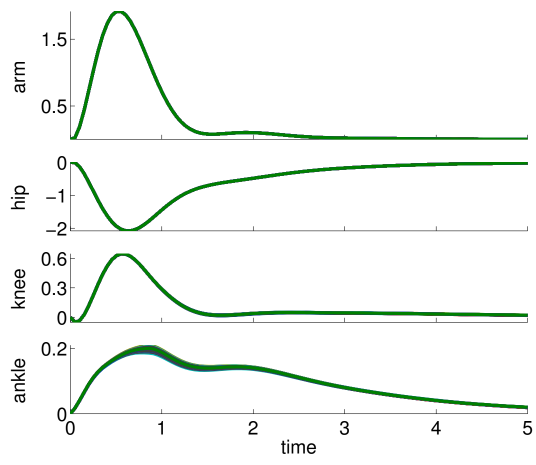
The first term $(t_s - T)^2$ is a punishment term for falling over. If the robot falls over, this term typically dominates. The precision matrix $\mathbf{R}_E$ determines how costly it is not to reach $\mathbf{x}_r$. The diagonal elements of $\mathbf{R}_E$ are set to 10^3 for joint angles and to 10 for joint velocities. Controls are punished by $\mathbf{H}_E = 5 \cdot 10^{-6}\mathbf{I}$. Because of the term $(t_s - T)^2$ we cannot directly encode the extrinsic cost function as a sum of intermediate costs, which is usually required for stochastic optimal control algorithms. But we can use PMPs to transform this reward signal into an intrinsic cost function for a local probabilistic planner.

Optimality of the solutions We use additive zero-mean Gaussian noise with a standard deviation $\sigma = 10$. In contrast to the simple via-point task where imitation learning was used to compare the trajectories shown in Figure 3.3 are the policies for the 4-link task learned from scratch. Figure 3.7 illustrates the best learned policies for DMPs (left column) and PMPs (right column). Shown are the joint angle trajectories Figure 3.7 (a-b) and the variance of these trajectories Figure 3.7 (c-d). The corresponding controls are illustrated in Figure 3.8. We evaluated 100 roll-outs of the best policies found by both approaches. While the DMPs cannot adapt the variance during the movement Figure 3.7 (c), the PMPs can exploit the power of stochastic optimal control and are able to reduce the variance at the learned via-point (marked by crosses) Figure 3.7 (d). As the PMPs are able to control the variance of the trajectory, we can see that the variance of the movement is much higher compared to the DMPs (c-d). Accuracy only matters at the via-points. We can also see that the arm trajectory has a high variance after the robot is close to a stable up-right posture Figure 3.7 (b), because it is not necessary to strictly control the arm in this phase. The best found policy of the DMPs had costs of 568 while the best result using PMPs was 307. This strongly suggests that it is advantageous to reduce the variance at certain points in time in order to improve the quality of the policy.

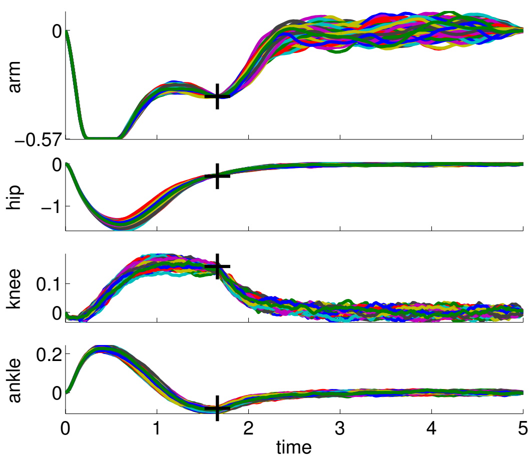
Robustness to noise for learning Next, we again want to assess the learning speed of both approaches. We again used CMA policy search for the PMPs and DMPs as well as PI^2 for the DMP approach. The average over 20 runs of the learning curves are illustrated in Figure 3.9. Using the PMPs as movement representation, good policies could be found at least one order of magnitude faster compared to the trajectory based DMP approach. The quality of the found policies was better for the PMP approach (mean values and standard deviations after learning: 993 ± 449 for the DMPs and 451 ± 212 for the PMPs). For the DMP approach we additionally evaluated PI^2 for policy search, however, PI^2 was not able to find good solutions—the robot always fell over.

Generalizability to different task settings In the next step we again test the generalization to different initial or final states. More specific we investigate how well the approaches can adapt to different priors of the arm joint.

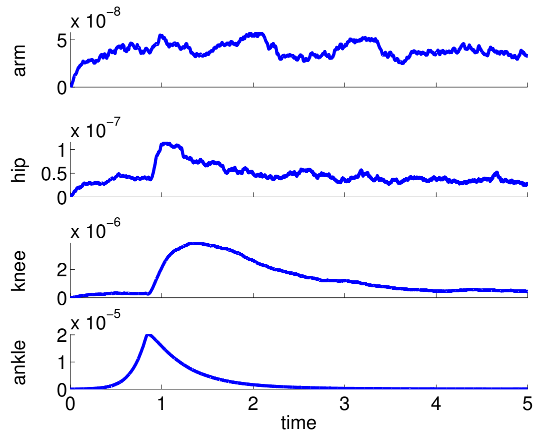
In the previous task the target was assumed to be known prior knowledge and the policy was learned for a final arm posture of $\phi_{T_{\mathrm{arm}}} = 0$. We used this learned policy to generate movements to different final targets of the arm joint $\phi_{T_{\mathrm{arm}}} \in \{3, 2.5, 2, 1.5, 1, 0.5, 0, -0.2, -0.4, -0.6\}$. We only change either the arm-position of the last via-point or the point attractor of the dynamical system. The results shown in Figure 3.10 confirm the findings of the one-dimensional via-point task. The PMPs first move to the via-point, always maintaining the extracted task constraints, and afterwards move the arm to the desired position while keeping balance. All desired target positions of the arm could be fulfilled. In contrast, the DMPs managed to keep



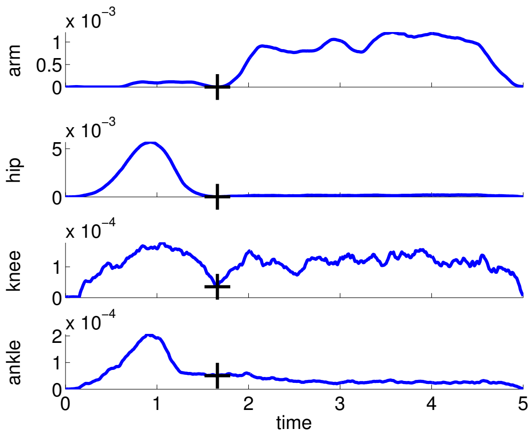
(a) Joint angles using DMPs



(b) Joint angles using PMPs

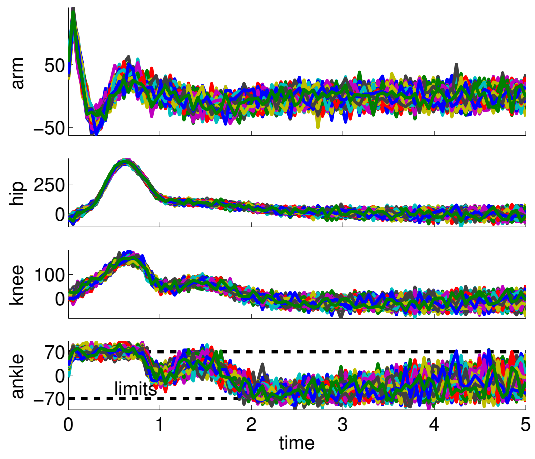


(c) Variance of the joints using DMPs

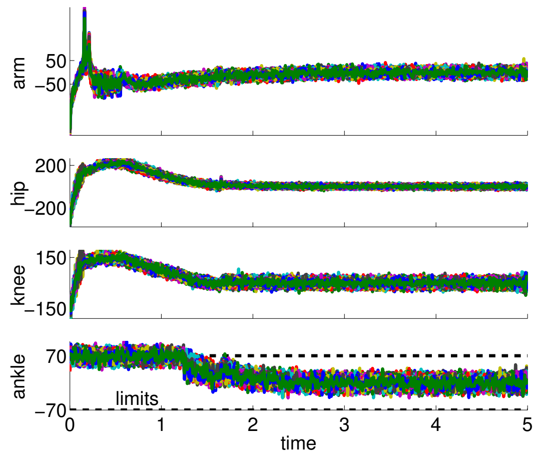


(d) Variance of the joints using PMPs

Figure 3.7: This figure illustrates the best learned policies for the 4-link balancing task using DMPs (left column) and PMPs (right column). Shown are the joint angle trajectories (a-b) and the variance of these trajectories (c-d). The applied controls are illustrated in Figure 3.8. We evaluated 100 roll-outs using the same parameter setting $\boldsymbol{\theta}$ for each approach. The controls were perturbed by zero-mean Gaussian noise with $\sigma = 10$Ns. PMPs can exploit the power of stochastic optimal control and the system is only controlled if necessary, see the arm joint trajectories in (b). The learned via-points are marked by crosses in (b) and (c). For DMPs the variance of the joint trajectories (c) is determined by the learned controller gains of the inverse dynamics controller. As constant controller gains are used the variance cannot be adapted during the movement and is smaller compared to (d). For DMPs the best available policy achieved cost values of 568 whereas the best result using PMPs was 307.



(a) Controls using DMPs



(b) Controls using PMPs

Figure 3.8: This figure illustrates the controls of best learned policies for the 4-link balancing task using DMPs (a) and PMPs (b). The corresponding joint angle trajectories are shown in Figure 3.7. The controls were perturbed by zero-mean Gaussian noise with $\sigma = 10$Nm. Shown are 100 roll-outs using the same parameter setting $\boldsymbol{\theta}$ for each approach. The dotted horizontal lines in the last row indicate the control constraints.

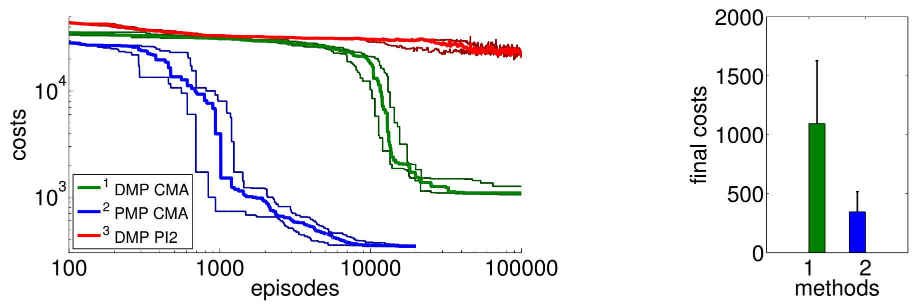
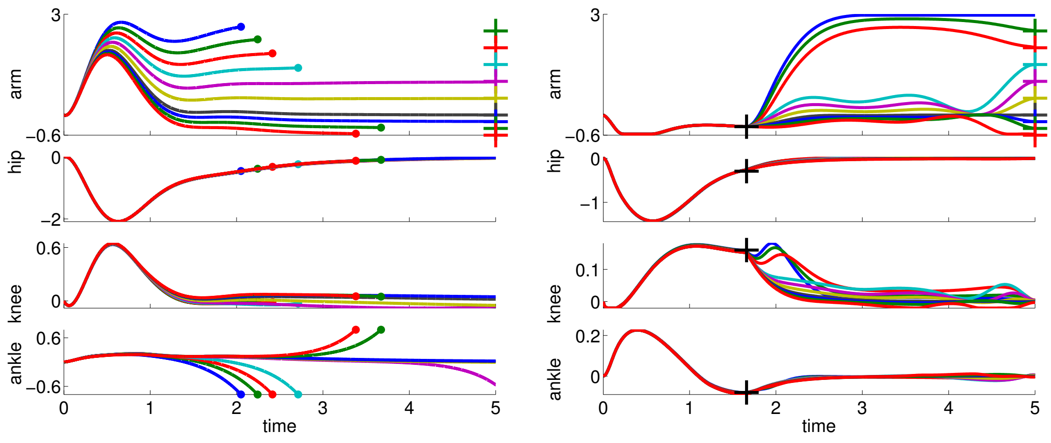


Figure 3.9: The figure illustrates the learning performance of the two movement representations, DMPs and PMPs for the 4-link balancing task. Illustrated are mean values and standard deviations over 20 runs after CMA policy search. The controls (torques) are perturbed by zero-mean Gaussian noise with $\sigma = 10$Nm. The PMPs are able to extract characteristic features of this task which is a specific posture during the bending movement, shown in Figure 3.7 (b). Using the proposed Planning Movement Primitives good policies could be found at least one order of magnitude faster compared to the trajectory based DMP approach. Also, the quality of the best-found policy was considerably better for the PMP approach (993 ± 449 for the DMPs and 451 ± 212 for the PMPs). For the DMP approach we additionally evaluated PI^2 for policy search, which could not find as good solutions as the CMA policy search approach.

balance only for few target positions. The valid range of the target arm position with DMPs was $\phi_{T_{\text{arm}}} \in [-0.2, 1]$. This shows the advantage of generalization while keeping task constraints versus generalization per using the DMP heuristics.

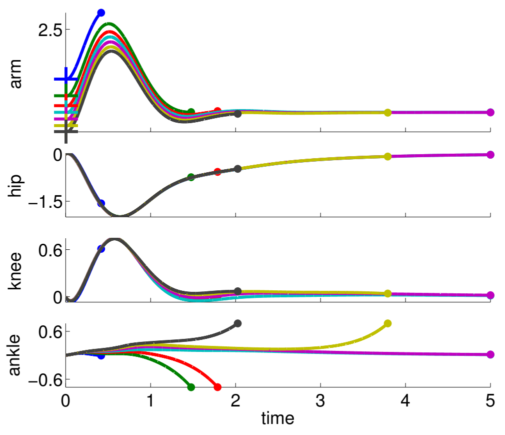


(a) DMPs for changing targets (b) PMPs for changing targets

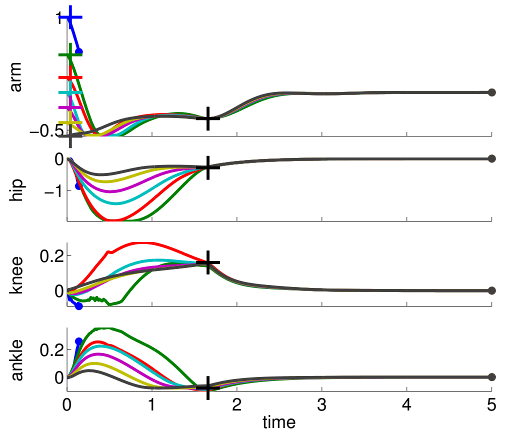
Figure 3.10: This figure illustrates the joint angle trajectories (arm, hip, knee and ankle) of a 4-link robot model during a balancing movement for different final targets of the arm joint $(3, 2.5, 2, 1.5, 1, 0.5, 0, -0.2, -0.4, -0.6)$. The applied policies were learned for a final arm posture of $\phi_{T_{\text{arm}}} = 0$. (a) The valid range of the arm joint using DMPs is $\phi_{T_{\text{arm}}} \in [-0.2, 1]$. Large dots in the plot indicates that the robot has fallen. (b) PMPs could generate valid policies for all final arm configurations.

The ability of the two approaches to adapt to different initial states is illustrated in Figure 3.11. We used the learned policy for $\phi_{1_{\text{arm}}} = 0$ to generate movements using different initial states of the arm joint: $\phi_{1_{\text{arm}}} = \{1, 0.5, 0.2, 0, -0.2, -0.4, -0.6\}$. The push perturbing the robot results in an immediate change of the joint velocities, which are shown in Table C.5 in the appendix for these different initial states. For the DMPs only the joint angles 0 and -0.2 resulted in successful policies. Whereas with PMPs the valid range of the initial arm position was $\phi_{1_{\text{arm}}} \in [-0.6, 0.5]$.

Model learning using PMPs So far all experiments for the PMPs were performed using the known model of the system dynamics, these experiments are denoted by PMP in Figure 3.12. Note that also for the DMPs the known system model has been used for inverse dynamics control. Now we want to evaluate how model learning affects the performance of our approach. This can be seen in Figure 3.12. In the beginning of learning the extrinsic costs are larger compared to motor skill learning with a given analytic model. However, as the number of collected data-points $\langle [\mathbf{x}_t; \mathbf{u}_t], \dot{\mathbf{x}}_t \rangle$ increases the PMPs



(a) DMPs for changing initial states



(b) PMPs for changing initial states

Figure 3.11: This figure illustrates the joint angle trajectories (arm, hip, knee and ankle) of a 4-link robot model during a balancing movement for different initial states of the arm joint $(1, 0.5, 0.2, 0, -0.2, -0.4, -0.6)$. The applied policies were learned for an initial arm posture of $\phi_{1_{\text{arm}}} = 0$. (a) The valid range of the arm joint using DMPs is $\phi_{1_{\text{arm}}} \in [-0.2, 0]$. Large dots in the plot indicates that the robot has fallen. (b) PMPs could generate valid policies for $\phi_{1_{\text{arm}}} \in [-0.6, 0.5]$.

with model learning quickly catch up and converge finally to the same costs. The PMP representation with model learning in parallel considerably outperforms the trajectory based DMP approach in learning speed and in the final costs.

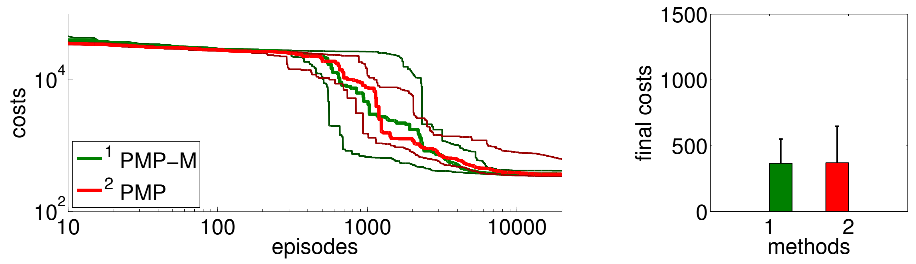


Figure 3.12: This figure shows the influence of model learning on the 4-link balancing task. Illustrated are mean values and standard deviations over 20 runs. The learning performance with the given system model is denoted by PMP. Instead of using the given model we now want to learn the system model from data (as described in Section 3.3.3), which is denoted by PMP-M. In the beginning of learning the extrinsic costs are larger compared to motor skill learning with a given analytic model. However, as the number of collected data-points $\langle[\mathbf{x}_t; \mathbf{u}_t], \dot{\mathbf{x}}_t\rangle$ increases the PMPs with model learning quickly catch up and converge finally to the same costs. The PMP representation with model learning in parallel considerably outperforms the trajectory based DMP approach in learning speed and in the final costs.

Computational time For our simulations we used a standard personal computer (2.6Ghz, 8Gb ram) with implementations of the algorithms in C++. For the 4-link pendulum the dynamic movement primitives could generate the movement trajectory within less than 0.1 seconds. With the proposed planning movement primitives it took less than 1 second (including model learning). The time horizon was 5 seconds and we used a time-step of $\Delta t = 10$ms.

Concluding summary In contrast to the via-point task the optimal solution for this dynamic balancing task is unknown. The comparison to the DMPs shows a similar result as for the via-point task. With the PMPs we could find movements of significantly higher quality compared to the DMPs and the motor skill could be learned up to one order of magnitude faster with PMPs. We again applied the same parameters to different initial or final states to demonstrate the generalization ability. Now we can see the advantage of the learned task-relevant features. While the PMPs still try to fulfill the extracted task relevant constraints and therefore succeeded for almost all initial / final state configurations, the DMPs again just heuristically scale the trajectory, which results in the robot falling in almost all but the learned configurations. Finally we showed that the dynamic model could be learned in parallel for the 4-link balancing task.

For all balancing experiments shown in this section the robot was pushed with the specific force $F = 25$Ns. We have performed the same evaluations for various forces and the results are basically the same. For example a comparison of the learning performances using the negative force $F = -25$Ns is shown in Figure 3.13. The executed movement of the best learned policy using PMPs is shown in Figure 3.14.

In total 40 weights and 8 control gains were learned with DMPs. For PMPs we used 2 via-points where the second one was fixed. Thus, we learned 8 parameters specifying the first via-point and additionally 12 importance weights for each via-point. This results in 32 parameters.

3.5 Discussion

In this paper we concentrated on three aspects of biological motor control, which are also interesting for robotic motor skill learning: (1) The modularity of the motor system, which makes it possible to represent high-dimensional action spaces in terms of lower-dimensional movement primitives, (2) its variability and behavior under stochasticity, and (3) the efficiency in learning movement strategies.

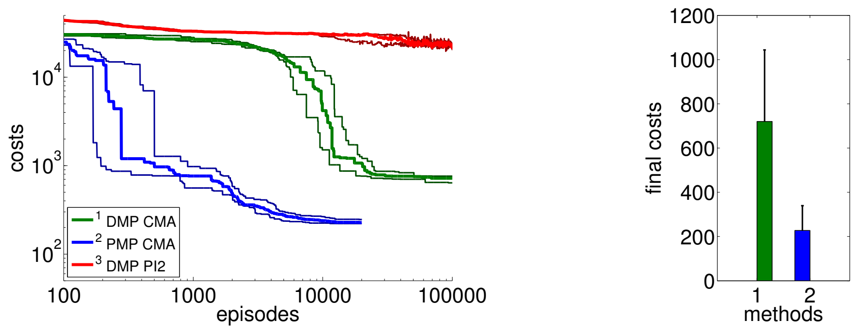


Figure 3.13: This figure illustrates the learning performance of the two movement representations, DMPs and PMPs for the 4-link balancing task. But in contrast to the evaluations shown in the experimental section did we here apply a *negative* force $F = -25\text{Ns}$. Illustrated are mean values and standard deviations over 20 runs after CMA policy search. The controls (torques) are perturbed by zero-mean Gaussian noise with $\sigma = 10\text{Nm}$. Using the proposed Planning Movement Primitives good policies could be found at least one order of magnitude faster compared to the trajectory based DMP approach. Also, the quality of the best-found policy was considerably better for the PMP approach (720 ± 323 for the DMPs and 227 ± 112 for the PMPs). For the DMP approach we additionally evaluated PI^2 for policy search which could not find good policies.

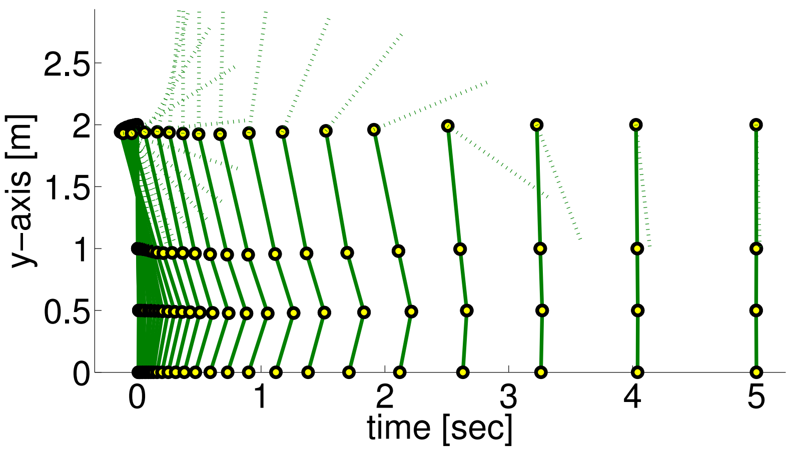


Figure 3.14: This figure illustrates a dynamic balancing movement learned using the proposed Planning Movement Primitives. The 4-link robot modelling a humanoid (70kg, 2m) gets pushed with the specific force ($F = -25\text{Ns}$) and has to move such that it maintains balance. The circles denote the ankle, the knee, the hip and the shoulder joint.

In order to achieve similar properties also for robotic motor skill learning we propose to exploit the power of stochastic optimal control (SOC) already at the level of the movement primitive (MP). Instead of endowing a MP with a dynamical system, like the Dynamic Movement Primitives (DMPs) [Schaal et al., 2003], we endow a MP with an intrinsic probabilistic planning system. The resulting MP is called Planning Movement Primitive (PMP). For the dynamical systems approach the parameters of the MP indirectly define the shape of the trajectory. In our case, the parameters of the MP define the intrinsic cost function of a graphical model, which represents a SOC problem. Performing inference in this graphical model yields the controls for executing the movement.

Due to the use of the intrinsic planning system our representation complies with basic principles of SOC. For example, the PMPs are able to account for the motor variability often observed in human motion. Instead of suppressing the noise of the system by following a single reference trajectory, the PMPs are able to learn to intervene the system only if it is necessary to fulfill a given task, also known as the minimum intervention principle [Todorov and Jordan, 2002]. This allows a much higher variance in parts of the trajectory where less accuracy is needed. Current methods which rely on a reference trajectory are not able to reproduce these effects.

The parameters of PMPs encode learned task-relevant features of the movement, which are used to specify the intrinsic cost function for the MP's intrinsic planning system. Our experiments have shown that such a task-related parametrization facilitates learning and generalization of movement skills. Policies of higher quality could be found an order of magnitude faster than with the competing DMP approach. In addition, as confirmed by our experiments, the intrinsic planner also allows a wide generalization of the learned movement, such as generalization to different initial or goal positions. The DMPs on the other hand have to use heuristics for this generalization [Pastor et al., 2009], which had the consequence that the robot typically fell over in a new situation. In this case relearning is needed for the DMPs while the PMPs allow to reuse the learned parameters.

In traditional SOC methods [Todorov and Li, 2005, Kappen, 2007, Toussaint, 2009] the intrinsic cost function is typically hand-crafted. In contrast we learn the cost function from experience. We considered a general class of motor skill learning tasks where only a scalar reward is observed for the whole movement trajectory. Thus, with PMPs this external sparse reward signal is used to learn the intrinsic cost function. We applied the second order stochastic search method CMA [Hansen et al., 2003] for finding appropriate intrinsic cost functions. In this paper we focused on the representation of movements, and placed less emphasis on a specific policy search method. We want to point out again that our method does not depend on the used policy search method, any episode-based exploring policy search method can be used. We also do not want to argue for using episode-based exploring methods for policy search, however, as our experiments show, these methods provide useful alternatives to the more commonly used step-based approaches such as the PI^2 algorithm [Theodorou et al., 2010]. Future work will concentrate on more grounded approaches for extracting immediate costs from a sparse reward signal. This can also be of interest for imitation learning, where we do not know the immediate costs used by the demonstrator, but often we can evaluate the demonstrators behavior by an external reward signal.

The planner requires to know the system dynamics, which we also learn from data. As shown by our experiments this can be done without significant loss of performance. Hence, our approach combines model-based and model-free reinforcement learning (RL). As in model-based RL, we learn a system model to plan with. Model-free RL is used as method to search for appropriate intrinsic cost functions. We used the Locally Weighted Regression [Atkeson et al., 1997] for learning the system dynamics as it is a very simple and effective approach. Future work will also concentrate on more complex robot models where more sophisticated methods like Vijayakumar et al. [2005], Nguyen-Tuong et al. [2008a,b] could be applied for model learning.

In our experiments the number of phases was fixed ($N = 2$). It was assumed as prior knowledge and can model the complexity of the movement representation (Similarly, the complexity of DMPs can be scaled by the number of Gaussian activation functions). During our experiments we also evaluated the balancing task with up to $N = 5$ phases, but more than 2 phases did not improve the quality of the learned policy. One via-point on the other hand was not sufficient to describe the movement.

A promising topic for future investigation is the combination of primitives in order to achieve several tasks simultaneously. This combination of primitives is still a mostly unsolved problem for current movement representations. Because of the non-linear task demands and system dynamics a naive linear combination in trajectory space usually fails. Here, our Planning Movement Primitives offers new opportunities. New movements can be inferred by a linear combination of cost functions, which results in a non-linear combination of the policies for the single tasks.

3.6 Acknowledgments

This chapter is based on the paper Rückert et al. [2013] written by Elmar Rückert (ER), Gerhard Neumann (GN), Marc Toussaint (MT), and Wolfgang Maass (WM). Together, GN and ER developed the basic ideas and the experimental design. The algorithms were implemented by ER, who also performed the experiments. The paper was written by ER and GN with highly appreciated assistance from MT. WM greatly improved the paper by contributing important thoughts on how to relate this work to salient features of biological motor control.

Chapter 4

Dynamic movement primitives with shared synergies

Contents

A salient feature of human motor skill learning is the ability to exploit similarities across related tasks. In biological motor control, it has been hypothesized that muscle synergies, coherent activations of groups of muscles, allow for exploiting shared knowledge. Recent studies have shown that a rich set of complex motor skills can be generated by a combination of a small number of muscle synergies. In robotics, dynamic movement primitives are commonly used for motor skill learning. This machine learning approach implements a stable attractor system that facilitates learning and it can be used in high-dimensional continuous spaces. However, it does not allow for reusing shared knowledge, i.e., for each task an individual set of parameters has to be learned. We propose a novel movement primitive representation that employs parametrized basis functions, which combines the benefits of muscle synergies and dynamic movement primitives. For each task a superposition of synergies modulates a stable attractor system. This approach leads to a compact representation of multiple motor skills and at the same time enables efficient learning in high-dimensional continuous systems. The movement representation

supports discrete and rhythmic movements and in particular includes the dynamic movement primitive approach as a special case. We demonstrate the feasibility of the movement representation in three multi-task learning simulated scenarios. First, the characteristics of the proposed representation are illustrated in a point-mass task. Second, in complex humanoid walking experiments, multiple walking patterns with different step heights are learned robustly and efficiently. Finally, in a multi-directional reaching task simulated with a musculoskeletal model of the human arm, we show how the proposed movement primitives can be used to learn appropriate muscle excitation patterns and to generalize effectively to new reaching skills.

4.1 Introduction

Reinforcement Learning of motor skills in robotics is considered to be very challenging due to the high-dimensional continuous state and action spaces. In many studies it has been shown that learning can be facilitated by the use of movement primitives [Schaal et al., 2003, Rückert et al., 2013]. Movement primitives are parametrized representations of elementary movements, where typically for each motor skill a small set of parameters is tuned or learned. However, many motor control tasks are related and could be learned more effectively by exploiting shared knowledge.

This concept is well known in motor neuroscience, where muscle synergies or coherent activations of groups of muscles [d'Avella et al., 2003, d'Avella and Bizzi, 2005, Bizzi et al., 2008] have been proposed to simplify the control problem of complex musculoskeletal systems. In analyzing muscle activation recordings it has been demonstrated that by combining only few muscle activation patterns multiple task instances of natural motor behaviors, e.g., fast reaching movements of humans [d'Avella et al., 2006], primate grasping movements [Overduin et al., 2008], or walking patterns of infants, toddlers, and adults [Dominici et al., 2011] could be efficiently modeled. One important finding of these studies is that the dimensionality of the motor control problem can be drastically reduced by reusing common knowledge of related tasks, i.e., grasping objects at different locations using a linear combination of shared muscle synergies. While this has been demonstrated in biological data analysis, only few robotic applications exist that use this shared task knowledge [Chhabra and Jacobs, 2006, Alessandro et al., 2012]. These methods demonstrate the advantages of shared synergies in learning robotic tasks. However, different procedures were applied to obtain a parametric description of synergies, i.e., in Chhabra and Jacobs [2006] a variant of non-negative matrix factorization [d'Avella et al., 2003] was used given a set of pre-computed trajectories and in Alessandro et al. [2012] the synergies were extracted from dynamic responses of a robot system with random initialization. In contrast, we propose to learn the synergies representation in a reinforcement learning framework, where task-specific and task-invariant parameters in a multi-task learning setting are learned simultaneously.

In robotics the most widely used approach for motor skill learning are Dynamic Movement Primitives (DMPs) [Schaal et al., 2003, Ijspeert et al., 2013]. This approach uses parametrized dynamical systems to determine a movement trajectory and has several benefits. First, as it is a model-free approach, there is no need to learn the typically nonlinear, high-dimensional dynamic forward model of a robot (However, the model is required when inverse dynamics controller are used to compute the control commands). Second, it provides a linear policy parametrization which can be used for imitation learning and policy search [Kober and Peters, 2011]. The complexity of the trajectory can be scaled by the number of parameters [Schaal et al., 2003] and one can adapt meta-parameters of the movement such as the movement speed or the goal state [Pastor et al., 2009, Kober et al., 2010]. Finally, the dynamical system is constructed such that the system is stable. This simplifies learning since even without modulating the dynamical system the movement trajectory is always attracted by a known (or learned) goal state. However, this parametrization does not allow for reusing shared knowledge, as proposed by the experimental findings studying complex musculoskeletal systems [d'Avella et al., 2003, Bizzi et al., 2008, d'Avella and Pai, 2010]. Thus, typically for each motor task an individual movement parametrization has to be learned.

In this paper we propose to use a superposition of learned basis functions or synergies to modulate the stable attractor system of DMPs. This allows for reusing shared knowledge for learning multiple related tasks simultaneously while preserving the benefits of the dynamical systems, i.e., the stability in learning complex motor behavior. The synergies and their activation in time are learned from scratch in a standard reinforcement learning setup. Note that imitation learning could also be applied to implement an initial guess for the synergies, e.g., by using decomposition strategies discussed in d'Avella and Tresch [2001]. However, an application of such decomposition strategies is beyond the scope of this paper. Moreover, our approach is like the DMPs applicable to discrete and rhythmic movements and allows for modeling time-varying synergies [d'Avella et al., 2006]. We therefore denote our approach *DMPSynergies*. By using for each task a combination of individual, temporally fixed, basis functions DMPs can be modeled as special case of this approach. The benefit of the common prior knowledge is even more drastic when generalizing to new motor tasks given the previously learned basis functions. Thus, for simpler synergies only the weights for the linear combination have to be acquired and for time-varying synergies additionally the time-shift parameters need to be learned. This benefit is demonstrated on a complex walking task and on reaching task using an arm actuated by muscles.

As in previous studies on DMPs [Meier et al., 2011, Mülling et al., 2013] we want to go beyond basic motor skills learning. However, in contrast to those studies that use a library of primitives for sequencing elementary movements [Meier et al., 2011] or mixing basic skills [Mülling et al., 2013], we implement the common shared knowledge among multiple tasks as prior in a hierarchical structure. On the lower level task related parameters, i.e., amplitude scaling weights or time-shift parameters are used to modulate a linear superposition of learned basis functions, the shared higher level knowledge. This has the promising feature that by combining just a small number of synergies diverse motor skills can be generated.

In the Materials and Methods, we will first briefly introduce DMPs [Schaal et al., 2003, Ijspeert et al., 2013] as we build on this approach. We then extend DMPs to allow for reusing shared task knowledge in the form of parametrized synergies. The advantage of the shared knowledge is evaluated in the Results on three multi-task learning scenarios. First, a simple via-point task is used to demonstrate the characteristics of the proposed representation. Then, rhythmic movements are learned in a dynamic 5-link planar biped walker environment. Finally, a musculoskeletal model of a human arm is used to evaluate our primitives on a muscle actuated system learning discrete reaching movements to multiple targets.

4.2 Materials and methods

4.2.1 Dynamic movement primitives

DMPs generate multi-dimensional trajectories by the use of nonlinear differential equations (simple damped spring models) [Schaal et al., 2003]. The basic idea is to use for each degree-of-freedom (DoF), or more precisely for each actuator, a globally stable, linear dynamical system of the form

$$\tau \dot{z} = \alpha_z(\beta_z(g - y^*) - z) + f, \qquad \tau \dot{y}^* = z, \tag{4.1}$$

which is modulated by a learnable nonlinear function f. The final position of a movement is denoted by g and the variables y and $\dot{y}$ represent the desired state in i.e., joint angles and joint velocities. The time constants α and β are usually pre-defined. The temporal scaling factor τ can be used for de- or accelerating the movement execution as needed. Finally z denotes an internal variable of the dynamical system. For each DoF an individual function f is used which is different for discrete and rhythmic movements.

For discrete movements the function f only depends on the phase s, which is an abstraction of time and was introduced to scale the movement duration [Schaal et al., 2003]. The function $f(s)$ is constructed of the weighted sum of N Gaussian basis functions Ψ_n

$$f(s) = \frac{\sum_{n=1}^{N} \Psi_n(s) w_n s}{\sum_{n'=1}^{N} \Psi_{n'}(s)}, \tag{4.2}$$

where for discrete movements these Gaussian basis functions are

$$\Psi_n(s) = \exp(-\frac{1}{2h_n^2}(s-\mu_n)^2), \qquad \tau\dot{s} = -\alpha_s s.$$

Only the weights w_n are parameters of the primitive which can modulate the shape of the movement. The centers or means $\mu_n \in [0,1]$ specify at which phase of the movement the basis function becomes active. They are typically equally spaced in the range of s and not modified during learning. The bandwidth of the basis functions is given by h_n^2 and is typically chosen such that the Gaussians overlap.

For rhythmic movements periodic activation functions are used [Ijspeert and Schaal, 2003]. The nonlinear function f reads

$$f(\phi) = \frac{\sum_{n=1}^{N} \Psi_n(\phi) w_n}{\sum_{n'=1}^{N} \Psi_{n'}(\phi)}, \tag{4.3}$$

where the periodic phase angle is denoted by $\phi \in [0, 2\pi]$. In Ijspeert and Schaal [2003] additionally a scalar variable was used to scale the amplitude of the oscillator, which was omitted for simplicity. The basis functions are given by

$$\Psi_n(\phi) = \exp(h_n(\cos(\phi - \mu_n) - 1)), \qquad \tau\dot{\phi} = 1,$$

which implement von Mises basis functions. Note that for the periodic basis functions the trajectory in Equation 4.1 oscillates around the attractor point or goal state g.

Integrating the dynamical systems in Equation 4.1 for each DoF results into a desired trajectory $\langle \mathbf{y}_t^*, \dot{\mathbf{y}}_t^* \rangle$ of the joint angles. To follow this trajectory, in the most simple case a linear feedback controller is subsequently used to generate appropriate control commands denoted by $\mathbf{u}_t$:

$$\mathbf{u}_t = \mathrm{diag}(\mathbf{k}_{pos})(\mathbf{y}_t^* - \mathbf{y}_t) + \mathrm{diag}(\mathbf{k}_{vel})(\dot{\mathbf{y}}_t^* - \dot{\mathbf{y}}_t). \tag{4.4}$$

For each actuator the linear weights $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_D]$ as well as the control gains $\mathbf{k}_{pos}$ and $\mathbf{k}_{vel}$ have to be specified, i.e. $\boldsymbol{\theta} = [\mathbf{W}, \mathbf{k}_{pos}, \mathbf{k}_{vel}]$. This results into $ND + 2D$ parameters for the movement representation, where D denotes the number of actuators or muscles of a system. The simulated trajectory is denoted by $\langle \mathbf{y}_t, \dot{\mathbf{y}}_t \rangle$.

In multi-task learning we want to learn $k = 1..K$ tasks simultaneously. For very simply tasks, such as the via-point experiments described below, it could be sufficient to adapt the goal state g. However, this simple adaptation approach is likely to fail for more complex motor skill learning tasks in robotics. With DMPs typically for each motor skill an individual movement parametrization $\boldsymbol{\theta}_k$ has to be learned. However, if we assume similarities among these tasks the learning problem could potentially be simplified by reusing shared knowledge. Inspired by experimental findings in biology [d'Avella et al., 2003, Bizzi et al., 2008, d'Avella and Pai, 2010] we extend these DMPs. Only the parametrization for the nonlinear function $f(s)$ for discrete movements or $f(\phi)$ for rhythmic movement changes. The dynamical system in Equation 4.1 and the linear feedback controller in Equation 4.4 remains the same.

4.2.2 Dynamic movement primitives with shared synergies (DMPSynergies)

For learning the kth task, we propose to use a linear combination of temporal flexible basis functions or synergies to parametrize the nonlinear function $f(s)$ in Equation 4.2 or for rhythmic movements $f(\phi)$ in Equation 4.3:

$$f(s,k) = \textstyle\sum_{m=1}^{M} \beta_{m,k} \Lambda(s, \boldsymbol{\theta}_m, \Delta s_{m,k}) s, \tag{4.5}$$

$$f(\phi,k) = \textstyle\sum_{m=1}^{M} \beta_{m,k} \Omega(\phi, \boldsymbol{\theta}_m, \Delta s_{m,k}), \tag{4.6}$$

where s denotes the phase variable which is only used for discrete movements. As with DMPs (Ψ_n in Equation 4.2) are the functions $\Lambda(.)$ and $\Omega(.)$ different for discrete and rhythmic movements.

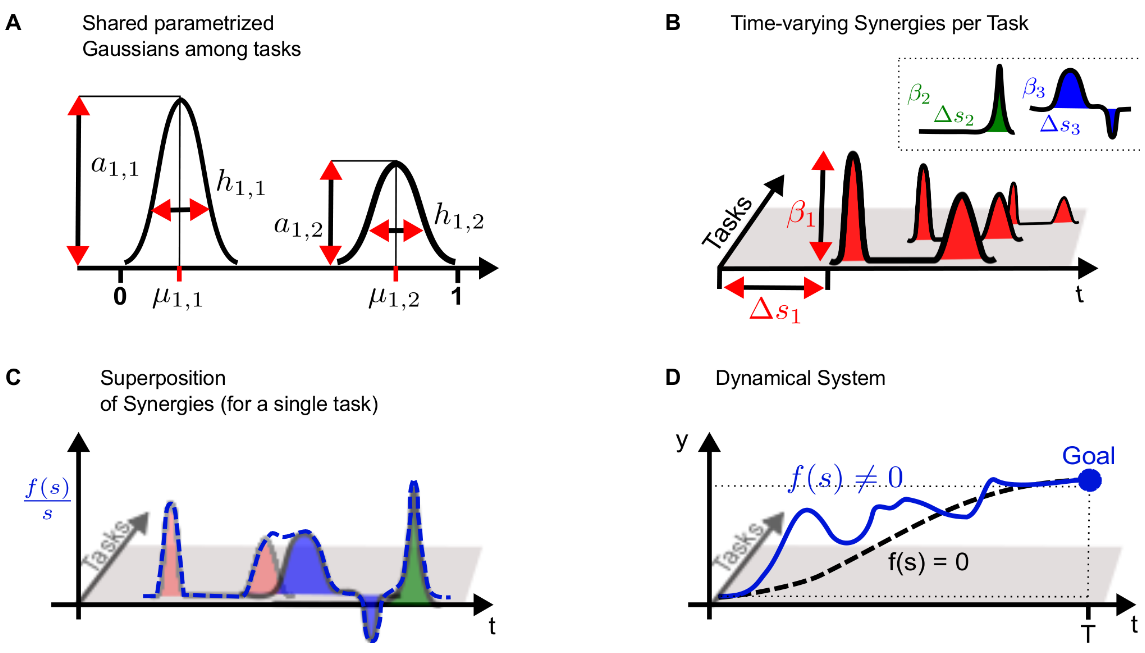


Figure 4.1: **Conceptual idea of using shared synergies in dynamical systems.** (A) A synergy is constructed by a superposition of parametrized Gaussians. The parameters are the amplitude $a_{m,n}$, the mean $\mu_{m,n}$ and the bandwidth $h_{m,n}$. In the example two Gaussians ($n = 1..2$) are used to model the first $m = 1$ synergy. (B) For each task only the activation β_m of a synergy is learned. Time-varying synergies additionally implement a time-shift Δs_m. The key concept is that multiple tasks share the same parametrized synergies shown in (A), which represent task related common knowledge. (C) For each task the nonlinear function $f(s)$ is given by the weighted sum of the (time-shifted) synergies. Shown is a normalized version of $f(s)$ to illustrate the effects of the superposition also at the end of the movement, which would usually converge towards zero. (D) Finally, the nonlinear function $f(s)$ is used to modulate a dynamical system. The unperturbed system with $f(s) = 0$ is denoted by the dashed line which is *attracted* by the goal state that is indicated by the large dot.

All K tasks share $m = 1..M$ synergies which are parametrized via the vector $\boldsymbol{\theta}_m$. Solely the weights $\beta_{m,k}$ and the time-shift $\Delta s_{m,k}$ are individual parameters for each task. The basic concept of the model is sketched in Figure 4.1 for a one-dimensional discrete movement.

The complexity of each synergy is controlled by the number of Gaussians for discrete movements or by the number of von Mises basis functions for rhythmic patterns. We denote this number by N, where we parametrize in both cases the amplitude, the mean and the bandwidth. Thus, each synergy is represented by a parameter vector $\boldsymbol{\theta}_m = [a_{m,1}, \mu_{m,1}, h_{m,1}, ..., a_{m,N}, \mu_{m,N}, h_{m,N}]$.

For discrete movements the function $\Lambda(.)$ reads

$$\Lambda(s, \boldsymbol{\theta}_m, \Delta s_{m,k}) = \sum_{n=1}^{N} a_{m,n} \exp(-\frac{1}{2{h_{m,n}}^2}(s - \mu_{m,n} + \Delta s_{m,k})^2). \tag{4.7}$$

For rhythmic movements a superposition of von Mises basis functions is used

$$\Omega(\phi, \boldsymbol{\theta}_m, \Delta s_{m,k}) = \sum_{n=1}^{N} a_{m,n} \exp(h_{m,n} \cos(\phi - \mu_{m,n} + \Delta s_{m,k}) - 1). \tag{4.8}$$

DMPs [Schaal et al., 2003] can be modeled as a special case of this formulation. For DMPs using $n = 1..N$ basis functions the mean $\mu_{m,n}$ and the bandwidth $h_{m,n}$ of the basis functions are fixed as discussed in Section 4.2.1. Solely the $n = 1..N$ amplitudes or weights $a_{m,n}$ are learned. By fixing these parameters and by modeling the nonlinear function $f(s)$ for discrete movements or $f(\phi)$ for rhythmic movements using a single ($M = 1$) synergy our representation can be used to implement DMPs.

4.2.2.1 Multi-dimensional systems

For multi-dimensional systems for each actuator $d = 1..D$ an individual dynamical system in Equation 4.1 and hence an individual function $f(s, k)$ in Equation 4.5 or $f(\phi, k)$ in Equation 4.6 is used [Schaal et al., 2003]. The phase variable s or ϕ is shared among all DoF (Note that $k = 1..K$ denotes the task.).

Extending our notation for multi-dimensional systems the nonlinear function $f(s, k)$ in Equation 4.5 can be written as

$$\underbrace{f(s, d, k)}_{1x1} = \sum_{m=1}^{M} \underbrace{\beta_{m,k,d}}_{1x1} \underbrace{\Lambda(s, \boldsymbol{\theta}_{m,d}, \Delta s_{m,k,d}) s}_{1x1}.$$

Depending on the dimension d different weights $\beta_{m,k,d}$, policy vectors $\boldsymbol{\theta}_{m,d}$ and time-shift parameters $\Delta s_{m,k,d}$ are used. Note that the policy vector $\boldsymbol{\theta}_{m,d}$ is task-independent. Interestingly, when implementing additionally dimension-independent policy vectors, i.e. $\boldsymbol{\theta}_m$ anechoic mixing coefficients [Giese et al., 2009] can be modeled.

Here, we only discuss discrete movement representations, however, the reformulation procedure applies also for rhythmic movement parametrizations. Let us also define a vector notation of $\mathbf{f}(s,k)$

$$\underbrace{\mathbf{f}(s,k)}_{\text{1xD}} = \sum_{m=1}^{M} \underbrace{\boldsymbol{\beta}_{m,k}}_{\text{1xD}} \circ \underbrace{\mathbf{w}_m(s,\boldsymbol{\theta}_{m,1..D},\Delta s_{m,k,d})}_{\text{1xD}}, \tag{4.9}$$

where the symbol $\circ$ denotes the Hadamard product, the element-wise multiplication of vectors. The synergy vectors are specified by

$$\mathbf{w}_m(s,\boldsymbol{\theta}_{m,1..D},\Delta s_{m,k,d}) = [\Lambda(s,\boldsymbol{\theta}_{m,1},\Delta s_{m,k,1})s, \Lambda(s,\boldsymbol{\theta}_{m,2},\Delta s_{m,k,2})s, ..., \Lambda(s,\boldsymbol{\theta}_{m,D},\Delta s_{m,k,D})s].$$

This vector notation is used in the following to compare to existing synergies representations [d'Avella et al., 2003, 2006].

4.2.3 Musculoskeletal models and muscle synergies

We also use the proposed movement representation, the DMPSynergies, to generate muscle excitation patters. These patterns are applied as input in a forward simulation of a musculoskeletal model. A schematic overview of such a simulation is shown in Figure 4.2. We briefly discuss all processes involved.

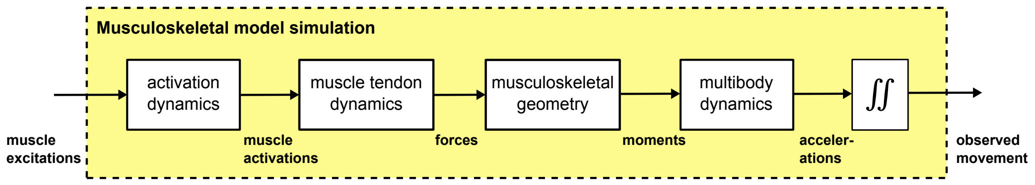


Figure 4.2: **Forward simulation of musculoskeletal models.** Muscle excitation patterns are used as input, which result in a delayed muscle activity response (activation dynamics). Muscle forces are the result of simulated muscle tendon dynamics, which are typically approximated by a Hill-type contractile element in series with tendon. These muscle forces are used to compute moments of force considering the musculoskeletal geometry. A physics engine is finally used to simulate multibody dynamics which are numerically integrated to generate movement trajectories.

Muscle synergies for generating muscle excitation patterns are used as input in forward dynamics simulations. In our simulation experiments we evaluate time-varying synergies [d'Avella et al., 2006], which are a particular instance of the DMPSynergies, i.e., the weights $\beta_{m,k}$ and time-shift parameters $\Delta s_{m,k}$ in Equation 4.9 are independent of the dimension d. Thus, for discrete movements in multi-dimensional systems $f(s,k)$ reads

$$\underbrace{\mathbf{f}(s,k)}_{\text{1xD}} = \sum_{m=1}^{M} \underbrace{\beta_{m,k}}_{\text{1x1}} \underbrace{\mathbf{w}_m(s + \Delta s_{m,k}, \boldsymbol{\theta}_{m,1..D}, 0)}_{\text{1xD}}, \tag{4.10}$$

where $\beta_{m,k}$ is a scalar and the time-shift parameter $\Delta s_{m,k}$ is directly added to the phase variable s. This allows for a comparison to e.g., the formulation of time-varying synergies given in d'Avella et al. [2006], where

$$\mathbf{x}(t,k) = \sum_{m=1}^{M} a_m^k \mathbf{v}_m(t - t_m^k).$$

Shared synergies are represented by time-dependent vectors $\mathbf{v}_m(t - t_m^k)$, where in contrast to the proposed DMPSynergies a minor difference is the sign of the time-shift parameter t_m^k.

In this formulation of time-varying synergies [d'Avella et al., 2006] only the time-invariant combination coefficients a_m^k are task-dependent, whereas the vector $\mathbf{v}_m$ is task-independent. However, by using task, spatial or temporal (in)variant implementations of the mixing coefficients a or the basis vectors $\mathbf{v}$ other representations of synergies [Ivanenko et al., 2004, d'Avella et al., 2003, Giese et al., 2009] can be also implemented.

Activation dynamics model the effect of the delayed force generating process in muscles, as they are not capable of generating force instantaneously. Typically, for each muscle a first order differential equation is used, i.e., $\dot{a} = (f(s,k)^2 - f(s,k)a)/\tau_{\text{rise}} + (f(s,k) - a)/\tau_{\text{fall}}$ [Zajac, 1989]. Here, $f(s,k)$ denotes the generated muscle excitation signal using e.g., the proposed DMPSynergies. The actual muscle activation is denoted by a, which is a function of the rise time constant τ_{rise} and the fall time constant τ_{fall}. For our evaluations we implemented $\tau_{\text{rise}} = 10\text{ms}$ and $\tau_{\text{fall}} = 40\text{ms}$ [Winters and Stark, 1985].

Muscle tendon dynamics describe the complex and nonlinear force generation properties of muscles. For an approximation a variety of models exist [Zajac, 1989]. In these models a muscle is approximated by a number of musculotendinous units, each of which is implemented by a Hill-type contractile element in series with tendon. Characteristic properties of muscles are the optimal fiber length L_0^M, the maximum isometric force F_0^M, and the muscle pennation angle α, which are shown in Table D.5 in the appendix for the investigated model of a human arm. The tendon dynamics in this musculoskeletal model were approximated by the muscle model proposed in Schutte et al. [1993].

Musculoskeletal geometry represents the path of a muscle from its origin to its insertion that can be implemented as a series of straight-line path segments, which pass through a series of via points [Delp et al., 2007]. To simulate how muscles wrap over underlying bone and musculature wrapping surfaces i.e., cylinders, spheres and ellipsoids are implemented, where this model is based on the upper extremity model discussed in Holzbaur et al. [2005]. A detailed description of the implemented musculoskeletal geometry is given in the supplement (in form of a simulation model file, .osim).

Multibody dynamics are simulated by the physics simulation application OpenSim [Delp et al., 2007, Seth et al., 2011]. It is an open source software that already implements a variety of muscle models [Zajac, 1989] and a large number musculoskeletal models are freely available. In our experiments the computational time needed to simulate a movement with a duration of e.g., 500ms takes between 10 and 20 seconds (OpenSim implements numerical integrators with an adaptive time step) on a standard computer (3GHz and 4GB memory). However, we exploited parallel computing techniques for policy search, which resulted in a gain of factor 10. Alternatively, the muscle dynamics could be approximated via regression methods to speed-up the simulations [Chadwick et al., 2009].

4.2.4 Learning with movement primitives

We denote the parametrization of a movement primitive by a policy vector $\boldsymbol{\theta}$. A widely used approach in robotics to learn these parameters is episodic reinforcement learning [Kober and Peters, 2011], which is outlined in Figure 4.3 (A). A policy search method is used to improve the movement primitive's representation $\boldsymbol{\theta}$ assuming a given objective or reward function $C(\tau) \in \mathbb{R}^1$. Throughout this manuscript $C(\tau)$ denotes a cost value that is equivalent to the negative reward in classical reinforcement learning [Sutton and Barto, 1998]. It indicates the quality of an executed movement. A trajectory $\tau = \langle \mathbf{y}_{1:T}, \mathbf{u}_{1:T-1} \rangle$ is specified by the simulated joint angles $\mathbf{y}$ and the applied controls (torques) $\mathbf{u}$, where T denotes the number of time steps. We want to find a movement primitive's parameter vector $\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ which minimizes the expected costs $J(\boldsymbol{\theta}) = \mathbb{E}\left[C(\tau)|\boldsymbol{\theta}\right]$. We assume that we can evaluate the expected costs $J(\boldsymbol{\theta})$ for a given parameter vector $\boldsymbol{\theta}$ by performing roll-outs (samples) on the real or simulated system. In other words each movement trajectory is quantified by a single scalar reward $C(\tau)$, which can be used by an optimization method to improve the best guess of the movement policy $\boldsymbol{\theta}$.

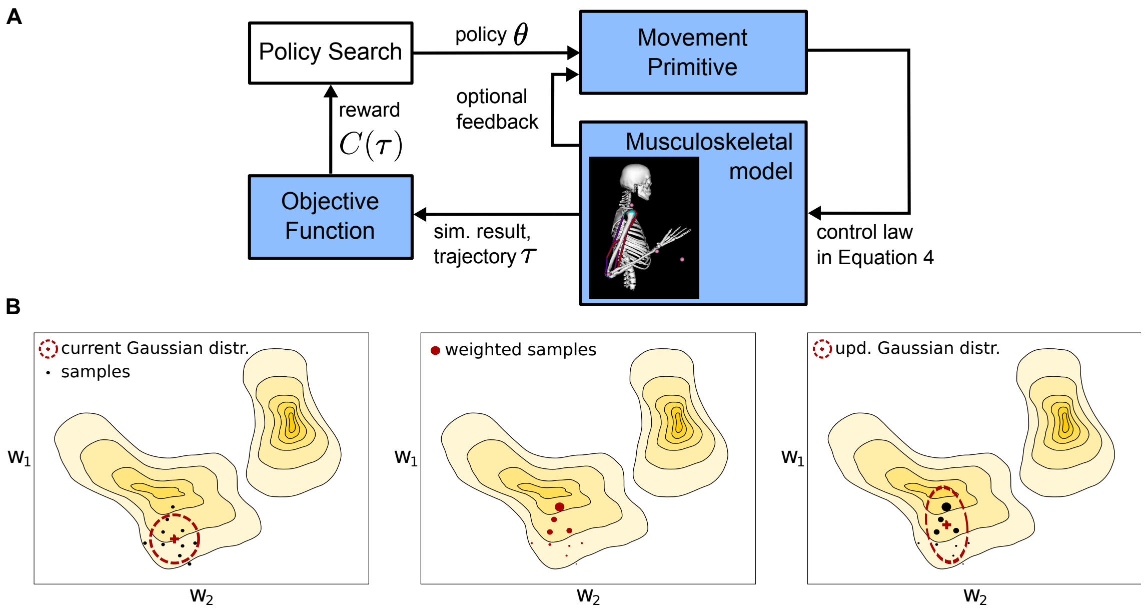


Figure 4.3: **Overview of the learning framework.** (A) A parametrized policy $\boldsymbol{\theta}$ modulates the output of a movement primitive that is used to generate a movement trajectory τ. The quality of the movement trajectory is indicated by a sparse reward signal $C(\tau)$ which is used for policy search to improve the parameters of the movement primitive. For a single iteration the implemented policy search method - Covariance Matrix Adaptation (CMA) [Hansen et al., 2003] is sketched in (B). The parameter space is approximated using a multivariate Gaussian distribution denoted by the ellipses, which is updated (from left to right) using second order statistics of roll-outs or samples that are denoted by the large dots (see text for details).

For learning or optimizing the policy parameters $\boldsymbol{\theta}$ a variety of policy search algorithms exist in the motor control literature. Examples are the REINFORCE [Williams, 1992], the episodic Natural Actor Critic [Peters and Schaal, 2008], the PoWER [Kober and Peters, 2011] or the PI^2 [Theodorou et al., 2010] algorithm, which are reviewed in Kober and Peters [2011]. Alternatively, standard optimization tools such as the 2nd order stochastic search methods [Hansen et al., 2003, Wierstra et al., 2008, Sehnke et al., 2010] can be used for policy search. These machine learning tools make no assumptions on a specific form of a policy and typically have just a single parameter to tune, the initial exploration rate. We therefore use the stochastic search method Covariance Matrix Adaptation (CMA) [Hansen et al., 2003] for learning the policy parameters in our experiments.

Roughly, CMA is an iterative procedure that locally approximates the function $C(\tau(\boldsymbol{\theta}))$ by a multivariate Gaussian distribution, which is denoted by the ellipse in the sketch in Figure 4.3 (B). From left to right a single optimization step for a two-dimensional policy vector $\boldsymbol{\theta} = [w_1, w_2]$ is shown. The colored regions denote the unknown optimization landscape, where solid lines depict equal $C(\tau)$ values. From the current Gaussian distribution, denoted by the ellipse in the left panel, CMA generates a number of samples, denoted by the black dots, evaluates the samples (the size of the dots in the center panel is proportional to their $C(\tau)$ values), computes second order statistics of those samples that reduced $C(\tau)$ and uses these to update the Gaussian search distribution, which is shown in right panel. For algorithmic details we refer to Hansen et al. [2003].

Note that for most interesting robotic tasks the unknown optimization landscape that is also sketched in Figure 4.3 (B) is multi-modal and policy search might converge to a local optimum. Thus, the result of learning is sensitive to the initial policy parameters $\boldsymbol{\theta}$ and for evaluating the convergence rate of different policy search methods multiple initial configurations should be considered [Kober and Peters, 2011]. However, in this manuscript we evaluate the characteristics of movement primitive representations and put less emphasis on a particular policy search method. As we will demonstrate in our experiments CMA is robust in terms of converging to “good” solutions given the initial values of the evaluated movement primitive representations listed in the appendix.

In our experiments we compare single task learning with DMPs to learning multiple tasks simultaneously with DMPSynergies. With DMPs for each task $k = 1..K$ an individual policy vector $\boldsymbol{\theta}_k$ is learned, where the objective function used in policy search takes the task index as additional argument, i.e., $C(\tau, k)$. For learning multiple tasks simultaneously with DMPSynergies the policy vector $\boldsymbol{\theta}$ encodes all K task specific parameters $\beta_{m,k}$ and $\Delta s_{m,k}$, and all shared parameters denoted by θ_m in Equation 4.5 or Equation 4.6. The objective function is the sum of the individual task dependent costs $C(\tau) = \sum_{k=1}^{K} C(\tau, k)$.

4.3 Results

We evaluated the proposed movement representation, the DMPSynergies, with simulations using three multi-task learning scenarios. A simple via-point task was used to illustrate the characteristics of the proposed movement representation. A challenging robotic learning task was used to generate rhythmic walking patterns for multiple step heights. Discrete reaching movements were learned using a musculoskeletal model of a human arm with eleven muscles.

4.3.1 Via-point reaching task with a simple toy model

The goal of this simple multi-task learning problem is to pass through $k = 1..5$ via-points ($\mathrm{vp}_k \in \{0.2, 0.1, 0, -0.1, -0.2\}$), denoted by the large dots in Figure 4.4 (A) and navigate to the goal state g at 1. We used a point mass system (1kg), where the state at time t is given by the position y_t and the velocity $\dot{y}_t$. The applied controls u_t shown in Figure 4.4 (B) are computed using the linear feedback control law with $k_{pos} = 400$ and $k_{vel} = 15$ specified in Equation 4.4. The finite time horizon is given by $T = 50$. For the dynamical system in Equation 4.1 we used the parameters $\alpha_z = 2, \beta_z = 0.9$ and $\tau = 0.1$. Further parameter settings used for policy search are summarized in Table D.1 in the appendix.

This task is specified by the objective function

$$C(k) = 10^5 (y_{t_{\mathrm{vp}_k}} - \mathrm{vp}_k)^2 + 10^4 (\dot{y}_T^2 + 10(y_T - g)^2) + 5 \cdot 10^{-4} \sum_{t=1}^{T} u_t.$$

The first two terms punish deviations from the via-point vp_k and the goal state g, where $y_{t_{\mathrm{vp}_k}}$ denotes the position of the state at the time index of the kth via-point. The last term punishes high energy consumption, where u_t denotes the applied acceleration. Note that for simplicity we did not introduce the variable τ denoting the movement trajectory in $C(\tau, k)$ in Subsection 4.2.4. We always add a Gaussian noise term with a standard deviation of $\sigma = 0.5$ to the control action to simulate motor noise.

We used a single synergy ($M = 1$) with $N = 2$ Gaussians to model the shared prior knowledge. The learning curve is shown in Figure 4.4 (C), where we compare to single-task learning using DMPs with $N = 8$ Gaussian basis functions. For the via-point task 8 Gaussians were optimal with respect to the convergence rate, where we evaluated representations using $N = 2..20$ Gaussians (not shown). Additionally, we compare to an incremental learning setup (DMP inc.) in Figure 4.4 (C), where the DMP representation is always initialized with the best learned solution from the previous task. On the x-axis the number of samples or trajectory evaluations on the point mass system is plotted. As it can be seen the proposed approach can benefit from the shared knowledge and has a faster overall learning performance.

In this experiment, for each task we fixed the time-shift $\Delta s_k = 0$ and only learned the $k = 1..5$ weights β_k in Equation 4.5 (Note that the synergy index m was omitted as only a single synergy was used). For each of the $N = 2$ Gaussians we learned the mean μ, the bandwidth h and the amplitude a in Equation 4.7. Thus, in total $5 + 2 \times 3 = 11$ parameters were learned. In contrast with DMPs 8 Gaussian amplitudes were optimized.

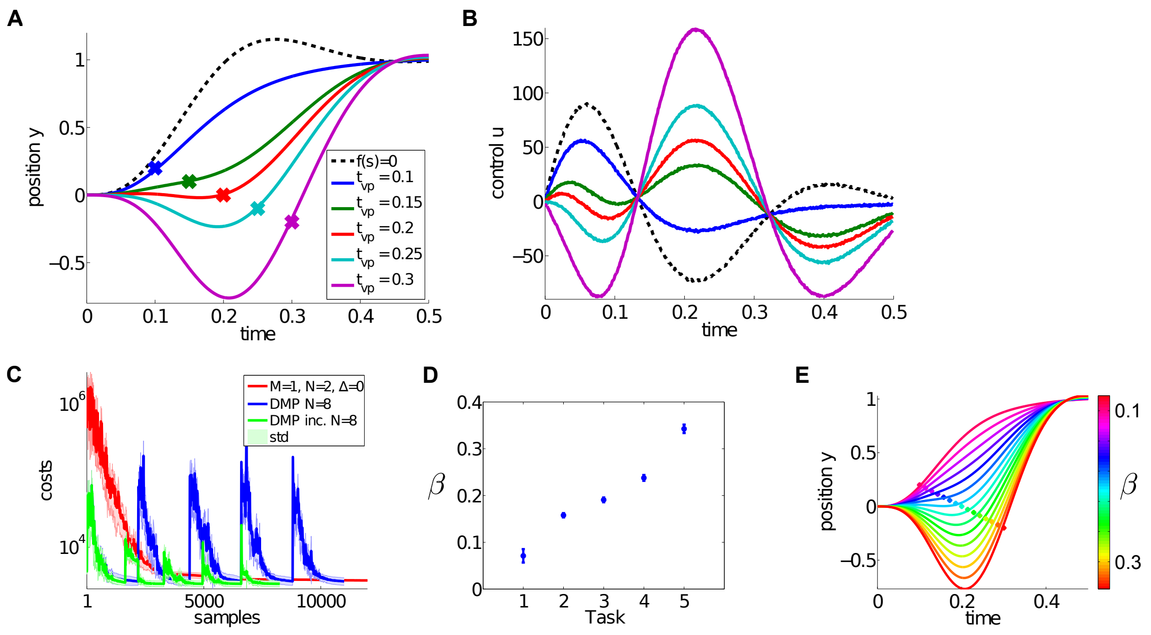


Figure 4.4: **Results for the dynamic via-point task.** The goal of this simple multi-task learning problem is to pass through five via-points, denoted by the large dots in (A) and navigate to the target state at 1. The corresponding controls (accelerations) of this dynamical system are shown in (B). These five trajectories are simultaneously learned using DMPSynergies with a single synergy ($M = 1$) represented by $N = 2$ Gaussians. We compare to dynamic movement primitives (DMPs) with $N = 8$ Gaussians and to an incremental variant of DMPs in (C). For the DMP approaches each task (via-point) has to be learned separately. Thus, the two learning curves have five peaks. In contrast with DMPSynergies we could learn these five tasks at once, which resulted in faster overall convergence. The plot in (D) illustrates the mean and the standard deviation of the learned β values for the DMPSynergy approach. Via interpolating β and by reusing the learned synergy new motor skills can be generated without re-learning. This generalization feature is illustrated in (E), where $\beta \in [0.07, 0.34]$.

The β values of the DMPSynergies representation for the five via-points are shown in Figure 4.4 (D) for 10 runs. New motor skills can be generated without re-learning via a simple linear interpolation. The resulting trajectories are shown in Figure 4.4 (E). However, this simple interpolation approach could only be applied in the simple via-point task. For the more complex tasks discussed in this manuscript these β values have to be learned.

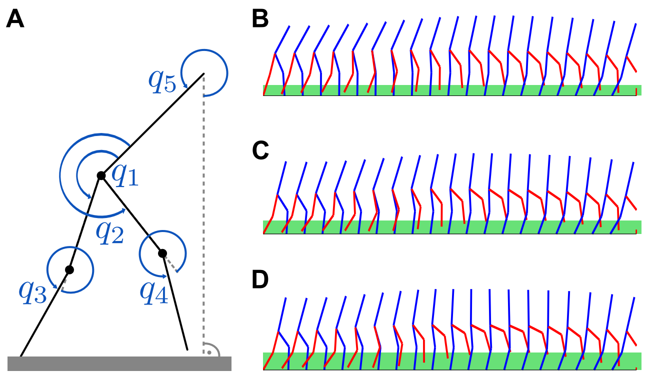


Figure 4.5: **Dynamic biped walker model.** (A) For the walker model, only the hip angles $\mathrm{q}_1, \mathrm{q}_2$ and the knee angles $\mathrm{q}_3, \mathrm{q}_4$ are actuated. The reference angle to the flat ground is denoted by q_5. In this multi-task learning experiment we want to learn walking patterns for different step heights. Examples for step heights of $0.15, 0.25$ and 0.3m for a single step are shown in (B-D). These patterns were learned using the proposed movement primitives with shared synergies ($M = 2$ and $N = 3$). The green bars in (B-D) denote the true (maximum) step heights, which are $0.19, 0.24$ and 0.31m.

4.3.2 Dynamic biped walker task

To evaluate the DMPSynergies on a multi-dimensional robotic task we learned multiple walking patterns using a 5 degree-of-freedom (DoF) dynamic biped robot model, which is shown in Figure 4.5 (A). We demonstrate that by exploiting the shared knowledge among multiple walking gaits, solutions could be found more robustly and more efficiently in terms of learning speed compared to single task learning. Further, the shared synergies could be used to generalize new skills. The model is only as complex as required to study difficulties like limb coordination, effective underactuation, hybrid dynamics or static instabilities. More details on the design and challenges can be found in Westervelt et al. [2004].

The 10-dimensional state $\mathbf{q}_t = [\mathrm{q}_{1:5}, \dot{\mathrm{q}}_{1:5}]$ of the robot is given by the hip angles (q_1 and q_2), the knee angles (q_3 and q_4), a reference angle to the ground (q_5), and the corresponding velocities $\dot{\mathrm{q}}_{1:5}$. Only the hip and the knee angles are actuated. Thus, 4 dynamical systems in Equation 4.1 are used to generate desired trajectories for the linear feedback controller in Equation 4.4. A phase resetting strategy is implemented to facilitate learning [Nakanishi et al., 2004]. At each impact of the swing leg the phase ϕ in Equation 4.8 is set to zero. This increases the stability of the robot as the gait cycle duration is implicitly given by the impact time.

The initial state $\mathbf{q}_1 \in \mathbb{R}^{10}$, the goal state $\mathbf{g}$ and the control gains in Equation 4.4 were optimized in advance for a desired step height of $r^* = 0.2$m to simplify learning.

The resulting values are shown in Table D.2 in the appendix. For rhythmic movements the goal state $\mathbf{g} \in \mathbb{R}^5$ models an attractor point which is only specified for joint angles and not for velocities in Equation 4.1. As for the via-point reaching task, Gaussian noise with $\sigma = 1$ is added to the simulated controls. For Equation 4.1 we used the parameters $\alpha_z = 2, \beta_z = 0.5$ and $\tau = 0.06$. The initial parameter values and the applied ranges used for policy search are shown in Table D.3 in the appendix.

In this multi-task learning experiment we want to learn walking patterns for different desired step heights, i.e., $r_k^* \in \{0.15, 0.2, 0.25, 0.3\}$m. Example patterns for step heights of $0.15, 0.25$ and 0.3m are shown in Figure 4.5 (B-D), where the green bars denote the maximum step heights during a single step $(0.19, 0.24$ and 0.31m).

The objective function for a single walking task is given by the distance travelled in the sagittal plane, the duration of the simulation and deviations from the desired step height r_k^* with $k = 1..4$:

$$C(k) = -0.6(x_T - x_1) + 0.2(5 - T \cdot \Delta t) + 50 \sum_{i=1}^{S} (r_i - r_k^*)^2, \tag{4.11}$$

where x denotes the x-coordinate of the hip, S the number of steps and r_i the maximal step height during the ith step. We used a time step $\Delta t = 2$ms. The time horizon $T \in [1, 5000]$ is given by the last valid state of the robot, where the biped does not violate the joint angle constraints specified by $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ in Table D.2 in the appendix.

With the proposed DMPSynergies the nonlinear function $f(\phi, k)$ in Equation 4.6 is generated by combining a set of learned synergies that are shared among multiple task instances, i.e. the four $(k = 1..4)$ desired step heights. This combination mechanism is illustrated for a representation using $M = 2$ synergies modeled by $N = 3$ Gaussians in Figure 4.6. For each actuator (left hip, right hip, left knee, and right knee) an individual function $f(\phi, k)$ is generated, which is subsequently used to modulate an attractor system shown in Equation 4.1 to compute the desired movement trajectories. The shared synergies shown in the last two rows in Figure 4.6 can be scaled and shifted in time. These shared synergies are indicated by the enclosing rectangles. Note that the color of the synergies is used to distinguish the four actuators of the walker model.

We evaluated different movement primitive representations with increasing complexity compared to single-task learning using DMPs with $N = 4$ and $N = 8$ Gaussians. The average final costs $\mathrm{C}_{\mathrm{mean}}$ after learning over 10 runs are shown in Table 4.1. In the most simple representation we used $M = 2$ synergies modeled by $N = 2$ Gaussians.

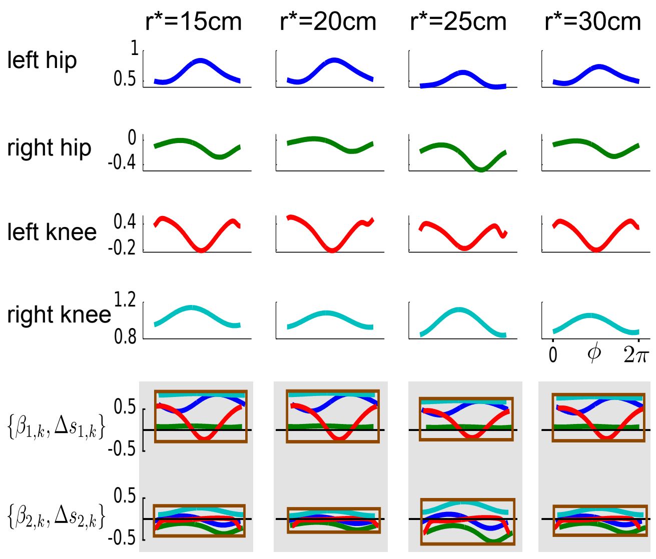


Figure 4.6: **Learned nonlinear functions $f(\phi, k)$ for the walker task.** The learned nonlinear functions $f(\phi, k)$ are illustrated in the first four rows. The four task instances, i.e., the desired step heights are shown from left to right. For each actuator (left hip, right hip, left knee, and right knee) an individual function $f(\phi, k)$ is used that is generated by combining two learned synergies shown in the last two rows. These synergies are shared among multiple task instances and can be scaled and shifted in time (via $\beta_{m,k}$ and $\Delta s_{m,k}$). These shared synergies are indicated by the enclosing rectangles, where the color of the synergies is used to distinguish the four actuators of the walker model.

More complex representations implementing time-varying synergies are denoted by the symbol $\Delta = 1$ in Table 4.1. Here, additionally the time-shifts $\Delta s_{1:M}$ were learned for all synergies and all actuators. However, the final learning performance did not outperform the representation with fixed time-shifts (i.e., $M = 2, N = 3$ and $\Delta = 0$: -21.4 ± 0.4 compared to $M = 2, N = 3$ and $\Delta = 1$: -20.5 ± 1.4). This can be also seen in Figure 4.7, where we plot the learning curve for synergies with $\Delta s_{1:M} = 0$ in Figure 4.7 (A) and the results for time-varying synergies in Figure 4.7 (B).

The average final cost value of the DMP representation is higher (i.e., DMP$_{N=8}$: -14.8 ± 1.4) compared to the best costs achieved with shared synergies ($M = 2, N = 3$ and $\Delta = 0$: -21.4 ± 0.4). This holds also for an incremental learning setup (e.g., DMP inc.$_{N=4}$: -19.2 ± 0.6), where DMPs were initialized with the best result from the previous task.

The joint angle trajectories of the left hip and knee joint for the DMPSynergy representation using $M = 2$ synergies modeled by $N = 3$ Gaussians and $\Delta = 1$ are illustrated in Figure 4.8. The average step heights were $r \in \{0.22, 0.22, 0.26, 0.28\}$, which do not match the desired step heights $r^* \in \{0.15, 0.2, 0.25, 0.3\}$. The reason for this mismatch is

Table 4.1: Achieved costs for the walker task, where the standard deviation is denoted by the symbol $\pm$. In the second column the symbol $\boldsymbol{\Delta}$ denotes if additionally $16M$ time-shift variables $\boldsymbol{\Delta s}$ are learned. The total number of parameters is denoted by the symbol #, where e.g., for the representation in the 1st row $4M = 8$ task related weights and $3 \cdot M \cdot N \cdot 4 = 48$ shared parameters were optimized. The best results with the lowest costs denoted by $\mathrm{C}_{\mathrm{mean}}$ are highlighted in gray shading.

Setup	$\boldsymbol{\Delta}$	$\mathbf{C}_{\mathbf{mean}}$	**best learned r** [m]	#
$M = 2$	0	-20.3 ± 1.2	$0.21, 0.21, 0.27, 0.27$	56
$N = 2$	1	-20.8 ± 0.9	$0.19, 0.23, 0.28, 0.26$	88
$M = 2$	0	-21.4 ± 0.4	$0.22, 0.24, 0.23, 0.26$	80
$N = 3$	1	-20.5 ± 1.4	$0.22, 0.22, 0.26, 0.28$	112
$M = 3$	0	-19.0 ± 2.3	$0.17, 0.19, 0.2, 0.26$	84
$N = 2$	1	-19.6 ± 1.9	$0.19, 0.22, 0.19, 0.2$	132
$\mathrm{DMP}_{N=4}$		-13.1 ± 0.8	$0.17, 0.17, 0.23, 0.28$	64
$\mathrm{DMP}_{N=8}$		-14.8 ± 1.4	$0.15, 0.2, 0.23, 0.29$	128
DMP inc.$_{N=4}$		-19.2 ± 0.6	$0.15, 0.20, 0.22, 0.3$	64
DMP inc.$_{N=8}$		-18.8 ± 0.5	$0.18, 0.19, 0.31, 0.26$	128

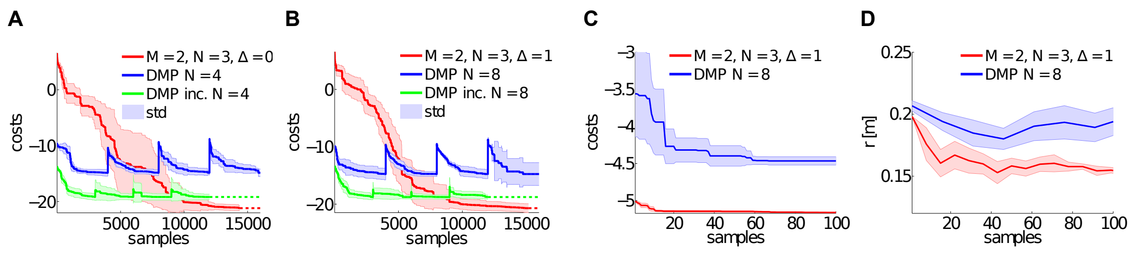


Figure 4.7: **Learning curves for the biped walker task.** This figure illustrates the learning performance over 10 runs of the proposed approach using $M = 2$ synergies with $N = 3$ Gaussian basis functions. In (A) the time-shift variables $\boldsymbol{\Delta}$**s** are not learned and set to zero. Whereas, in (B) also these $\boldsymbol{\Delta}$**s** variables are adapted during learning. We compare to the dynamic movement primitives (DMP) with $N = 4$ Gaussians in (A) and to DMPs with $N = 8$ Gaussians in (B). *DMP inc.* denotes an incremental learning setup, where DMPs were initialized with the best result from the previous task. Generalizing to a new step height ($r^* = 0.1$m) is shown in (C), where we applied the best learned policy for DMPSynergies from (B) and only optimized the weights $\beta_{1:2}$ for the two (fixed) synergies. The corresponding average step height over all steps is shown in (D). We compare to DMPs with $N = 8$ Gaussians.

that the objective function in Equation 4.11 is designed to prefer correct multi-step walking movements over exact matches of the step heights since learning to walk is already a complex learning problem (approximately 90 percent of the costs are determined by the travelled distance and only 5 percent are caused by the distance to the desired step

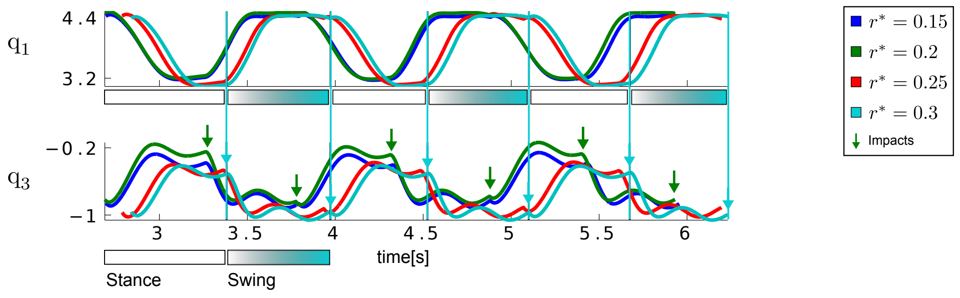


Figure 4.8: **Results for the biped walker task.** This figure illustrates the (initially) left hip angle denoted by q_1 and the left knee angle (q_3) for the multi-task learning scenario. Shown are the best learned trajectories using the proposed approach (with $M = 2$, $N = 3$ and $\Delta = 1$) for the desired step heights of $r^* \in \{0.15, 0.2, 0.25, 0.3\}$. The true step heights of the learned walking patterns are $0.22 \pm 0.07, 0.22 \pm 0.08, 0.26 \pm 0.08, 0.28 \pm 0.08$. The points in time of the ground contacts are denoted by large arrows for desired step heights of 0.25m and 0.3m. For the later additionally the duration of the stance and the swing phases are illustrated by large boxes.

heights). However, for the different desired step heights the shape of the trajectories as well as the moment of the impact vary. The moments of impact are denoted by arrows in Figure 4.8.

While generalizing to new motor skills was straightforward for the simple via-point task, for the walking tasks a linear interpolation turns out to be ineffective. We therefore demonstrate in Figure 4.7 (C-D) how a new walking pattern for a desired step height of $r^* = 0.1$m can be learned be reusing the previously learned prior knowledge (taking the best solution for $r^* = 0.25$m) for $M = 2$ synergies modeled by $N = 3$ Gaussians and $\Delta = 1$. Only the weights $\beta_{1:M}$ are optimized in this experiment, keeping the learned time-shifts fixed. The costs in Figure 4.7 (C) and the average step height r in Figure 4.7 (D) demonstrate the advantage of using a fixed prior, where we compare to DMPs with $N = 8$ Gaussians.

4.3.3 Multi-directional reaching task with a musculoskeletal model of the human arm

A simplified model of a human arm based on the model by Holzbaur et al. [2005] was used to learn six reaching tasks simultaneously. The shoulder and the elbow joint were modeled by hinge joints. Thus, only movements in the sagittal plane were possible. The initial arm configuration and the six target locations (with a distance of 15cm to a marker placed on the radial stylion) are shown in Figure 4.9 (A). A learned example movement

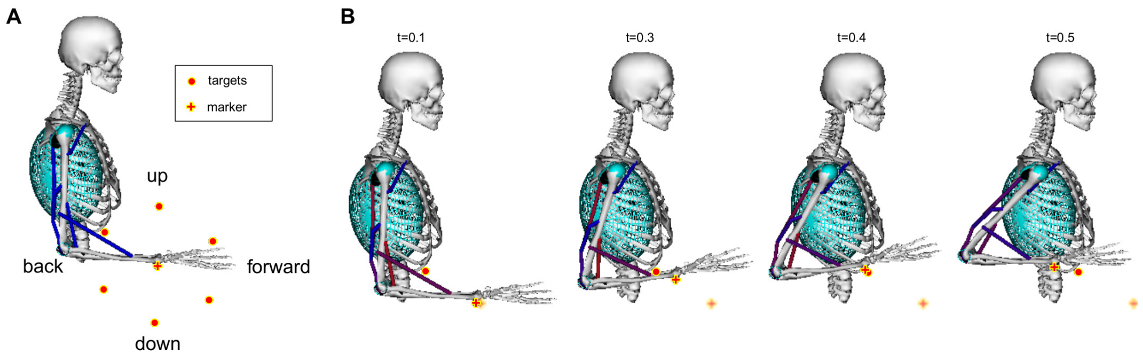


Figure 4.9: **Musculoskeletal model for learning reaching tasks.** A model of a human arm with eleven muscles shown in Table D.5 in the appendix was used to learn six reaching skills in the sagittal plane (A). As reward signal we encoded the distance to a marker placed on the radial stylion (denoted by the *plus* symbol) and punished large muscle excitation signals. Targets are denoted by large dots. We focused on fast reaching skills of 500ms duration, where an example movement is shown in (B). To simulate how muscles wrap over underlying bone and musculature wrapping surfaces are implemented as cylinders, spheres and ellipsoids [Holzbaur et al., 2005].

is illustrated in Figure 4.9 (B), where the cylinders, the spheres and the ellipsoids denote wrapping surfaces discussed in Section 4.2.3.

We focused on fast reaching movements of 500ms duration ($T = 500$ and $\Delta t = 1$ms) that can be implemented in an open-loop control scheme. Note that with our approach also closed-loop systems with feedback could be implemented, as discussed below. Thus, the learnable nonlinear function $\mathbf{f}(s,k)$ in Equation 4.10 is directly used as input to the system in form of muscle excitation patterns. The parameter settings for learning are shown in Table D.4 in the appendix.

For learning the reaching tasks we evaluated the Euclidean distance of a marker $\mathbf{v}_k(t)$ placed on the radial stylion to a given target $\mathbf{g}_k$, where $k = 1..6$ denotes the task index. Additionally, large muscle excitations signals are punished:

$$C(k) = \sum_{k=1}^{6} 3 \cdot \frac{1}{T} \sum_{t=1}^{T} \|\mathbf{g}_k - \mathbf{v}_k(t)\| + 10^{-3} \int_{s=0}^{1} \mathbf{f}(s,k)^T \mathbf{f}(s,k) \mathrm{d}s, \qquad (4.12)$$

where $\|.\|$ denotes the Euclidean distance between the marker $\mathbf{v}_k(t)$ and the target $\mathbf{g}_k$ at time t.

We evaluated five movement representations, defined in Equation 4.10, with an increasing number of shared synergies, i.e., $M = \{1, 2, 3, 4, 5\}$. Each synergy is represented by a single ($N = 1$) Gaussian. For each target and for each synergy the task-specific

Table 4.2: Details of the evaluated parametrizations and achieved costs for the reaching task. M denotes the number of implemented synergies, C_{mean} the final cost values, and the symbol $\pm$ the standard deviation. We use $\#_K$ to denote the number of task-specific parameters $(6 \cdot M)$ and $\#_M$ to denote the number of task-invariant or shared parameters $(3 \cdot 11 \cdot M)$.

No. of synergies	$\mathbf{C_{mean}}$	$\#_K$	$\#_M$	**Total**
$M = 1$	2.38 $\pm$0.05	12	33	45
$M = 2$	1.52 $\pm$0.12	24	66	90
$M = 3$	1.15 $\pm$0.12	36	99	135
$M = 4$	1.15 $\pm$0.05	48	132	180
$M = 5$	1.17 $\pm$0.05	60	165	225

parameters $\beta_{k,m}$ and $\Delta s_{k,m}$ are learned. The number of task-specific and the number of task-invariant or shared parameters is shown in Table 4.2.

We hypothesized that a muscle excitation signal can be generated by combining a small number of learned synergies. An example for the anterior deltoid muscle (DeltA) is shown in Figure 4.10 for two movement directions. Here, DMPSynergies with $M = 4$ synergies were used to generate the muscle excitation patterns. The muscle excitation patterns for all six movement directions and all eleven muscles are shown in Figure 4.11. Two observations can be made: first, as our objective function in Equation 4.12 punishes large muscle excitation signals a sparse representation of multiple motor skills is learned. Second, the learned muscle patterns partially show the typical triphasic behavior of human movement [Hallett et al., 1975, Angel, 1975, Berardelli et al., 1996, Chiovetto et al., 2010], where individual muscles (e.g., DeltA, PectClav and BRA in the first column in Figure 4.11) become activated at the onset of the movement, shortly before the velocity peak to decelerate, and finally, multiple muscles co-contract at the target location. These three events are denoted by the labels 1, 2, and 3 in the last row in Figure 4.11, where a threshold of 2 cm s^{-1} was used to determine the movement onset and the termination of the movement.

Comparing all five movement representations ($M = \{1, 2, 3, 4, 5\}$), we found that at least three synergies were necessary to accomplish all reaching tasks. This observation is best shown in Figure 4.12 (A), where with only one ($M = 1$) or two synergies ($M = 2$) not all targets can be reached. Shown are the marker trajectories of three independent learning sessions (out of ten runs). Note that similar findings were obtained in analyzing human arm reaching movements, where four to five synergies were observed [d'Avella

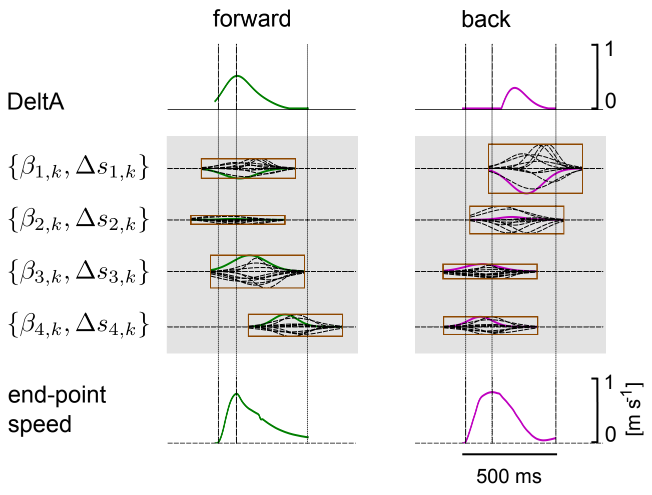


Figure 4.10: **Synergy combination mechanism.** We hypothesize that a muscle excitation signal can be generated by combining a small number of learned synergies. Here, we illustrate this combination process for the deltoid anterior (DeltA) with four synergies for two movement directions. For the two movement directions different combination coefficients $\beta_{m,k}$ and different time-shift parameters $\Delta s_{m,k}$ were learned. The synergies are represented by a single parametrized Gaussian, where the corresponding basis function for DeltA is denoted by a bold line in the enclosing rectangles.

et al., 2006]. The corresponding learning curves for all five movement representations are shown in Figure 4.12 (B), where the parametrizations with $M = 3..5$ synergies perform equal. This equal performance of the three parametrizations is also reflected in the final costs shown in Table 4.2 (rows 3..5). As an example the marker trajectories and the tangential velocity profiles for the representation using $M = 4$ synergies are illustrated in Figure 4.12 (C). As we evaluated an open-loop control scheme these marker trajectories did not exactly terminate at the target location (after the limited number of episodes for learning). However, by increasing the number of episodes or by adding feedback the terminal accuracy could be improved.

For testing the generalization ability of DMPSynergies we rotated all six targets by 30 degrees and only re-learned the task-specific coefficients, i.e., the mixing coefficients $\beta_{m,k}$ and the time-shift parameters $\Delta s_{m,k}$. Interim solutions with a movement representation implementing $M = 4$ synergies are shown in Figure 4.13 (A). Note that, as we evaluated an open-loop controller, the rotated targets were unknown to the controller. Solely the objective function in Equation 4.12 quantifies deviations from the targets. After 15 episodes a first trend towards the new targets was visible, however, most of the trajectories (three learned solutions are illustrated) ended at the original targets. The corresponding

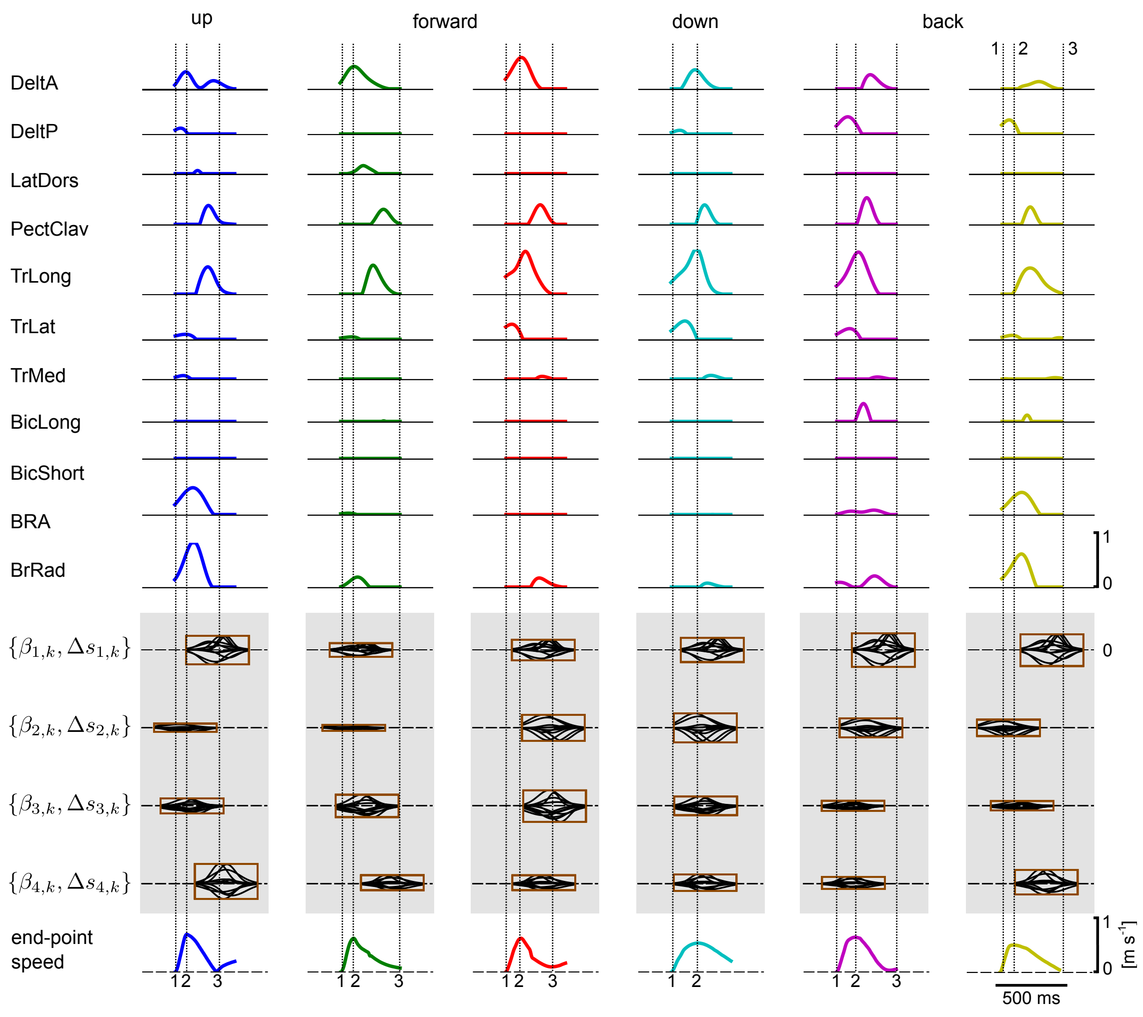


Figure 4.11: **Learned muscle excitation patterns.** Shown are the muscle excitation patterns for all six targets (from left to right) and for all muscles (rows one to eleven). In the last row the tangential velocity profiles of the marker placed on the radial stylion is illustrated (see text for details).

learning curves for DMPSynergies with three ($M = 3$) and four ($M = 4$) synergies are shown in Figure 4.13 (B). The learning curve for the unperturbed scenario from the previous experiment is denoted by the dashed line ($M = 4$ orig.). Note that in both - the unperturbed and the perturbed experiments $K = 6$ reaching movements were learned, which demonstrates the benefit of the shared learned knowledge when generalizing new skills. For a comparison the blue line denoted by DMP $N = 4$ illustrates the convergence rate of single task learning with DMPs, where DMPSynergies ($M = 4$ orig.) can compete in terms of learning speed.

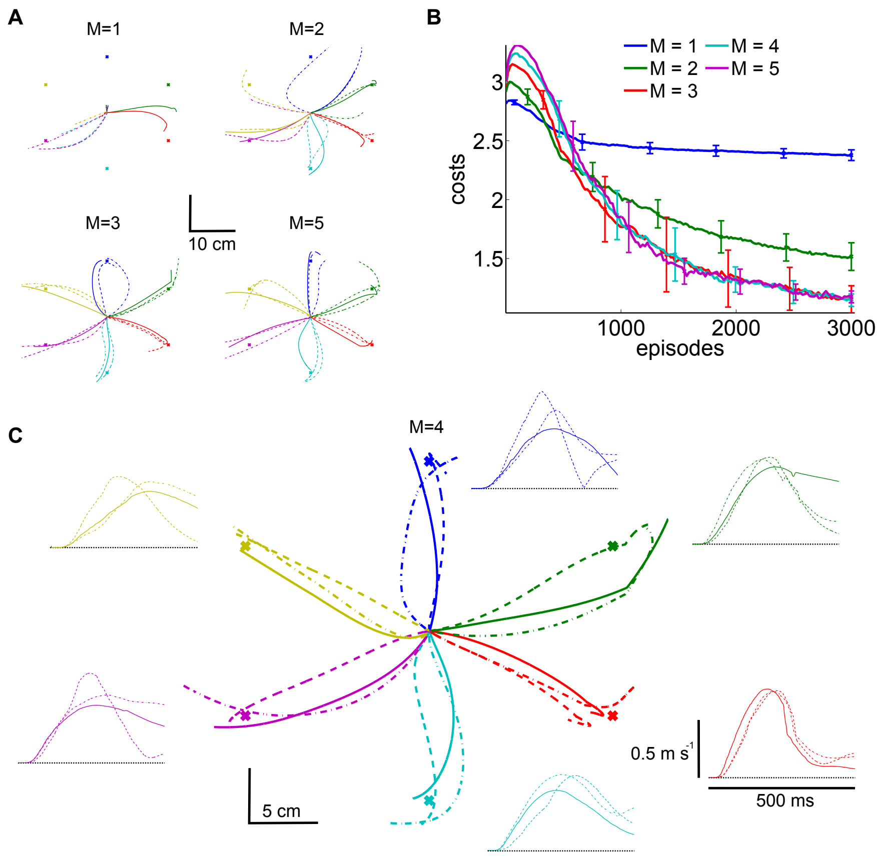


Figure 4.12: **Learning multi-directional reaching movements.** We evaluated five movement representations with an increasing number of shared synergies, i.e., $M = \{1, 2, 3, 4, 5\}$. The resulting trajectories of the marker placed on the radial stylion are shown in (A) and (C), where with less than three synergies not all targets can be reached. Illustrated are three independent learning results. In (B) we illustrate the average learning curves over 10 runs for these movement representations. For the representation using $M = 4$ synergies shown in (C) additionally the tangential velocity profiles are illustrated.

4.4 Discussion

We proposed a movement representation based on learned parametrized synergies (DMP-Synergies) that can be linearly combined and shifted in time. These learned synergies are shared among multiple task instances significantly facilitating learning of motor control policies. This was demonstrated on simulated robotic and on musculoskeletal systems. Below we discuss the significance and the implication of our findings with respect to robotics and biological motor control.

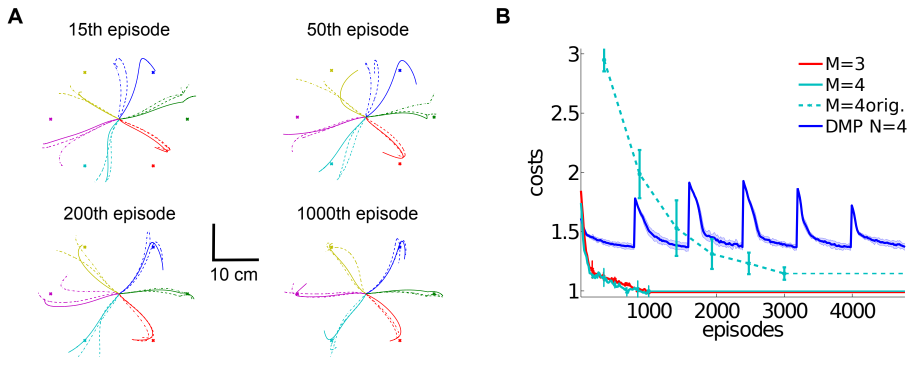


Figure 4.13: **Generalization to new reaching directions.** For testing the generalization ability of the proposed DMPSynergies we fix the learned shared synergies and only adapt the task-specific parameters, i.e., the mixing coefficients $\beta_{m,k}$ and the time-shift parameters $\Delta s_{m,k}$. The $K = 6$ targets were rotated by 30 degrees, where in (A) the marker trajectories after $15, 50, 200$, and 1000 episodes for a movement representation with $M = 4$ synergies are shown. In (B) we show the averaged learning curves for DMPSynergies with three and four synergies over 10 runs ($M = 3$ and $M = 4$). The learning curve for the unperturbed scenario from the previous experiment is denoted by the dashed line ($M = 4$ orig.). For a comparison the blue line denoted by DMP $N = 4$ illustrates the convergence rate of single task learning.

4.4.1 Exploiting shared synergies for motor skill learning in robotics

For motor skill learning in robotics a common strategy is to use parametrized elementary movements or movement primitives [Kober and Peters, 2011]. In this paper we proposed a generalization of the most widely used movement primitive representation in robotics, dynamic movement primitives (DMPs) [Schaal et al., 2003, Ijspeert et al., 2013]. DMPs evaluate parametrized dynamical systems to generate trajectories. The dynamical system is constructed such that the system is stable. This movement representation has many advantages. It is a model-free approach, partially explaining its popularity in robotics as model learning in high-dimensional stochastic robotics systems is challenging. Further, its stable attractor system facilitates learning and DMPs can represent both rhythmic and discrete movements. Meta parameters can be used for adapting the movement speed or the goal state. Finally, the movement representation depends linearly on the policy parametrization, i.e., the learnable function f depends linearly on the parameters $\boldsymbol{\theta}$ of the movement primitive, $f(s) = \boldsymbol{\Phi}(s)^T\boldsymbol{\theta}$, where s is the time or phase variable. As a result, imitation learning for DMPs is straightforward, as this can simply be done by performing linear regression [Schaal et al., 2003]. However, for each task k an individual set of parameters $\boldsymbol{\theta}_k$ has to be learned, which unnecessarily complicates learning of a large number of related motor skills. In contrast we proposed a generalization that allows for reusing shared knowledge among multiple related motor skills, i.e., the parameter vector $\boldsymbol{\theta}$ is task-invariant.

In particular, we replaced the nonlinear modulation function $f(.)$ in DMPs by a hierarchical function approximator. On the lower level task related parameters (amplitude scaling weights and time-shift parameters) are used to modulate a linear superposition of basis functions. These basis functions encode shared higher level knowledge and are modeled by a mixture of Gaussians. With the proposed DMPSynergies representation discrete and rhythmic movements can be generated. By using Gaussians at the higher level DMPs can be implemented as special case. However, the DMPSynergies can compete with DMPs in terms of learning efficiency while allowing for learning multiple motor skills simultaneously.

This was demonstrated in two robotic multi-task learning scenarios, where we showed that, with the DMPSynergies, good policies could be found more reliably, i.e., local minima with high cost values were more often avoided, more efficiently (fewer samples were needed), and new skills could be generalized by exploiting the previously learned shared knowledge. A simple via-point task was used to demonstrate the characteristics of the approach, where the proposed movement representation could be used to generalize new movement trajectories by applying a linear interpolation on the synergy's weights β. In a second robotic task, a biped walker task, the hierarchical representation was used to learn walking patterns with multiple step heights. In this complex reinforcement learning task, it was shown that better solutions were found more reliably by exploiting the learned shared knowledge, which is a strong feature of a movement representation. While also with the classical DMP approach high quality movement gaits were learned, on average the achieved costs were higher compared to the proposed hierarchical synergies representation, i.e., -19.2 ± 0.6 for DMPs with 4 Gaussians (and with incremental learning) compared to -21.4 ± 0.4 when using $M = 2$ synergies with $N = 3$ Gaussians (where the time-shift parameters were fixed and set to zero, $\Delta = 0$). In this experiment 10000 samples were needed to learn 4 walking gaits simultaneously, where the DMPSynergies approach can compete with DMPs (15000 samples). Additionally, we demonstrated in a generalization experiment that walking patterns for an unknown step height ($r^* = 0.1$m) could be learned with 100 samples by exploiting the previously learned prior knowledge.

While DMPs [Schaal et al., 2003, Ijspeert et al., 2013] are most closely related to our shared synergies approach, there exist a few other approaches [Chhabra and Jacobs, 2006, Alessandro et al., 2012] also implement shared knowledge. In Chhabra and Jacobs [2006] a variant of non-negative matrix factorization [d'Avella et al., 2003] was used to compute the synergies given a set of trajectories created by applying stochastic optimal control methods [Li and Todorov, 2004]. In Alessandro et al. [2012] an exploration phase was introduced to compute the dynamic responses of a robot system with random initialization. After a reduction phase, where a small number of proto-tasks were executed, a reduced set of dynamic responses was used to compute the synergies matrix by solving a linear system of equations. We proposed an alternative for learning the synergies and their combination parameters, where all unknowns are learned in a reinforcement learning setting from a single sparse reward signal. Moreover, for robotic tasks we embed the synergies approach in stable dynamical systems like in DMPs. This combines the benefits of DMPs and muscle synergies, namely the efficient learning ability of DMPs in high-dimensional systems and the hierarchical representation of movements that can be used for multi-task learning.

As with DMPs the complexity of the DMPSynergies representation can be scaled by the number of combined synergies or the number of implemented Gaussians modeling these synergies. However, as the trajectories generated with our representation depend nonlinearly on the policy parameters (in contrast to DMPs) more sophisticated decomposition strategies like for example d'Avella and Tresch [2001], Chiovetto et al. [2013] are needed for imitation learning. With such approaches the extracted synergies could be implemented as initial solutions in our learning framework.

4.4.2 Learned shared synergies for biological movement generation

The idea of reusing shared knowledge for movement generation is a well known concept in biological motor control. Muscle activation patterns recorded during multiple task instances of natural motor behavior, i.e., fast reaching movements of humans [d'Avella et al., 2006], primate grasping movements [Overduin et al., 2008], or walking patterns [Dominici et al., 2011], could be efficiently modeled by combining only few muscle activation patterns. In particular, time-varying muscle synergies [d'Avella et al., 2003, Bizzi et al., 2008] were proposed to be a compact representation of muscle activation patterns. The key idea of this approach is that muscle activation patterns are linear sums of simpler, elemental functions or synergies. Each muscle synergy can be shifted in time and scaled with a linear factor to construct a large variety of activation patterns. In this manuscript we proposed a generative model to represent and learn time-varying synergies [d'Avella et al., 2006].

The proposed framework allows for studying the concept of muscle synergies from a generative perspective in contrast to the analytical approach, where muscle synergies are identified from observed data. Applying such a generative approach to a musculoskeletal model, we could provide a proof-of-concept of the feasibility of a low-dimensional controller based on shared synergies and a demonstration of its learning efficiency. Moreover, we could ask different question, i.e., how does performance scale with the complexity of the movement representation, how sparse is the encoding of the muscle patterns to solve particular tasks, and how well does the learned representation generalize to new movements? We addressed these questions in a multi-directional reaching task, where we investigated a musculoskeletal model of the upper limb with 11 muscles. Motor skills for 6 reaching directions were learned within 3000 episodes and by exploiting the learned shared synergies movements for rotated target directions can be generalized 3 times faster (Figure 4.13). We found that a minimum of three synergies were necessary to solve the task (Figure 4.12 B). In our objective function large muscle excitation signals were punished, which resulted in a sparse representation of muscle excitation patterns. This sparse representation illustrated in Figure 4.11 shows similarities to observed electromyographic activity recorded in related human reaching tasks [d'Avella et al., 2006], i.e., triphasic muscle patterns, where some of the muscles contributed at the movement onset, some at point of the maximum tangential velocity, and some at the end of the movement to co-contract. However, sensor feedback might be an important modulation signal to make this effect more pronounced.

The model was designed to capture salient features of human musculoskeletal system, such as muscle activation dynamics, Hill-type musculotendinous units, realistic geometry. However, to reduce the computational effort needed to simulate a movement we made a few simplifying assumptions. First, a limited number of muscles (11) were implemented, where simplified wrapping objects and muscle paths were modeled. Further, we implemented the shoulder and the elbow joint as hinge joints. Thus, only reaching movements in the sagittal plane could be performed. Finally, we focused on fast reaching movements in an open-loop control scheme. This was a valid assumption for comparing to human data for fast reaching movements [d'Avella et al., 2006]. However, our proposed learning and control framework also allows for implementing closed-loop controllers, i.e., when introducing an inverse kinematics model $\mathbf{\Psi} \in \mathbb{R}^{Dx3}$ in Equation 4.1, i.e., $\tau\dot{\mathbf{z}} = \mathbf{\Psi}(\alpha_z(\beta_z(\mathbf{g}-\mathbf{y}^*)-\mathbf{z}))+\mathbf{f}$, where D denotes the number of actuators and we assumed that the goal state $\mathbf{g}$ lives in a three-dimensional Cartesian space. The inverse kinematics model $\mathbf{\Psi}$ maps the feedback error signal into the muscle pattern space and modulates the learned muscle excitation basis $\mathbf{f} \in \mathbb{R}^D$. With such closed-loop systems we might better understand the contribution of feedback to muscle control in biological movement generation [Lockhart and Ting, 2007].

Musculoskeletal models have been used before to investigate movement generation with muscle synergies [Berniker et al., 2009, Neptune et al., 2009, McKay and Ting, 2012]. Berniker and colleagues used model-order reduction techniques to identify synergies as a low-dimensional representation of a non-linear system's input/output dynamics and optimal control to find the activations of these synergies necessary to produce a range of movements. They found that such a set of synergies was capable of producing effective control of reaching movements with a musculoskeletal model of a frog limb and that it was possible to build a relatively simple controller whose overall performance was close to that of the system's full-dimensional nonlinear controller. Neptune and colleagues generated muscle-actuated forward dynamics simulations of normal walking using muscle synergies identified from human experimental data using non-negative matrix factorization as the muscle control inputs. The simulation indicated that a simple neural control strategy involving five muscle synergies was sufficient to perform the basic sub-tasks of walking. McKay and Ting, studying an unrestrained balance task in cats, used a static quadrupedal musculoskeletal model of standing balance to identify patterns of muscle activity that produced forces and moments at the center of mass (CoM) necessary to maintain balance in response to postural perturbations. CoM control could be accomplished with a small number of muscle synergies identified from experimental data, suggesting that muscle synergies can achieve similar kinetics to the optimal solution, but with increased control effort compared to individual muscle control. In line with these simulation studies, we also found that a small number of muscle synergies was sufficient to perform multiple reaching tasks in a forward dynamic simulation of a musculoskeletal model. However, we did not use experimental data or model-order reduction techniques to identify muscle synergies. In our framework, both synergy structural parameters and synergy combination parameters were found with reinforcement learning, supporting the generality of the solutions identified. Moreover, we were able to test the generalization ability of the synergies in the same framework by optimizing only the task-specific synergy combination parameters.

The proposed reinforcement learning framework with movement primitives relates to optimal control approaches in the biological motor control literature [Erdemir et al., 2007, Delp et al., 2007]. In these simulation studies muscle patterns are parametrized by e.g., bang-bang (on-off) controls, constant control values, or control vectors approximated with polynomials (see Table 2 in Erdemir et al. [2007] for an overview of different control strategies). However, to the best of or knowledge non of these approaches implemented shared synergies as control signal representation for learning multiple task instances simultaneously. Even with complex representations, e.g. with $M = 5$ synergies learning 225 parameters converged within 3000 episodes, which is a promising feature of the proposed approach for studies on more complex musculoskeletal models.

In this manuscript we demonstrated how time-varying synergies [d'Avella et al., 2006] can be implemented and learned from scratch. Interestingly, by adding an additional constraint on the movement representation, i.e., by using a single policy vector for all actuators anechoic mixing coefficients [Giese et al., 2009] can be implemented. However, in general any synergy representation such as synchronous synergies [Ivanenko et al., 2004, Dominici et al., 2011] used for locomotion can be learned. Thus, we do not argue for a particular synergy representation. Our representation was motivated to extend the widely used DMPs [Schaal et al., 2003] for exploiting shared task-invariant knowledge for motor skill learning in robotics.

4.4.3 Conclusion

We proposed a movement primitive representation implementing shared knowledge in form of learned synergies. The representation competes with the state-of-the-art, it can implement DMPs [Schaal et al., 2003] as a special case, and it allows for an efficient generalization to new skills. Importantly, shared knowledge simplifies policy search in high-dimensional spaces, which was demonstrated in a dynamic biped walking task. Further, the proposed learned synergies are a compact representation of high-dimensional muscle excitation patterns, which allows us to implement reinforcement learning in musculoskeletal systems. In such frameworks muscle patterns are learned from scratch using a sparse reward signal, where we could investigate how muscles and muscle synergies contribute to a specific task, how complex a task-invariant representation must be, and how well the learned synergies generalize to changes in the environment. In a multi-directional arm reaching experiment we provided first insights to these questions. In future research the proposed movement generation and learning framework will be used to study feedback signals and feedback delays, imitation learning from biological data and the effect of simulated muscle surgeries.

4.5 Acknowledgments

This chapter is based on the paper Rückert and d'Avella [2013] written by Elmar Rückert (ER), Andrea d'Avella (AD). ER provided the basic ideas, the algorithmic implementation and conducted the experiments. AD provided guidance and important comments on the experimental design. Paper writting was mostly done by ER with highly appreciated assistance from AD. Gerhard Neumann was involved in early discussions and provided important ideas and comments to the model.

Appendix A

List of publications

[1] E. Rückert, G. Neumann. (2011). A study of Morphological Computation by using Probabilistic Inference for Motor Planning. In *Proceedings of the 2nd International Conference on Morphological Computation (ICMC 2011)*, pp. 51-53, Venice, Italy, 2011.

[2] E. Rückert, G. Neumann. (2012). Stochastic Optimal Control Methods for Investigating the Power of Morphological Computation. In *Artificial Life*, vol. 19, no. 1, pp. 1-17, 2012. DOI=10.1162/ARTL_a_00085.

[3] E. Rückert, G. Neumann, M. Toussaint, W. Maass. (2013). Learned graphical models for probabilistic planning provide a new class of movement primitives. In *Frontiers in Computational Neuroscience*, vol. 6, no. 97, 2013. DOI=10.3389/fncom.2012.00097.

[4] E. Rückert, A. d'Avella. (2013). Learned Muscle Synergies as Prior in Dynamical Systems for Controlling Bio-mechanical and Robotic Systems. In *Abstracts of Neural Control of Movement Conference (NCM 2013)*, Conference Talk, pp. 27-28, Puerto Rico, USA, 2013.

[5] E. Rückert, A. d'Avella. (2013). Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. In *Frontiers in Computational Neuroscience*, vol. 6, no. 97, 2013. DOI=10.3389/fncom.2013.00138.

A.1 Comments and Contributions to Publications

The paper *A study of Morphological Computation by using Probabilistic Inference for Motor Planning* was written by myself (ER) and Gerhard Neumann (GN). Together, the two authors developed the basic ideas, the experimental design, and did the writting. The algorithms and experiments were implemented by ER. The paper was presented at the *2nd International Conference on Morphological Computation* (ICMC) 2011. It was selected by the conference committee members to be submitted to the Special Issue on Morphological Computation in the journal *Artificial Life*. This approach was extended in the journal paper *Stochastic Optimal Control Methods for Investigating the Power of Morphological Computation*, which was written by the same authors, published in *Artificial Life* 2012. This publication includes a novel approximate inference algorithm for planning problems with control constraints as well as a number of additional simulation experiments. It provides the basis for Chapter 2 of this thesis.

The paper *Learned graphical models for probabilistic planning provide a new class of movement primitives* was written by myself (ER), Gerhard Neumann (GN), Marc Toussaint (MT), and Wolfgang Maass (WM). Together, GN and ER developed the basic ideas and the experimental design. The algorithms were implemented by ER, who also performed the experiments. Most of the manuscript was written by ER and GN with highly appreciated assistance from MT. WM greatly improved the paper by contributing important thoughts on how to relate this work to salient features of biological motor control. This paper was published in *Frontiers in Computational Neuroscience* and provides the basis for Chapter 3 of this thesis.

The paper *Learned Muscle Synergies as Prior in Dynamical Systems for Controlling Bio-mechanical and Robotic Systems* was written by myself (ER) and Andrea d'Avella (AD). ER provided the basic ideas, the algorithmic implementation and conducted the experiments. AD provided guidance and important comments on the experimental design. Paper writting was mostly done by ER with highly appreciated assistance from AD. This paper was selected in a highly competitive review process for a conference talk at the *Neural Control of Movement Conference* (NCM) 2013. This approach was extended in the journal paper *Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems* by the same authors, published in *Frontiers in Computational Neuroscience* 2013. This publication includes a detailed model description, links the approach to biological movement control strategies, and includes simulations of musculoskeletal systems. It is the basis for Chapter 4 of this thesis.

Appendix B

Appendix to Chapter 2: A study of morphological computation using probabilistic inference

B.1 Extension of the approximate inference control method

The original formulation of the Approximate Inference Control (AICO) method [Toussaint, 2009] does not consider a linear term for the control costs. However, this linear term is needed to encode torque limits, which are important for our dynamic balancing experiments, and hence, we needed to extend AICO algorithm.

The introduction of a linear term for the control costs yields not only in a modified cost function, but also results in different update equations for the messages and finally in different equations of the optimal feedback controller. For completeness we will first recap the main steps to derive the AICO method and will then discuss the modifications to implement control constraints.

B.1.1 Approximate inference control without torque limits

For motor planning we consider the stochastic process:

$$P(\mathbf{x}_{1:T}, \mathbf{u}_{1:T-1}, \mathbf{z}_{1:T}) = P(\mathbf{x}_1) \prod_{t=1}^{T-1} P(\mathbf{u}_t|\mathbf{x}_t) \prod_{t=1}^{T} P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \prod_{t=1}^{T-1} P(z_t|\mathbf{x}_t, \mathbf{u}_t),$$

where $P(\mathbf{u}_t|\mathbf{x}_t)$ denotes the state dependent prior for the controls, the distribution $P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ the state transition model and $P(\mathbf{x}_1)$ the initial state distribution. Here, we assume that the prior of the controls is independent of the states and thus we will simply use $P(\mathbf{u}_t|\mathbf{x}_t) = P(\mathbf{u}_t)$ for the rest of the appendix. The time horizon is fixed to T time-steps. The binary task variable z_t denotes a reward event, its probability is defined by $P(z_t = 1|\mathbf{x}_t, \mathbf{u}_t) \propto \exp(-c_t(\mathbf{x}_t, \mathbf{u}_t))$, where $c_t(\mathbf{x}_t, \mathbf{u}_t)$ is the immediate cost function[*] for time-step t. It expresses a performance criteria (like avoiding a collision, or reaching a goal).

We want to compute the posterior $P(\mathbf{x}_{1:T}, \mathbf{u}_{1:T}|z_{1:T} = 1)$ over trajectories, conditioned on observing a reward ($z_t = 1$) at each time-step t. This posterior can be computed by using message passing in the given graphical model of Figure 2.1. To simplify the computations we integrate out the controls:

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t) = \int_{\mathbf{u}_t} P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)P(\mathbf{u}_t|\mathbf{x}_t)d\mathbf{u}_t. \tag{B.1}$$

The marginal belief $b_t(\mathbf{x}_t)$ of a state at time t is given by:

$$b_t(\mathbf{x}_t) = \alpha_t(\mathbf{x}_t)\beta_t(\mathbf{x}_t)\phi_t(\mathbf{x}_t), \tag{B.2}$$

where $\alpha_t(\mathbf{x}_t)$ is the forward message, $\beta_t(\mathbf{x}_t)$ is the backward message $\phi_t(\mathbf{x}_t)$ is the current task message. The messages are given by:

$$\begin{aligned}
\alpha_t(\mathbf{x}_t) &= \int_{\mathbf{x}_{t-1}} P(\mathbf{x}_t|\mathbf{x}_{t-1})\alpha_{t-1}(\mathbf{x}_{t-1})\phi_{t-1}(\mathbf{x}_{t-1})d\mathbf{x}_{t-1}, && \text{(B.3)}\\
\beta_t(\mathbf{x}_t) &= \int_{\mathbf{x}_{t+1}} P(\mathbf{x}_{t+1}|\mathbf{x}_t)\beta_{t+1}(\mathbf{x}_{t+1})\phi_{t+1}(\mathbf{x}_{t+1})d\mathbf{x}_{t+1}, && \text{(B.4)}\\
\phi_t(\mathbf{x}_t) &= P(z_t|\mathbf{x}_t). && \text{(B.5)}
\end{aligned}$$

We consider discrete-time, nonlinear stochastic systems with zero mean Gaussian noise

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{x}_{t+1}|f_{\mathrm{Dyn}}(\mathbf{x}_t, \mathbf{u}_t), \mathbf{Q}_t).$$

The nonlinear stochastic system f_{Dyn} is approximated by a Linear dynamics, Quadratic costs and Gaussian noise system (LQG) by Taylor expansion [Toussaint, 2009, Todorov

[*]In this paper the immediate cost function is composed of the intrinsic costs and the constraint costs, i.e., $c_t(\mathbf{x}_t, \mathbf{u}_t) + c_p(\mathbf{x}_t, \mathbf{u}_t)$

and Li, 2005]:

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t). \tag{B.6}$$

Thus, the system is linearized along a given trajectory $\langle\hat{\mathbf{x}}_{1:T}, \hat{\mathbf{u}}_{1:T-1}\rangle$ at every point in time. We will use f_t as shorthand for $f_{\mathrm{Dyn}}(\mathbf{x}_t, \mathbf{u}_t)$. Then, the state transition matrix $\mathbf{A}_t$ is given by $\mathbf{A}_t = (\mathbf{I} + \frac{\delta f_t}{\delta \mathbf{x}_t}\Delta t)$, the control matrix $\mathbf{B}_t$ is $\mathbf{B}_t = \frac{\delta f_t}{\delta \mathbf{u}_t}\Delta t$ and the linear term reads $\mathbf{a}_t = (f_t - \frac{\delta f_t}{\delta \mathbf{x}_t}\mathbf{x}_t - \frac{\delta f_t}{\delta \mathbf{u}_t}\mathbf{u}_t)\Delta t$.

In the original formulation of the AICO algorithm the cost function c_t is approximated as:

$$c_t(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T\mathbf{R}_t\mathbf{x}_t - 2\mathbf{r}_t^T\mathbf{x}_t + \mathbf{u}_t^T\mathbf{H}_t\mathbf{u}_t.$$

Note that there is no linear term for the control costs as we only punish quadratic controls. We can now write $P(z_t = 1|\mathbf{x}_t, \mathbf{u}_t) = P(z_t = 1|\mathbf{x}_t)P(\mathbf{u}_t)$ as

$$P(z_t = 1|\mathbf{x}_t) \propto \mathcal{N}[\mathbf{x}_t|\mathbf{r}_t, \mathbf{R}_t], \tag{B.7}$$

$$P(\mathbf{u}_t) = \mathcal{N}[\mathbf{u}_t|\mathbf{0}, \mathbf{H}_t], \tag{B.8}$$

where the distributions in Equation B.7 and B.8 are given in canonical form. The canonical form of a Gaussian is used because numerical operations such as products or integrals are easier to calculate in this notation. The canonical form is indicated by the *square* bracket notation and given by

$$\mathcal{N}[\mathbf{x}|\mathbf{a}, \mathbf{A}] = \frac{\exp(-1/2\mathbf{a}^T\mathbf{A}^{-1}\mathbf{a})}{|2\pi\mathbf{A}^{-1}|^{1/2}}\exp(-1/2\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{a}).$$

A Gaussian in normal form can always be transformed into the canonical form by $\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A}) = \mathcal{N}[\mathbf{x}|\mathbf{A}^{-1}\mathbf{a}, \mathbf{A}^{-1}]$. For more details we refer to the Gaussian Identities in Toussaint [2011].

We can see in Equation B.8 that our prior for applying the control $\mathbf{u}_t$ is given by the control costs, i.e., $\mathcal{N}[\mathbf{u}_t|\mathbf{0}, \mathbf{H}_t]$. By integrating out the controls from our system dynamics we get the following state transition probabilities

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t) = \int_{\mathbf{u}_t} \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t)\mathcal{N}[\mathbf{u}_t|\mathbf{0}, \mathbf{H}_t]d\mathbf{u}_t, \tag{B.9}$$

$$= \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t, \mathbf{Q}_t + \mathbf{B}_t\mathbf{H}_t^{-1}\mathbf{B}_t^T), \tag{B.10}$$

where the integral was solved using a reformulation of the *propagation* rule in Toussaint

[2011].

As we can see, all distributions in the approximated LQG system in Equation B.10 are Gaussian, and thus, also all messages are Gaussians and can be calculated analytically. The resulting messages are given in Toussaint [2009].

B.1.2 Approximate inference control with torque limits

In order to implement torque and joint limits we introduce an additional cost function c_p which punishes the violation of the given constraints. The function c_p is just added to the current immediate costs. We use separate cost terms for control constraints c_t^u and joint constraints c_t^x, i.e., $c_p(\mathbf{x}_t, \mathbf{u}_t) = c_t^x(\mathbf{x}_t) + c_t^u(\mathbf{u}_t)$. Here, we will only discuss how to implement the function $c_t^u(\mathbf{u}_t)$ for the torque constraints, joint constraints are implemented similarly.

The cost function c_t^u is quadratic in $\mathbf{u}$ and punishes leaving the valid control limits of $\mathbf{u}$. In order to implement the upper bound $\mathbf{u}_{\max}$ for the torques, we use the following cost function

$$\begin{aligned} c_t^u(\mathbf{u}_t) &= \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t + (\mathbf{u}_t - \mathbf{u}_{\max})^T \mathbf{H}_t^U (\mathbf{u}_t - \mathbf{u}_{\max}), \\ &= \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t + \mathbf{u}_t^T \mathbf{H}_t^U \mathbf{u}_t - 2\mathbf{u}_{\max}^T \mathbf{H}_t^U \mathbf{u}_t + \mathbf{u}_{\max}^T \mathbf{H}_t^U \mathbf{u}_{\max}, \\ &= \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t + \mathbf{u}_t^T \mathbf{H}_t^U \mathbf{u}_t - 2\mathbf{u}_{\max}^T \mathbf{H}_t^U \mathbf{u}_t + \text{const}, \end{aligned}$$

where matrix $\mathbf{H}_t$ denotes the quadratic control costs. The constrained costs are only imposed for the control variable u_i if the torque value exceeds the upper bound $u_{\max,i}$. In order to do so $\mathbf{H}_t^U$ is a diagonal matrix where the ith diagonal entry is zero if $u_i \leq u_{\max,i}$ and non-zero otherwise. The lower bound $\mathbf{u}_{\min}$ is implemented likewise using an individual diagonal matrix $\mathbf{H}_t^L$.

We can again implement $c_t^u(\mathbf{u}_t)$ as prior distribution $P(\mathbf{u}_t)$ for the controls.

$$P(\mathbf{u}_t) \;\propto\; \mathcal{N}[\mathbf{u}_t | \mathbf{h}_t, \mathbf{H}_t], \tag{B.11}$$

where $\mathbf{h}_t = \mathbf{u}_{\max}^T \mathbf{H}_t^U + \mathbf{u}_{\min}^T \mathbf{H}_t^L$ and the precision $\hat{\mathbf{H}}_t = \mathbf{H}_t + \mathbf{H}_t^U + \mathbf{H}_t^L$. As we can see, the linear term $\mathbf{h}_t$ of the prior distribution $P(\mathbf{u}_t)$ is now *non-zero*. This yields different message equations.

Joint-limits can be imposed similarly by using additional terms costs for $c_t^x(\mathbf{x}_t)$. However, for joint limits the update equations stay the same because $P(z_t = 1|\mathbf{x}_t)$ has already a non-zero mean denoted by $\mathbf{r}_t$ in Equation B.7.

To derive the messages we will first integrate out the controls to get the state transition probabilities:

$$\begin{aligned} P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= \int_{\mathbf{u}_t} \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t)\mathcal{N}[\mathbf{u}_t|\mathbf{h}_t, \hat{\mathbf{H}}_t]d\mathbf{u}_t, \\ &= \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{B}_t\hat{\mathbf{H}}_t^{-1}\mathbf{h}_t, \mathbf{Q}_t + \mathbf{B}_t\hat{\mathbf{H}}_t^{-1}\mathbf{B}_t^T). \qquad \text{(B.12)} \end{aligned}$$

Note that, since the cost function $c_t^u(\mathbf{u}_t)$ contains a non-zero linear term $\mathbf{h}_t$, we get a new linear term $\hat{\mathbf{a}}_t = \mathbf{a}_t + \mathbf{B}_t\mathbf{H}_t^{-1}\mathbf{h}_t$ in the transition dynamics. The forward and the backward messages are the same like in Toussaint [2009] except that $\mathbf{a}_t$ is replaced by $\hat{\mathbf{a}}_t$ and $\mathbf{H}_t$ by $\hat{\mathbf{H}}_t$.

Like in Toussaint [2009] we use the canonical representations for the forward and the backward message:

$$\begin{aligned} \alpha_t(\mathbf{x}_t) &= \mathcal{N}[\mathbf{x}_t|\mathbf{s}_t, \mathbf{S}_t], \\ \beta_t(\mathbf{x}_t) &= \mathcal{N}[\mathbf{x}_t|\mathbf{v}_t, \mathbf{V}_t], \\ \phi_t(\mathbf{x}_t) &= P(z_t|\mathbf{x}_t) = \mathcal{N}[\mathbf{x}_t|\mathbf{r}_t, \mathbf{R}_t]. \end{aligned}$$

The messages are represented by Gaussians in canonical form, for which mathematical operations like products are simply performed by adding the linear terms and the precisions. The mean of the belief is given by $b_t(\mathbf{x}_t) = (\mathbf{S}_t + \mathbf{V}_t + \mathbf{R}_t)^{-1}(\mathbf{s}_t + \mathbf{v}_t + \mathbf{r}_t)$ (multiplying three canonical messages and a subsequent transformation to normal form). Furthermore we use the shorthand $\bar{\mathbf{Q}}_t = \mathbf{Q}_t + \mathbf{B}_t\hat{\mathbf{H}}_t^{-1}\mathbf{B}_t^T$ for the covariance in Equation B.12.

The messages are computed by inserting the state transition probabilities given in Equation B.12 in the message passing Equations B.3 and B.4. Subsequently the integrals are solved using the *propagation* rule in Toussaint [2011]. The final equations in canonical form are:

$$\begin{aligned} \mathbf{S}_t &= (\mathbf{A}_{t-1}^{-T} - \mathbf{K}_s)\mathbf{S}_{t-1}\mathbf{A}_{t-1}^{-1}, && \text{(B.13)} \\ \mathbf{s}_t &= (\mathbf{A}_{t-1}^{-T} - \mathbf{K}_s)(\bar{\mathbf{s}}_{t-1} + \mathbf{S}_{t-1}\mathbf{A}_{t-1}^{-1}(\hat{\mathbf{a}}_{t-1} + \mathbf{B}_{t-1}\hat{\mathbf{H}}_{t-1}^{-1}\mathbf{h}_{t-1})), && \text{(B.14)} \\ \mathbf{K}_s &= \mathbf{A}_{t-1}^{-T}\mathbf{S}_{t-1}(\mathbf{S}_{t-1} + \mathbf{A}_{t-1}^{-T}\bar{\mathbf{Q}}_{t-1}^{-1}\mathbf{A}_{t-1}^{-1})^{-1}. && \text{(B.15)} \end{aligned}$$

And for the backward messages:

$$\mathbf{V}_t = (A_t^T - \mathbf{K}_v)\bar{\mathbf{V}}_{t+1}\mathbf{A}_t, \tag{B.16}$$

$$\mathbf{v}_t = (\mathbf{A}_t^T - \mathbf{K}_v)(\bar{\mathbf{v}}_{t+1} - \bar{\mathbf{V}}_{t+1}(\hat{\mathbf{a}}_t + \mathbf{B}_t\hat{\mathbf{H}}_t^{-1}\mathbf{h}_t)), \tag{B.17}$$

$$\mathbf{K}_v = \mathbf{A}_t^T\bar{\mathbf{V}}_{t+1}(\bar{\mathbf{V}}_{t+1} + \bar{\mathbf{Q}}_t^{-1})^{-1}. \tag{B.18}$$

To obtain the same mathematical form as in Toussaint [2009] one needs to apply the Woodbury identity and reformulate the equations. In contrast to the update message in normal form [Toussaint, 2009], direct inversions of $\bar{\mathbf{S}}_{t-1}$ and $\bar{\mathbf{V}}_{t+1}$ are not necessary in the canonical form and therefore, the iterative updates are numerically more stable.

Finally, in order to compute the optimal feedback controller we calculate the joint state-control posterior

$$\begin{aligned}P(\mathbf{u}_t, \mathbf{x}_t) &= P(\mathbf{u}_t, \mathbf{x}_t | z_t = 1)\\ &= \int_{\mathbf{x}_{t+1}} \alpha_t(\mathbf{x}_t)\phi_t(\mathbf{x}_t)P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)P(\mathbf{u}_t)\beta_{t+1}(\mathbf{x}_{t+1})\phi_{t+1}(\mathbf{x}_{t+1})d\mathbf{x_{t+1}},\\ &= P(\mathbf{x}_t)P(\mathbf{u}_t)\int_{\mathbf{x}_{t+1}} P(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\mathcal{N}[\mathbf{x}_{t+1}|\bar{\mathbf{v}}_{t+1}, \bar{\mathbf{V}}_{t+1}]d\mathbf{x_{t+1}}.\end{aligned}$$

The conditional distribution is given by $P(\mathbf{u}_t|\mathbf{x}_t) = P(\mathbf{u}_t, \mathbf{x}_t)/P(\mathbf{x}_t)$, and the solution is

$$P(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{u}_t|\mathbf{M}_t^{-1}(\mathbf{B}_t^T\mathbf{V}_*(\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1} - \mathbf{A}_t\mathbf{x}_t - \hat{\mathbf{a}}_t) + \mathbf{h}_t), \mathbf{M}_t^{-1}),$$

where $\mathbf{V}_* = (\mathbf{Q} + \bar{\mathbf{V}}_{t+1}^{-1})^{-1}$ and $\mathbf{M}_t = \mathbf{B}_t^T\mathbf{V}_*\mathbf{B}_t + \hat{\mathbf{H}}_t$. After a reformulation we can obtain an optimal feedback controller of the form $\mathbf{u}_t = \mathbf{o}_t + \mathbf{O}_t\mathbf{x}_t$ with

$$\mathbf{o}_t = \mathbf{M}_t^{-1}(\mathbf{B}_t^T\mathbf{V}_*\bar{\mathbf{V}}_{t+1}^{-1}\bar{\mathbf{v}}_{t+1} - \mathbf{B}_t^T\mathbf{V}_*\mathbf{a}_t + \mathbf{h}_t), \tag{B.19}$$

$$\mathbf{O}_t = -\mathbf{M}_t^{-1}\mathbf{B}_t^T\mathbf{V}_*\mathbf{A}_t. \tag{B.20}$$

Similar to Toussaint [2009], we use an iterative message passing approach where we approximate the nonlinear system by an Linear dynamics, Quadratic costs and Gaussian noise system (LQG) at the new mode of the trajectory. In Toussaint [2009], this iterative optimization approach uses a learning rate on the current modes of the belief. However, in difference to Toussaint [2009], we also need an estimate of the optimal action $\mathbf{u}_t$ in order to impose the control constraints. Using a learning rate on the control action $\mathbf{u}_t$ turned out to be very ineffective because feedback is attenuated. For this reason we will use a

Algorithm 1: Approximate Inference Control for Constrained Systems

```
Data: initial trajectory $\hat{\mathbf{x}}_{1:T}$, learning rate $\eta$
Result: $\mathbf{x}_{1:T}$ and $\mathbf{u}_{1:T-1}$
initialize $\mathbf{S}_1 = 1e10 \cdot \mathbf{I}$, $\mathbf{s}_1 = \mathbf{S}_1\mathbf{x}_1$, $k = 0$, $\hat{\mathbf{o}}_{1:T-1} = \mathbf{0}$, $\hat{\mathbf{O}}_{1:T-1} = 0 \cdot \mathbf{I}$ ;
while $L(\tau)$ not converged do
    for $t \leftarrow 1$ to $T$ do
        Linearize Model: $\mathbf{A}_t, \mathbf{a}_t, \mathbf{B}_t$ using Equation B.6
        Compute Costs: $\hat{\mathbf{H}}_t, \mathbf{h}_t, \mathbf{R}_t, \mathbf{r}_t$ using Equation B.7, B.11
    for $t \leftarrow 1$ to $T$ do
        Forward Messages: $\alpha_t(\mathbf{x}_t)$ using Equation B.13 - B.15
    for $t \leftarrow T-1$ to $1$ do
        Backward Messages: $\beta_t(\mathbf{x}_t)$ using Equation B.16 - B.18
    for $t \leftarrow 1$ to $T-1$ do
        Feedback Controller: $\mathbf{o}_t, \mathbf{O}_t$ using Equation B.19, B.20
        if $k == 0$ then
            $\mathbf{u}_t = \mathbf{o}_t + \mathbf{O}_t\mathbf{x}_t$
        else
            $\hat{\mathbf{o}}_t = (1-\eta)\hat{\mathbf{o}}_t + \eta\mathbf{o}_t$
            $\hat{\mathbf{O}}_t = (1-\eta)\hat{\mathbf{O}}_t + \eta\mathbf{O}_t$
            $\mathbf{u}_t = \hat{\mathbf{o}}_t + \hat{\mathbf{O}}_t\mathbf{x}_t$
        $\mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t$
    $k = k + 1$
```

learning rate on the feedback controller.

The complete message passing algorithm considering state and control constraints is listed in Algorithm 1. This algorithm is a straightforward implementation of Gaussian message passing in linearized systems, similar to an extended Kalman smoother.

In Toussaint [2009] or Rawlik et al. [2010] more time efficient methods are presented, where for each time-step the belief is updated until convergence in contrast to updating all messages and iterating until the intrinsic costs $L(\tau)$ converge. The computational benefits of such an approach still needs to be evaluated for our messages.

B.2 4-link robot model specifications

For the 4-link model the 8-dimensional state $\mathbf{x} = [\phi_1, \dot{\phi}_1, \phi_2, \dot{\phi}_2, \phi_3, \dot{\phi}_3, \phi_4, \dot{\phi}_4]$ is composed of the ankle, the knee, the hip and the arm positions and their velocities. In our experiments the velocities are instantaneously affected by the applied force F [Atkeson and

Joint	**Initial velocities**	**Lower Bound**	**Upper Bound**
ankle	$+1.2 \cdot 10^{-2} F$	-0.8	0.8
knee	$-7.4 \cdot 10^{-2} F$	-0.05	2.5
hip	$+5.1 \cdot 10^{-2} F$	-2.0	0.1
arm	$-4.2 \cdot 10^{-2} F$	-0.6	3.0

Table B.1: Joint angle configurations where a robot gets pushed by a force F.

Stephens, 2007]. These initial velocities and the valid joint angle range are shown in Table B.1.

In addition we use multiple initial joint angles. The ankle angle is given by $\phi_1 = -\alpha$, the knee angle by $\phi_2 = -\alpha\mathrm{f}$, the hip angle by $\phi_3 = \alpha\mathrm{f}$ and the arm joint angle is $\phi_4 = 2\alpha\mathrm{f}$, where α is sampled from $\alpha \sim \mathcal{U}[0, 7]\pi/180$ and the linear factor is uniform distributed with $\mathrm{f} \sim \mathcal{U}[-3.3, -2.3]$. Four example initial states are illustrated in Figure 2.2.

Appendix C

Appendix to Chapter 3: Learned graphical models for probabilistic planning

C.1 Dynamic movement primitives

The most prominent representation for movement primitives used in robot control are the Dynamic Movement Primitives (DMP) [Schaal et al., 2003]. We therefore used the DMPs as a baseline in our evaluations and will briefly review this approach in order to clarify differences to our work. For our experiments we implemented an extension of the original DMPs [Pastor et al., 2009], which considers an additional term in the dynamical system which facilitates generalization to different target states. For more details we refer to Schaal et al. [2003], Pastor et al. [2009].

DMPs generate multi-dimensional trajectories by the use of non-linear differential equations. The basic idea is to a use for each degree-of-freedom (DoF) of the robot a globally stable, linear dynamical system which is modulated by learnable non-linear functions f :

$$\tau\dot{z} \;=\; \alpha_z\beta_z(g-y) - \alpha_z z - \alpha_z\beta_z(g-y_1)s + f, \tau\dot{y} = z,$$

where the desired final position of the joint is denoted by g and the initial position of the joint is denoted by y_1. The variables y and $\dot{y}$ denote a desired joint position and joint velocity, which represent our movement plan. The temporal scaling factor is denoted by

τ and α_z and β_z are time constants. The non-linear function f directly modulates the derivative of the internal state variable z. Thus, f modulates the desired acceleration of the movement plan. s denotes the phase of the movement.

For each DoF of the robot an individual dynamical system, and hence an individual function f is used. The function f only depends on the phase s of a movement, which represents time, $\tau\dot{s} = -\alpha_s s$. The phase variable s is initially set to 1 and will converge to 0 for a proper choice of τ and α_s. With α_s we can modulate the desired movement speed. The function f is constructed of the weighted sum of K Gaussian basis functions Ψ_i

$$f(s) = \frac{\sum_{i=1}^{K} \Psi_i(s) w_i s}{\sum_{i=1}^{K} \Psi_i(s)}, \qquad \Psi_i(s) = \exp(-\frac{1}{2\sigma_i^2}(s - c_i)^2).$$

As the phase variable s converges to zero also the influence of f vanishes with increasing time. Hence, the dynamical system is globally stable with g as point attractor.

In our setting, only the linear weights w_i are parameters of the primitive which can modulate the shape of the movement. The centers c_i specify at which phase of the movement the basis function becomes active and are typically equally spaced in the range of s and not modified during learning. The bandwidth of the basis functions is given by σ_i^2.

Integrating the dynamical systems for each DoF results into a desired trajectory $\langle \mathbf{y}_t, \dot{\mathbf{y}}_t \rangle$ of the joint angles. We will use an inverse dynamics controller to follow this trajectory [Peters et al., 2008]. The inverse dynamics controller receives the desired accelerations $\ddot{\mathbf{q}}_{\text{des}}$ as input and outputs the control torques $\mathbf{u}$. In order to calculate the desired accelerations we use a simple decoupled linear PD-controller

$$\ddot{\mathbf{q}}_{\text{des}} = \text{diag}(\mathbf{k}_{\text{pos}})(\mathbf{y}_t - \mathbf{q}_t) + \text{diag}(\mathbf{k}_{\text{vel}})(\dot{\mathbf{y}}_t - \dot{\mathbf{q}}_t). \tag{C.1}$$

Unfortunately standard inverse dynamics control did not work in our setup because we had to deal with control limits of multi-dimensional systems. Thus, we had to use an inverse dynamics controller which also incorporates control constraints. For this reason we performed an iterative gradient ascent using the difference between the actual (using constrainted controls) and the desired accelerations $\ddot{\mathbf{q}}_{\text{des}}$ as error function. This process was stopped after at most 25 iterations.

For our comparison, we will learn the linear weights $\mathbf{w}$ for each DoF as well as the controller gains $\mathbf{k}_{\text{pos}}$ and $\mathbf{k}_{\text{vel}}$, i.e., $\boldsymbol{\theta} = [\mathbf{w}_1, \ldots, \mathbf{w}_D, \mathbf{k}_{\text{pos}}, \mathbf{k}_{\text{vel}}]$. This results into $KD + 2D$ parameters for the movement representation, where D denotes the number of DoF of the robot.

$\mathbf{K}$	α_s	α_z	β_z	τ
10	1	2	0.9	0.1

Table C.1: Via-point task: DMP movement primitive parameters.

	$\mathbf{w}$	$\mathbf{k_{pos}}$	$\mathbf{k_{vel}}$
lower bound	-100	0	0
upper bound	$+100$	100	100

Table C.2: Via-point task: DMP policy search configuration parameters.

	$d^{[1]}$	$g^{[1]}$	$\mathbf{r}^{[i]}$	$\mathbf{h}^{[i]}$
lower bound	0.05	-2	$[1, 10^{-6}]$	10^{-4}
upper bound	0.4	+2	$[10^{6}, 10^{4}]$	10^{-2}

Table C.3: Via-point task: PMP policy search configuration parameters with $i = 1, 2$.

C.2 Task settings and parameters

In this section the movement primitive parameters and constants are specified for the one-dimensional via-point task and for the humanoid balancing task.

C.2.1 One-dimensional via-point task

For the one-dimensional via-point task the parameters of the Dynamic Movement Primitives are listed in Table C.1. The valid configuration space for the policy search algorithm is listed in Table C.2. The CMA policy search algorithm has just one parameter, the exploration rate. Where the best exploration rate using DMPs for this task found was 0.05.

The limits of the parametrization of the Planning Movement Primitives (see Equation 3.4) is listed in Table C.3. For the via-point task we choose $N = 2$, where the second via-point $g^{[N]} = g_T$ was given. The exploration rate was set to 0.1 in all experiments.

C.2.2 Dynamic humanoid balancing task

The DMP parameters for the balancing task are listed in Table C.4. The policy search parameters are the same like for the via-point task, Table C.2. The exploration rate was

$\mathbf{K}$	α_s	α_z	β_z	τ
10	1	5	5	1

Table C.4: Balancing task: DMP movement primitive parameters.

$\phi_{0_{\mathbf{arm}}}$	**arm**	**hip**	**knee**	**ankle**
1	-2.39	7.56	-8.54	1.42
0.5	-3.61	6.12	-7.9	1.32
0.2	-4.15	5.29	-7.52	1.25
0	-4.27	5.09	-7.43	1.24
-0.2	-4.15	5.29	-7.52	1.25
-0.4	-3.82	5.8	-7.75	1.29
-0.6	-3.43	6.38	-8	1.34

Table C.5: Initial velocities (multiplied by $10^2/F$) for different initial states of the arm joint $\phi_{0_{\mathrm{arm}}}$.

	$d^{[1]}$	$\mathbf{r}^{[i]}$	$\mathbf{h}^{[i]}$
lower bound	0.1	10^{-2} for angles and 10^{-4} for velocities	$10^{-9}\mathbf{1}$
upper bound	4.6	10^{4} for angles and 10^{2} for velocities	$10^{-3}\mathbf{1}$

Table C.6: Balancing task: PMP policy search configuration parameters with $i = 1, 2$. Vector $\mathbf{1}$ denotes a 4-dimensional column vector, where all elements are equal to 1.

set to 0.1.

The PMPs were again evaluated with $N = 2$ via-points, where the second via-point $g^{[N]} = g_T$ (the up-right robot posture) was given and for the first via-point the valid joint angle configuration is shown in Table 1 in [Rückert and Neumann, 2012]. The exploration rate was 0.1 and the policy search algorithm configuration is listed in Table C.6.

In the generalization experiment we applied the same learned policy of the 4-link balancing task to different initial states. For different initial arm joint configurations the push modulated with F resulted in different initial joint velocities, which are shown in Table C.5.

Appendix D

Appendix to Chapter 4: Dynamic movement primitives with shared synergies

D.1 Parameter settings

For the via-point task the parameter settings for learning are shown in Table D.1. Initial parameter values and parameter settings for policy search for the biped walker task are shown in Table D.2 and in Table D.3. In Table D.4 we list the learning settings for the multi-directional reaching task using a musculoskeletal model of a human arm.

Table D.1: Parameter settings for the discrete via-point task. In brackets are the variable names for the dynamic movement primitives, which are used for comparison.

Parameter	**Range**	**Initial Value**
a, (w)	$\in [-5, 5]$	$a^0 = 0$
μ, (μ)	$\in [0, 1]$	μ^0 spaced linearly in $[0, 1]$
h, (h)	$\in [0.01, 1]$	$h^0 = 0.1$
β	$\in [0, 100]$	$\beta^0 = 1$

D.2 Characteristic muscle parameters

The implemented muscles and their characteristic parameters are shown in Table D.5.

Table D.2: Biped walker setting of pre-optimized quantities.

Variable	**Value**
$\mathbf{q}_1$	$[3.5, 4.4, -0.07, -0.5, -0.7,$ $-1.5, -0.6, -1.0, 0.3, -0.4]$
$\mathbf{g}$	$[2.8, 4.5, -0.3, -1.8]$
$\mathbf{k}_{pos}$	$[632.5, 885.6, 463.4, 643.7]$
$\mathbf{k}_{vel}$	$[14.5, 13.2, 38.6, 40.6]$
$\mathbf{q}_{min}$	$[2.8, 2.8, -2.6, -2.6, -1.04]$
$\mathbf{q}_{max}$	$[4.7, 4.7, 0, 0, 1.0]$

Table D.3: Policy search parameter settings for the rhythmic walking task. In brackets are the variable names for the dynamic movement primitives.

Parameter	**Range**	**Initial Value**
a, (w)	$\in [-2, 2]$	$a^0 = 0$
μ, (μ)	$\in [0.8\mu^0, 1.2\mu^0]$	μ^0 spaced linearly in $[0, 2\pi]$
h, (h)	$\in [0.1, 10]$	$h^0 = 1$
β	$\in [0, 2]$	$\beta^0 = 1$
Δs	$\in [0, 0.5]$	$\Delta s^0 = 0$

Table D.4: Parameter settings for the multi-directional reaching task.

Parameter	**Range**	**Initial Value**
a	$\in [-1, 1]$	$a^0 = 0.5$
μ	$\in [0.3, 0.7]$	μ^0 spaced linearly in $[0.3, 0.7]$
h	$\in 8[10^{-5}, 10^{-3}]$	$h^0 = 8 \cdot 10^{-4}$
β	$\in [0, 1]$	$\beta^0 = 1$
Δs	$\in [-0.2, 0.2]$	$\Delta s^0 = 0$

Table D.5: Characteristic muscle parameters of an upper limb model taken from Garner and Pandy [2001]. The tendon slack length is denoted by L_s^T, the maximum isometric force by F_0^M, the optimal fiber length by L_0^M, and the muscle pennation angle by α. To increase the reachable space, we adapted the tendon slack length L_s^T of a small number of muscles (bold numbers versus the original values in brackets).

	L_0^M [cm]	L_s^T [cm]	F_0^M [N]	α [rad]
anterior deltoid (DeltA)	14.68	**9.3** (1.64)	277.48	0
posterior deltoid (DeltP)	17.02	5.93	567.15	0
latissimus dorsi-thoracic (LatDors)	34.87	14.75	173.43	0
pectoralis major-clav (PectClav)	22.65	0.45	342.46	0
triceps brachii-long (TrLong)	15.24	**25.05** (19.05)	629.21	0.26
triceps brachii-lateral (TrLat)	6.17	19.64	1268.87	0.26
triceps brachii-medial (TrMed)	4.9	**18.19** (12.19)	619.67	0.26
biceps-long (BicLong)	15.36	**32.93** (22.93)	392.91	0.17
biceps-short (BicShort)	13.07	**26.98** (22.98)	461.76	0.17
brachialis (BRA)	10.28	**9.75** (1.75)	853	0.17
brachioradialis (BrRad)	27.03	6.04	101.58	0.08

Bibliography

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first International Conference on Machine learning*, (ICML 2004), pages 1–8, New York, NY, USA, 2004. ACM.

Cristiano Alessandro, JuanPablo Carbajal, and Andrea d'Avella. Synthesis and adaptation of effective motor synergies for the solution of reaching tasks. In *From Animals to Animats (SAB 2012)*, volume 7426 of *Lecture Notes in Computer Science*, pages 33–43, Odense, Denmark, 2012. ISBN 978-3-642-33092-6.

R. W. Angel. Electromyographic patterns during ballistic movement of normal and spastic limbs. *Brain Research*, 99(2):387–392, 1975. ISSN 0006-8993. doi: 10.1016/0006-8993(75)90042-6.

C. G. Atkeson, A. W. Moore, and S. Schaal. Locally Weighted Learning for Control. *Artificial Intelligence Review*, 11:75–113, 1997.

C.G. Atkeson and B. Stephens. Multiple balance strategies from one optimization criterion. In *Proceedings of the 7th International Conference on Humanoid Robots*, pages 57–64, Pittsburgh, USA, 2007.

A. Berardelli, M. Hallett, J. C. Rothwell, R. Agostino, M. Manfredi, P. D. Thompson, and C. D. Marsden. Single-joint rapid arm movements in normal subjects and in patients with motor disorders. *Brain*, 119(2):661–674, April 1996. ISSN 1460-2156. doi: 10.1093/brain/119.2.661.

Max Berniker, Anthony Jarc, Emilio Bizzi, and Matthew C. Tresch. Simplified and effective motor control based on muscle synergies to exploit musculoskeletal dynamics. *Proceedings of the National Academy of Sciences (PNAS)*, 106(18):7601–7606, 2009. doi: 10.1073/pnas.0901512106.

N. A. Bernstein. *The Co-ordination and regulation of movements.* Pergamon Press Ltd., first english edition edition, 1967.

Dimitri P Bertsekas and Steven E Shreve. *Stochastic optimal control: The discrete time case*, volume 139. Academic Press New York, 1978.

Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, 1995.

E. Bizzi, V.C.K Cheung, A. d'Avella, P. Saltiel, and Tresch M. Combining modules for movement. *Brain Research Reviews*, 57(1):125–133, 2008. ISSN 0165-0173. doi: 10.1016/j.brainresrev.2007.08.004.

Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. *Journal of Machine Learning Research - Proceedings Track*, 15:182–189, 2011.

E.K. Chadwick, D. Blana, A.J. van den Bogert, and R.F. Kirsch. A real-time, 3-d musculoskeletal model for dynamic simulation of arm movements. *Biomedical Engineering, IEEE Transactions on*, 56(4):941–948, 2009. ISSN 0018-9294. doi: 10.1109/TBME.2008.2005946.

Manu Chhabra and Robert A. Jacobs. Properties of synergies arising from a theory of optimal motor behavior. *Neural Computation*, 18(10):2320–2342, 2006. doi: 10.1162/neco.2006.18.10.2320.

E. Chiovetto, B. Berret, and T. Pozzo. Tri-dimensional and triphasic muscle organization of whole-body pointing movements. *Neuroscience*, 170(4):1223 – 1238, 2010. ISSN 0306-4522. doi: 10.1016/j.neuroscience.2010.07.006.

Enrico Chiovetto, Andrea d'Avella, and Martin Giese. A unifying framework for the identification of kinematic and electromyographic motor primitives. In *Abstracts of Neural Control of Movement Conference (NCM 2013)*, San Juan, Puerto Rico, USA, 2013. Conference talk.

A. d'Avella and E. Bizzi. Shared and specific muscle synergies in natural motor behaviors. *Proceedings of the National Academy of Sciences (PNAS)*, 102(8):3076–3081, 2005. doi: 10.1073/pnas.0500199102.

A. d'Avella and M. C. Tresch. Modularity in the motor system: decomposition of muscle patterns as combinations of time-varying synergies. In *Advances of Neural Information*

Processing Systems, (NIPS 2001), pages 141–148, Vancouver, British Columbia, Canada, 2001. MIT Press.

A. d'Avella, A. Portone, L. Fernandez, and F. Lacquaniti. Control of fast-reaching movements by muscle synergy combinations. *The Journal of Neuroscience*, 26(30):7791–7810, 2006. doi: 10.1523/jneurosci.0830-06.2006.

Andrea d'Avella and Dinesh K Pai. Modularity for sensorimotor control: Evidence and a new prediction. *Journal of Motor Behavior*, 42(6):361–369, 2010. doi: 10.1080/00222895.2010.526453.

Andrea d'Avella, Philippe Saltiel, and Emilio Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6(3):300–308, March 2003. doi: 10.1038/nn1010.

S.L. Delp, F.C. Anderson, A.S. Arnold, P. Loan, A. Habib, C.T. John, E. Guendelman, and D.G. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *Biomedical Engineering, IEEE Transactions on*, 54(11):1940–1950, nov. 2007. ISSN 0018-9294. doi: 10.1109/TBME.2007.901024.

Nadia Dominici, Yuri P. Ivanenko, Germana Cappellini, Andrea d'Avella, Vito Mondi, Marika Cicchese, Adele Fabiano, Tiziana Silei, Ambrogio Di Paolo, Carlo Giannini, Richard E. Poppele, and Francesco Lacquaniti. Locomotor primitives in newborn babies and their development. *Science*, 334(6058):997–999, 2011. doi: 10.1126/science.1210617.

Ahmet Erdemir, Scott McLean, Walter Herzog, and Antonie J. van den Bogert. Model-based estimation of muscle forces exerted during movements. *Clinical biomechanics (Bristol, Avon)*, 22(2):131–154, 2007. doi: 10.1016/j.clinbiomech.2006.09.005.

B. Garner and M. Pandy. Musculoskeletal model of the upper limb based on the visible human male dataset. *Computer Methods in Biomechanics and Biomedical Engineering*, 4(2):93–126, 2001. doi: 10.1080/10255840008908000.

Martin A. Giese, Albert Mukovskiy, Aee-Ni Park, Lars Omlor, and Jean-Jacques E. Slotine. Real-time synthesis of body movements based on learned primitives. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, volume 5604 of *Lecture Notes in Computer Science*, pages 107–127. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03060-4.

M. Hallett, B. T. Shahani, and R. R. Young. EMG analysis of stereotyped voluntary movements in man. *Journal Of Neurology, Neurosurgery, And Psychiatry*, 38(12):1154–1162, 1975. ISSN 00223050.

N. Hansen, S.D. Muller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003. doi: 10.1162/106365603321828970.

Helmut Hauser, Auke Jan Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105(5-6):355–370, 2011.

V. Heidrich-Meisner and C. Igel. Neuroevolution Strategies for Episodic Reinforcement Learning. *Journal of Algorithms*, 64(4):152–168, oct 2009a.

V. Heidrich-Meisner and C. Igel. Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search. In *Proceedings of the 26th International Conference on Machine Learning*, pages 401–408, Montreal, Canada, 2009b. ACM.

Kappen H.J., Gómez V., and Opper M. Optimal control as a graphical model inference problem. *Machine Learning*, pages 1–24, 2012. ISSN 0885-6125.

Katherine R.S. Holzbaur, Wendy M. Murray, and Scott L. Delp. A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control. *Annals of Biomedical Engineering*, 33:829–840, 2005. ISSN 0090-6964. doi: 10.1007/s10439-005-3320-7.

Mark W Howe, Patrick L Tierney, Stefan G Sandberg, Paul EM Phillips, and Ann M Graybiel. Prolonged dopamine signalling in striatum signals proximity and value of distant rewards. *Nature*, 500(7464):575–579, 2013.

Fumiya Iida and Rolf Pfeifer. Sensing through body dynamics. *Robotics and Autonomous Systems*, 54(8):631–640, 2006.

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. doi: 10.1162/NECO_a_00393.

Auke Jan Ijspeert and Stefan Schaal. Learning Attractor Landscapes for Learning Motor Primitives. In *Advances in Neural Information Processing Systems 15*, (NIPS 2003), pages 1523–1530. MIT Press, Cambridge, MA, 2003.

Y. P. Ivanenko, R. E. Poppele, and F. Lacquaniti. Five basic muscle activation patterns account for muscle activity during human locomotion. *The Journal of Physiology*, 556 (1):267–282, 2004. doi: 10.1113/jphysiol.2003.057174.

Bert Kappen, Vicenç Gómez, and Manfred Opper. Optimal Control as a Graphical Model Inference Problem. *Computing Research repository (CoRR)*, arXiv:0901.0633, 2009.

H. J. Kappen. An Introduction to Stochastic Control Theory, Path Integrals and Reinforcement Learning. In *Cooperative Behavior in Neural Systems*, volume 887 of *American Institute of Physics Conference Series*, pages 149–181, Granada, Spain, 2007.

S. M. Khansari-Zadeh and Aude Billard. Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. *IEEE Transaction on Robotics*, 27(5):943–957, 2011.

J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):171–203, 2011. doi: 10.1007/s10994-010-5223-6.

Jens Kober, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust robot movements to new situations. In *Proceedings of the 2010 Robotics: Science and Systems Conference (RSS 2010)*, pages 2650–2655, Zaragoza, Spain, 2010.

J. Kuffner and S. LaValle. RRT-Connect: An efficient Approach to Single-Query Path Planning. In *Proceedings of the 2000 IEEEE International Conference on Robotics and Automation ICRA, San Francisco,CA*, pages 995–1001, April 2000.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics*, (ICINCO 2004), pages 222–229, Setúbal, Portugal, 2004.

Daniel B. Lockhart and Lena H. Ting. Optimal sensorimotor transformations for balance. *Nature Neuroscience*, 10(10):1329–1336, October 2007. doi: 10.1038/nn1986.

J. L. McKay and L. H. Ting. Optimization of muscle activity for task-level goals predicts complex changes in limb forces across biomechanical contexts. *PLoS Computational Biology*, 8(4):e1002465, 2012. doi: 10.1371/journal.pcbi.1002465.

F. Meier, E. Theodorou, F. Stulp, and S. Schaal. Movement segmentation using a primitive library. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (IROS 2011), pages 3407–3412, San Francisco, CA, USA, 2011.

Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar. Adaptive Optimal Feedback Control with Learned Internal Dynamics Models. In *From Motor Learning to Interaction Learning in Robots*, pages 65–84. Springer-Verlag, 2010. ISBN 978-3-642-05180-7.

Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013. doi: 10.1177/0278364912472380.

Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79–91, 2004. doi: 10.1016/j.robot.2004.03.003.

R. R. Neptune, D. J. Clark, and S. A. Kautz. Modular control of human walking: A simulation study. *Journal of Biomechanics*, 42(9):1282–1287, 2009. doi: 10.1016/j.jbiomech.2009.03.009.

G. Neumann, W. Maass, and J. Peters. Learning Complex Motions by Sequencing Simpler Motion Templates. In *Proceedings of the 26th International Conference on Machine Learning*, (ICML 2009), pages 753–760, Montreal, Canada, 2009.

D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf. Learning inverse dynamics: A comparison. In *16th European Symposium on Artificial Neural Networks*, (ESANN 2008), pages 13–18, Bruges, Belgium, 2008a.

D Nguyen-Tuong, M Seeger, and J Peters. Local Gaussian Process Regression for Real Time Online Model Learning and Control. In *Proceedings of 22nd Annual Conference on Neural Information Processing Systems*, (NIPS 2008), pages 1193–1200, Vancouver, BC, Canada, 2008b.

A. Overduin, A. d'Avella, J. Roh, and E. Bizzi. Modulation of muscle synergy recruitment in primate grasping. *The Journal of Neuroscience*, 28(4):880–892, 2008. doi: 10.1523/JNEUROSCI.2869-07.2008.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and Generalization of Motor Skills by Learning from Demonstration. In *International Conference on Robotics and Automation (ICRA 2009)*, Kobe, Japan, 2009.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008. doi: 10.1016/j.neunet.2008.02.003.

J. Peters, M. Mistry, F. E. Udwadia, J. Nakanishi, and S. Schaal. A Unifying Methodology for Robot Control with Redundant DOFs. *Autonomous Robots*, 24(1):1–12, 2008.

Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: A new view of intelligence*. The MIT Press, October 2006. ISBN 0262162393.

Rolf Pfeifer, Max Lungarella, and Fumiya Iida. Self-organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088–1093, 2007. doi: 10.1126/science.1145803.

B.E. Pfeiffer and D.J. Foster. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, (7447):74–79, 2013. doi: 10.1038/nature12112.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. An approximate inference approach to temporal optimization in optimal control. In *Proceedings of 24nd Annual Conference on Neural Information Processing Systems*, (NIPS 2010), pages 2011–2019, Vancouver, BC, Canada, 2010.

Elmar Rückert and Andrea d'Avella. Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in Computational Neuroscience (Special Issue on Modularity in motor control: from muscle synergies to cognitive action representation)*, 7(138), 2013. ISSN 1662-5188. doi: 10.3389/fncom.2013.00138.

Elmar A. Rückert and Gerhard Neumann. Stochastic Optimal Control Methods for Investigating the Power of Morphological Computation. *Artificial Life*, 19(1):1–17, 2012.

Elmar A. Rückert, Gerhard Neumann, Marc Toussaint, and Wolfgang Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience*, 6(97), 2013. ISSN 1662-5188. doi: 10.3389/fncom.2012.00097.

Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Jan Ijspeert. Learning Movement Primitives. In *International Symposium on Robotics Research*, (ISRR 2003), pages 561–572, Lucerne, Switzerland, 2003.

L.M. Schutte, M.M. Rodgers, F.E. Zajac, and R.M. Glaser. Improving the efficacy of electrical stimulation-induced leg cycle ergometry: an analysis based on a dynamic musculoskeletal model. *Rehabilitation Engineering, IEEE Transactions on*, 1(2):109–125, jun 1993. ISSN 1063-6528. doi: 10.1109/86.242425.

F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, May 2010. doi: 10.1016/j.neunet.2009.12.004.

Ajay Seth, Michael Sherman, Jeffrey A. Reinbolt, and Scott L. Delp. Opensim: a musculoskeletal modeling and simulation framework for in silico investigations and exchange. *Procedia International Union of Theoretical and Applied Mathematics (IUTAM 2011)*, 2(0):212–232, 2011. doi: 10.1016/j.piutam.2011.04.021.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Boston, MA, 1998. ISBN 0262193981.

Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT, 2005.

E. Theodorou, J. Buchli, and S. Schaal. Reinforcement Learning of Motor Skills in High Dimensions: a Path Integral Approach. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2397–2403, Anchorage, Alaska, USA, 2010.

E. Todorov and M. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5:1226–1235, 2002.

E. Todorov and Weiwei Li. A generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems. In *Proceedings of the 24th American Control Conference*, volume 1 of *(ACC 2005)*, pages 300 – 306, Portland, Oregon, USA, 2005.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th International Conference on Machine Learning*, (ICML 2009), pages 1049–1056, Montreal, Canada, 2009.

Marc Toussaint. Lecture notes: Gaussian identities. Technical report, Freie Universität Berlin, 2011. URL `http://userpage.fu-berlin.de/mtoussai/notes/gaussians.pdf`.

J. Trommershauser, S. Gepshtein, L. Maloney, and M. Banks. Optimal Compensation for Changes in Task-Relevant Movement Variability. *Journal of Neuroscience*, 25:7169–7178, 2005.

S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, dec 2005.

Oskar von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992.

E. R. Westervelt, Gabriel Buche, and J. W. Grizzle. Experimental validation of a framework for the design of controllers that induce stable walking in planar bipeds. *The International Journal of Robotics Research*, 23(6):559–582, 2004. doi: 10.1177/0278364904044410.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Episodic Reinforcement Learning by Logistic Reward-Weighted Regression. In *Proceedings of the 18th international conference on Artificial Neural Networks, Part I*, (ICANN 2008), pages 407–416, Berlin, Heidelberg, 2008. Springer-Verlag.

Ronald J.. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. doi: 10.1023/A:1022672621406.

Jack M. Winters and L. Stark. Analysis of fundamental human movement patterns through the use of in-depth antagonistic muscle models. *Biomedical Engineering, IEEE Transactions on*, BME-32(10):826–839, 1985. ISSN 0018-9294. doi: 10.1109/TBME.1985.325498.

R.J. Wood. Design, fabrication, and analysis of a 3dof, 3cm flapping-wing MAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (IROS 2007), pages 1576–1581, Oct-Nov 2 2007.

F. E. Zajac. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering*, 17(4):359–411, 1989. ISSN 0278-940X.

Marc Ziegler, Fumiya Iida, and Rolf Pfeifer. "Cheap" underwater locomotion: Roles of morphological properties and behavioral diversity. In *9th International Conference on Climbing and Walking Robots*, (CLAWAR 2006), Brussels, Belgium, September 2006.