Robot Skill Learning

Jan Peters, Katharina Mülling, Jens Kober, Duy Nguyen-Tuong, Oliver Krömer [1]

Abstract. Learning robots that can acquire new motor skills and refine existing ones have been a long standing vision of robotics, artificial intelligence, and the cognitive sciences. Early steps towards this goal in the 1980s made clear that reasoning and human insights will not suffice. Instead, new hope has been offered by the rise of modern machine learning approaches. However, to date, it becomes increasingly clear that off-the-shelf machine learning approaches will not be adequate for robot skill learning as these methods often do not scale into the high-dimensional domains of manipulator and humanoid robotics, nor do they fulfill the real-time requirement of the domain. As an alternative, we propose to divide the generic skill learning problem into parts that can be well-understood from a robotics point of view. After designing appropriate learning approaches for these basic components, these will serve as the ingredients of a general approach to robot skill learning. In this paper, we discuss our recent and current progress in this direction. As such, we present our work on learning to control, learning elementary movements, as well as our steps towards the learning of complex tasks. We show several evaluations using both real robots as well as physically realistic simulations.

1 Introduction

Despite an increasing number of motor skills exhibited by manipulator and humanoid robots, the general approach to the generation of such motor behaviors has changed little over the last decades [1]. The roboticist models the task as accurately as possible and uses human understanding of the required motor skills in order to create the desired robot behavior, as well as to eliminate all uncertainties of the environment. In most cases, such a process boils down to recording a desired trajectory in a pre-structured environment with precisely placed objects. If inaccuracies remain, the engineer creates exceptions using human understanding of the task. Such highly engineered approaches are feasible in highly structured industrial or research environments. However, it is obvious that if robots should ever leave factory floors and research environments, we will need to reduce the strong reliance on hand-crafted models of the environment and the robots. Instead, we need a general framework which allows us to use compliant robots that are designed for interaction with less structured and uncertain environments in order to reach domains outside industry. Such an approach cannot rely solely on human knowledge but instead has to be acquired from data generated both by human demonstrations of the skill as well as trial & error of the robot.

The tremendous progress in machine learning over the last decades offers us the promise of less human-driven approaches to motor skill acquisition. However, despite offering the most general methods for data-driven acquisition of motor skills, generic machine learning techniques (which do not rely on an understanding of motor systems) often do not scale into the realt-time domain of manipulator or humanoid robotics due to their high dimensionality. Therefore, instead of attempting to apply a standard machine learning framework to motor skill aquisition, we need to develop approaches suitable for this particular domain. To cope with the complexities involved in robot skill learning, the inherent problems of task representation, learning and execution should be addressed separately in a coherent framework employing a combination of imitation, reinforcement and model learning. The advantage of such a concerted approach is that it allows the separation of the main problems of motor skill acquisition, refinement and control. Instead of either having an unstructured, monolithic machine learning approach or creating hand-crafted approaches with pre-specified trajectories, we are capable of acquiring skills from demonstrations and represented as policies which become refined by trial and error (as discussed in Section 4). Additionally, we can learn how to activate and adapt the task-related parameters in order to achieve more complex tasks as discussed in Section 5. Finally, using learning-based approaches, we can achieve accurate control without accurate analytical models of the complete system as discussed in Section 3.

2 Our Skill Learning Framework

In order to create a robot skill learning framework that is sufficiently general, we need to discuss three basic components for such an approach. For this, *a general representation* is required that can encapsulate elementary and frequently used motions. We need to be able to *learn* these motions efficiently, and a *supervisory module* must be able to use these basic elements. Finally, *execution* is required that can adapt to changes in the environment. The resulting control architecture is shown in Figure 1. Let us now briefly discuss each of these aspects in the remainder of this section.

Motor Primitives. For the representation of motor skills, we can rely on the insight that humans, while being capable of performing a large variety of complicated movements, restrict themselves to a smaller amount of primitive motions [3]. As suggested by Ijspeert et

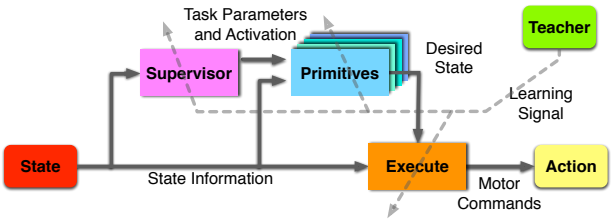


Figure 1. This figure illustrates the generic components of a motor skill learning system, i.e., the *supervisor system* activates *motor primitives* and sets their task parameters. These elementary movements are *executed* by a learned motor control law. The learning signals are provided with the help of a teacher or scoring system.

[1]Technische Universität Darmstadt, and Max Planck Institute for Intelligent Systems, email: mail@jan-peters.net

al. [4], such primitive movements can be represented by nonlinear dynamic systems. As a result, we may represent elementary tasks by elementary policies of the type[2]

$$\dot{\mathbf{x}}^d = \boldsymbol{\pi}_i(\mathbf{x}^d, \mathbf{x}, t, \boldsymbol{\rho}_i) \tag{1}$$

where $\mathbf{x}^d$ is the internal state of the system, t denotes the time, $i \in \{1, 2, \ldots, n\}$ is the index of the motor primitive in a library of movements, and task parameters $\boldsymbol{\rho}_i = [\boldsymbol{\theta}_i, d, \mathbf{g}, \mathbf{A}, \ldots]$ determine the shape of movement primitive i using $\boldsymbol{\theta}_i \in \mathbb{R}^L$, duration d, goal $\mathbf{g}$ and amplitude $\mathbf{A}$, etc, of the motion. The resulting system is linear in the shape parameters $\boldsymbol{\theta}_i$ and can therefore be learned efficiently. They are robust towards perturbations and, as they are time-continuous, they are well-suited for control.Both primitives in task-spaces as well as in joint-space can be learned. A key element of the Ijspeert formulation is that the shape is solely determined by $\boldsymbol{\theta}_i$ but that it is invariant under changes of duration, goal or amplitude of the movement. Hence, the resulting primitives can be reused efficiently by a higher-level supervisory module.

Supervisor. The supervisory level is an increasingly hot topic for research as it allows the usage of the motor primitive policies $\boldsymbol{\pi}_i$ in a multitude of novel ways. First, it may reuse a movement primitive with the same shape in various situations by simply modifying the duration, the goal, the amplitude or other task parameters. As we will see in Section 5.1, it is straightforward to *learn subgoal functions* that set the task context variables based on the external state. The supervisory level allows the genereralization of learned movements by creating a *mixture of motor primitives*, i.e., a new movement policy $\boldsymbol{\pi}$ results from a convex combination of existing movements $\boldsymbol{\pi}_i$. In the same context, we can treat the selection of motor primitives. Here, the primitive with the maximal weight is activated while in generalization several primitives using this state-dependent weight. These topics are discussed in Section 5.2. Other tasks of the supervisor are *sequencing* motion primitives as well as *blending* the transitions between them and the *superposition* of different movements.

Execution. The execution of a motor primitive $\boldsymbol{\pi}_i$ on compliant robot systems, which are safe in the interaction with humans, adds another level of complexity. It requires that we generate motor commands $\mathbf{u} = \eta(\dot{\mathbf{x}}^d, \mathbf{x}^d, \mathbf{x})$ so that the motor primitives get executed precisely while not introducing large feedback gains. If accomplished using hand-crafted control laws, the quality of the analytical models is essential and, low gain control can only be achieved with very accurate models. Hence, in the presence of unmodeled, time-variant nonlinearities resulting from stiction, cable drives, or the hydraulic tubes, it will become essential to learn accurate models and to adapt them online. We are developing efficient real-time regression methods for online model learning based on the state-of-the-art in machine learning, see Section 3.1. If a motor primitive is only acting in a limited subspace, it can often be better to directly learn a mapping from primitives and states to motor command. While learning such an operational space control is no longer a standard regression problem, it can still be solved using a reward-weighted regression when using insights from mechanics.

Learning is required for acquiring and refining the motor primitives discussed before. However, it is also needed for adapting the execution to changes in the environement and to learn the supervisory module, as can be observed in Figure 1. Learning motor primitives is achieved by adapting the parameters $\boldsymbol{\theta}_i$ of motor primitive i. The high dimensionality of our domain prohibits the exploration of the complete space of all admissible motor behaviors, rendering the application of many standard machine learning techniques impossible as these require exhaustive exploration. Instead, we have to rely on a combination of imitation and reinforcement learning to acquire motor skills where supervised learning is used to obtain the initialization of the motor skill, while reinforcement learning is used in order to improve it. Therefore, the aquisition of a novel motor task consists out of two phases, i.e., the 'learning robot' attempts to reproduce the skill acquired through supervised learning and then improve the skill from experience by trial-and-error through reinforcement learning. See Section 4 for more details on this part. When learning to execute, we are interested in two topics: learning better models of the robots dynamics in order to improve the model-based control laws of the system (as discussed in Section 3.1), and to directly learn policies that transform task-space motor primitives policies into motor command (see Section 3.2). The supervisory layer poses a variety of learning problems such learning mappings from states to motor primitive task parameters (see Section 5.1), learning activation functions for selection and generalization of motor primitives (see Section 5.2), sequencing, blending and superposition of primitives, as well as parsing longer trajectories into motor primitive automata (see [7]) or determining how many movement primitives might be included in a data set [8].

These components allow us to create a robot skill learning framework in a bottom-up manner wherein we can understand each component well from an analytical robotics point of view.

3 Learning for Control

Bringing anthropomorphic robots into human daily life requires backdrivable robots with compliant control in order to ensure safe interactions with human beings. In contrast, traditional industrial robots employ high control gains which results in an inherent stiffness and, thus, are ill-suited for this aim. To achieve accurate but compliant tracking, it is essential to predict the torques required for the current movement accurately. It is well-known that for sufficiently complex robots (e.g., humanoids, service robots), the standard rigid body dynamics (RBD) models no longer describe the dynamics properly, and data-driven approximation methods become a promising alternative. Using modern machine learning techniques has a multitude of advantages ranging from higher precision torque prediction to adaptation with online learning if the dynamics are altered.

In this section, we will discuss two learning-to-control problems, i.e., learning models for control in Section 3.1 and learning operational space control in Section 3.2.

3.1 Learning Models for Control

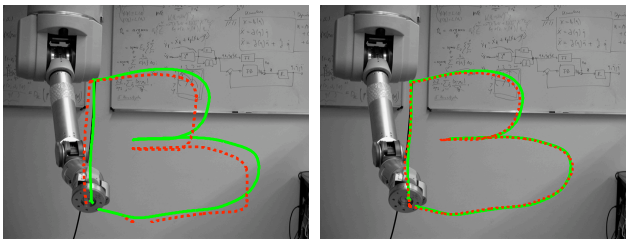
In theory, learning models of the robot dynamics is a straightforward and well-defined regression problem, wherein we can observe joint angles $\mathbf{q}$, joint velocities $\dot{\mathbf{q}}$, joint accelerations $\ddot{\mathbf{q}}$ and motor commands $\mathbf{u}$. We intend to infer the unique mapping $\mathbf{f}$ from state variables $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]$ and $\dot{\mathbf{x}}$ to motor commands $\mathbf{u}$ of which we have some prior knowledge[3]

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{G}(\mathbf{q}) + \boldsymbol{\varepsilon}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$$

with mass matrix $\mathbf{M}(\mathbf{q})$, coriolis and centrifugal forces $\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q})$, gravity $\mathbf{G}(\mathbf{q})$ and the unmodeled nonlinearities $\boldsymbol{\varepsilon}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q})$.

[2] Note that Equation (1) is in state-space formulation and, in fact, a second order system.

[3] We can in fact straightforwardly use this knowledge as described in [6].



(a) RBD Model (b) Offline Learned Model

Figure 2. This figure exhibits the effects of offline and online learning in low-gain control. The green line shows the trajectory of the letter B (previously exhibited by haptic input) as a reference trajectory and the robot is supposed to reproduce this trajectory with reproduction shown as a dashed red line. In (a), a standard control law using an analytical model provided by the manufacturer Barrett is shown. In (b), local GP (LGP) have been learned based on letter A and improve online while executing letter B. As a result, there is an improved tracking performance.

However, despite being a well-posed problem, and contrary to all progress in machine learning, online learning of robot dynamics still poses a tremendous technical challenge for any learning method. It has to deal with an endless stream of high-dimensional data while learning needs to take place in real-time at sampling rates of approximately 100Hz. While modern machine learning approaches such as Gaussian process regression (GPR) and support vector regression (SVR), yield significantly higher accuracy than traditional RBD models, their computational requirements can become prohibitively costly as they grow with number of data points. Thus, it is infeasible to simply use off-the-shelf regression techniques and the development of domain-appropriate versions of these methods is essential in order to make progress in this direction [5].

One possibility for reducing the computational cost is the partitioning of the data such that only the regionally interesting data is included in a local regression and, subsequently, combining these local predictions into a joint prediction. This approach was inspired by LWPR [2], which employs linear models. Using the more powerful Gaussian process models, we can achieve a higher prediction accuracy with less tuning of the algorithm. As a result of the localization and the resulting smaller local models, we can reach a significantly higher learning and prediction speed than for standard kernel regression techniques while having a comparable accuracy. While our approach is not as fast as LWPR, it has a significantly improved prediction accuracy in comparison and requires less manual tuning of the hyperparameters of the algorithm. The resulting method is called *Local GPR* or LGP [5] as it employs Gaussian process regression (GPR) for learning each local model i using

$$\hat{u}_i^i = \mathbf{k}^{i\,T}\,(\mathbf{K}^i + \sigma_n^2\mathbf{I})^{-1}\mathbf{U}_i = \mathbf{k}^{i\,T}\boldsymbol{\alpha}_i,$$

where u_j^i is the torque for joint j predicted by model i, $\mathbf{K}^i$ is the kernel matrix with $K_{ml}^i = k(\mathbf{x}_m^i, \mathbf{x}_l^i)$, the kernel vector $\mathbf{k}^i$ with $k_m^i = k(\mathbf{x}, \mathbf{x}_l^i)$ between the new input $\mathbf{x}$ and the stored data points $\mathbf{x}_l$, as kernel k a Gaussian kernel is employed (however, Matern kernels and rigid-body kernels have been used successfully in this context), past actions $\mathbf{U}_i$ and the so-called prediction vector $\mathbf{a}_i$. This prediction vector can be updated incrementally which is computationally feasible as we only have small local models. A weighted average allows the combination of the local models

$$\hat{\mathbf{u}} = \frac{\sum_{i=1}^n w_i \hat{\mathbf{u}}_i}{\sum_{i=1}^n w_i},$$

where the weights $w_i = \exp(-0.5\sigma_i^{-2}\,\|\mathbf{x} - \mathbf{c}_i\|^2)$ are used to reweight the model i in accordance to the proximity of the input $\mathbf{x}$ to the centers of the model $\mathbf{c}_i$.

Due to the reduced computational cost, this approach was successfully implemented on a real Barrett WAM arm where it was able to improve the tracking performance while learning online. When using the learned model in a computed torque setup where the learned model is employed to predict the required torque while stabilized by a linear low-gain control law. It can be shown that the learned model outperforms RBD models and, due to the online improvement, also most global regression techniques. Figure 2 exhibits the difference between these methods. In Figure 2(a), the performance of a low-gain feedback control law with a RBD model is shown for tracking the letter B, Figure 2(b) shows an online-learned model. For details on the approach please refer to [5].

3.2 Learning Operational Space Control

Operational space control (OSC) is one of the most elegant approaches to task control for complex, redundant robots. Its potential for dynamically consistent control, compliant control, force control, and hierarchical control has not been exhausted to date. Applications of OSC range from basic end-effector control of manipulators [16] to balancing and gait execution for humanoid robots [19]. If the robot model is accurately known, operational space control is well-understood and a variety of different solution alternatives are available. However, as many new robotic systems are supposed to operate safely in human environments, compliant, low-gain operational-space control is desired. As a result, the practical use of operational space control becomes increasingly difficult in the presence of unmodeled nonlinearities, leading to reduced accuracy or even unpredictable and unstable null-space behavior in the robot system.

Learning control methods are a promising potential solution to this problem. However, learning methods do not easily provide the highly structured knowledge required in traditional operational space control laws, e.g., Jacobians, inertia matrices, and Coriolis/centripetal and gravity forces, since all these terms are not always instantly observable. They are therefore not suitable for formulating supervised learning as traditionally used in learning control approaches.

We have designed novel approaches to learning operational space control that avoid extracting such structured knowledge as much as ill-posed problems and rather aim at learning the operational space control law directly, i.e., we pose OSC as a direct inverse model learning problem where we acquire an execution policy of the type $\mathbf{u} = \boldsymbol{\eta}(\dot{\mathbf{x}}^d, \mathbf{x}^d, \mathbf{x}, \mathbf{u}_0)$ in which $\mathbf{x}^d = [\dot{\mathbf{p}}^d, \mathbf{p}^d]$ and $\dot{\mathbf{x}}^d$ denote the desired behavior prescribed by the motor primitives in task space while the state $\mathbf{x} = [\dot{\mathbf{p}}, \mathbf{p}, \dot{\mathbf{q}}, \mathbf{q}]$ of the robot is still described by both state-space and task-space components as well as a null-space behavior $\mathbf{u}_0$. Similarly, if we wanted to directly learn the operational space control law as done for model learning in Section 3.1, we would have an ill-posed regression problem as averaging over a non-convex data set is not directly possible. However, the first important insight for this paper is that a physically correct solution to the inverse problem with redundant degrees-of-freedom does exist when learning of the inverse map is performed in a suitable piecewise linear way [17, 17]. The second crucial component for our work is based on the insight that many operational space controllers can be understood in terms of a constrained optimal control problem [16]. The cost function associated with this optimal control problem allows us to formulate a learning algorithm that automatically synthesizes a globally consistent desired resolution of redundancy while learning the operational space controller. From the machine learning point of view, this

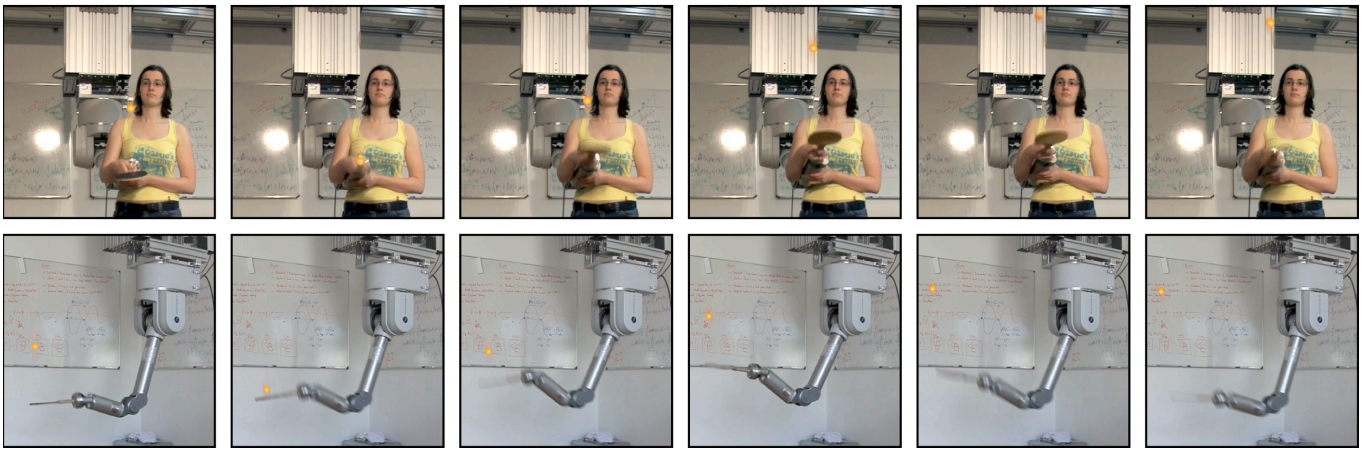


Figure 3. This figure shows how a ball-on-a-string task can be learned by imitation. The human demonstration presents a rhythmic movement with an initial discrete transient where the generic movement is represented by a rhythmic motor primitive modulated by a discrete motor primitive handling the start-up.

learning problem corresponds to a reinforcement learning problem that maximizes an immediate reward. We employ an expectation-maximization policy search algorithm in order to solve this problem. Evaluations on a simulated three degrees of freedom robot arm show that the approach always converges to the globally optimal solution if provided with sufficient data [17].

The application to a physically realistic simulator of the anthropomorphic SARCOS Master arm demonstrates feasibility for complex high degree-of-freedom robots. We also show that the proposed method works in the setting of learning resolved motion rate control on a Mitsubishi PA-10 medical robotics arm [17] and a high-speed Barrett WAM robot arm.

The presented approach also allows us to learn hierachies of operational space controllers where a higher level operational space control law i given by $\mathbf{u}_i = \boldsymbol{\eta}(\dot{\mathbf{x}}_i^d, \mathbf{x}_i^d, \mathbf{x}, \mathbf{u}_{i-1})$ is simply fed the output of the next lower-level operational space control law $\mathbf{u}_{i-1}$ as input. This kind of daisy-chaining of learned control laws may in the future allow us to properly solve the problem of superimposing motor primitives.

4 Learning Motor Primitives

Humans and many mammals appear to rely on motor primitives [3] in order to generate their highly agile movements. In many cases, e.g., when learning to play tennis, humans acquire elementary actions from a teacher. This instructor takes the student by the hand and shows him how to perform forehand and backhand swings. Subsequently, the student tries to play by himself and improves as he observes the results of his own successes and failures.

4.1 Imitation with Motor Primitives

When viewed from a probabilistic perspective, imitation learning can be seen as a relatively straightforward problem. When we have observed trajectories $\boldsymbol{\tau} = [\dot{\mathbf{x}}, \mathbf{x}]$ as well as their distribution $p(\boldsymbol{\tau})$, we will try to reproduce these movements by matching this distribution with a distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ that is determined by the policy parameters $\boldsymbol{\theta}$. While such a policy can be either deterministic or stochastic, it is often easier to model it as a stochastic policy to take the variation in the data into account.

This policy is represented by a motor primitive modeled by a dynamical system as described by Equation (1). Here, imitation learning reduces to inferring the set of parameters so that the distance $D(p(\boldsymbol{\tau})||p_{\boldsymbol{\theta}}(\boldsymbol{\tau}))$ between the observed distribution $p(\boldsymbol{\tau})$ and the reproduced behavior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ is minimized. The Kullback-Leibler divergence is known to be the natural distance measure between probability distributions and is hence employed here.

From this point of view, one can straightforwardly derive regression algorithms such as the ones in [4, 13] to imitate using both the standard formulation of motor primitives [4] as well as the perceptually coupled formulation [13]. As a result, we can learn complicated tasks such as paddling a ball [13] simply by imitation, see Figure 3. This formulation can be made to work both with imitations captured using a VICON setup, see [13], as well as for kinethetic teach-in as in [13].

However, in most real life situations, imitation learning does not suffice and self-improvement is required. E.g., for the Ball-in-a-cup shown in Figure 4, an imitation only suffices for bringing the ball somewhere in the proximity of the cup.

4.2 Self-Improvement by Reinforcement Learning

Reinforcememt learning is in general a much harder problem. Unlike in imitation learning, its focus no longer lies on simply reproducing a presented behavior, but rather on improving a behavior with respect to rewards r. Hence, the system has to try out new actions and, from these actions, infer the policy parameters $\boldsymbol{\theta}^*$ that maximizes the expected return

$$J(\boldsymbol{\theta}) = E\left\{\frac{1}{T}R_{1:T}\right\} = E\left\{\frac{\delta t}{d}\sum\nolimits_{i=1}^{d/\delta t} r_t\right\},$$

where $1/\delta t$ is the sampling rate of the system, d the duration, $T = d/\delta t$ the number of steps and $R_{1:d/\delta t}$ is the return of an episode. In the general setting, reinforcement learning might be an unsolvable problem. Finding a generically optimal policy requires exhaustive try-outs of possible state-action pairs, wherein the number of possibilities grows exponentially with the number of degrees of freedom involved in the task. As anthropomorphic robot exhibit a high dimensionality, they remain beyond the reach of generic reinforcement learning methods.

However, the full reinforcement learning problem appears to be solved rarely in human motor control. For example, olympic high jumper used to refine a variety of different techniques (e.g., straddles, scissor jumps and eastern cut-offs) that all involved running towards the bar and jumping forward. It took until 1968 when the

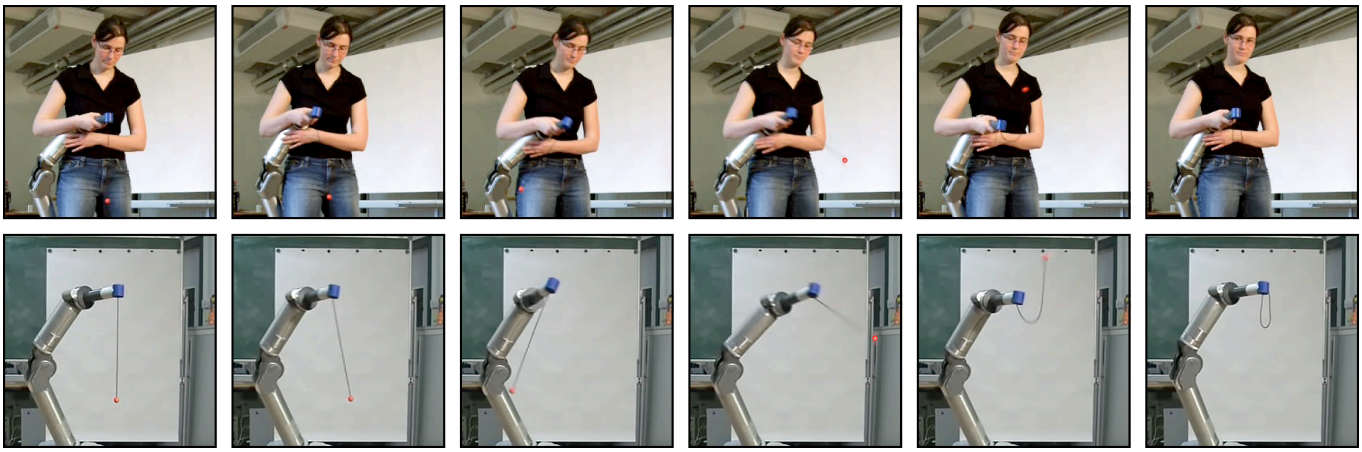


Figure 4. This figure exhibits the general approach, first, a robot is taught the basic movement which is turned into a motor primitive using imitation learning. Subsequently, reinforcement learning is applied to the problem until the robot obtains a motor primitive policy where it slings the ball perfectly into the cup every single time. The imitation is shown in the upper time series while the optimal learned policy is shown in the lower row.

athlete Dick Fosbury accidentally found out that approaching the bar from the side and jumping backwards might be a significantly superior policy. While no reinforcement learning method is in sight that will provide us automatically with such insights, we can design local reinforcement methods that allow us to improve existing policies incrementally. To do so, we rely on obtaining initial parameters $\boldsymbol{\theta}_0$ from an imitation and, subsequently, optimize this policy by self-improvement with respect to the expected return.

Pursuing this type of approach for several years, we have been developing a series of different methods. We originally started out by following the policy gradient approach [10] where the policy improvement is achieved by following the gradient of expected return with respect to its parameters. The resulting update rule can be denoted by

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \alpha_k \left.\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k},$$

where α_k denotes a learning rate at update k and $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is a policy gradient. However, the standard or 'vanilla' policy gradient proved to be suprisingly slow and, thus, not applicable on real robots. It turned out that a covariant or 'natural' policy gradient was able to provide us with the learning speed required for basic motor primitive learning in robotics and we were able to optimize basic movements as well as a T-Ball swing [10]. Nevertheless, the resulting algorithms had open parameter such as the learning rate and the learning process would be too slow for some tasks. As a result, we studied the similarity between expectation-maximization (EM) algorithms and policy gradients. It turned out [9, 17, 17, 11] that as a new cost function we can maximize the distance $D(R(\boldsymbol{\tau})p(\boldsymbol{\tau})||p_{\boldsymbol{\theta}}(\boldsymbol{\tau}))$ between return- or reward-weighted observed path distribution $R(\boldsymbol{\tau})p(\boldsymbol{\tau})$ and the new path distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$. This cost function can become part of a lower bound on the expected return $J(\boldsymbol{\theta})$ and, hence, maximizing it iteratively as in

$$\boldsymbol{\theta}_k = \mathrm{argmax}_{\boldsymbol{\theta}} D(R(\boldsymbol{\tau})p_{\boldsymbol{\theta}_k}(\boldsymbol{\tau})||p_{\boldsymbol{\theta}}(\boldsymbol{\tau}))$$

will at least converge to a locally optimal policy. Such algorithms allow us to show that the problem of policy search can been framed in the parameter estimation setting and, as the similarity to the equations in Section 4.1 makes clear, we have obtained a reward-weighed imitation. At this point, one needs to think about exploration and the type of exploration determines the type of parameter estimation that can be used. For instance, Gaussian exploration with constant variance will result in the reward-weighted regression algorithm [17, 17] and heteroscedastic Gaussian exploration will result in the PoWER algorithm [11].

The PoWER algorithm has been used successfully in a variety of settings, most prominently, it has been able to learn ball-in-a-cup. Here, it started to learn with a policy obtained by imitation that could barely bring the ball into the proximity of the cup. Subsequently, it has learned how to catch the ball in the cup and after less than a hundred trials, it manages to succeed at every trial.

5 Learning to Supervise

In order to get a step closer to creating complex tasks that require a supervisor, various other topics need to be addressed as already outline in Section 2. We will first discuss two topics where we have made recent progress, i.e., goal learning in Section 5.1, and the mixture of motor primitives in Section 5.2. Further topics for learning the supervisory layer are sequencing, blending and superposition of primitives as well as the parsing of longer trajectories into motor primitive automata (see [7]) or determining how many distinct movement primitives are included in a data set ([8]).

5.1 Adjust Motor Primitives to Goals

Previous work in learning for motor primitives has largely focussed on learning the shape parameters $\boldsymbol{\theta}_i$ (see Section 4) while duration d, goal $\mathbf{g}$, amplitude $\mathbf{A}$, etc., were simply considered constant parameters optimized along with the shape [10] or set based on an external stimuli. Here, we attempt to learn mappings from the state to these parameters which allow us to take movements of the same shape and use them for various different contexts. Nevertheless, in goal learning, we assume that we have to respond to constantly changing external stimuli, and always adapt the external parameters appropriately. For example, assume that you are playing a dart game where you are told to hit predetermined fields on the dart board in a certain sequence as well as in robot table tennis (as in Figure 5). In this case, all movements will simply be slight variations of that same throwing movement and can be represented by the same movement primitive. Hence, the proper way to adapt motor primitive to the square that you intend to hit is by altering its duration d and goal $\mathbf{g}$. However, in order to learn this dart game faster than can be achieved using the shape parameters, we also need another method. We discovered

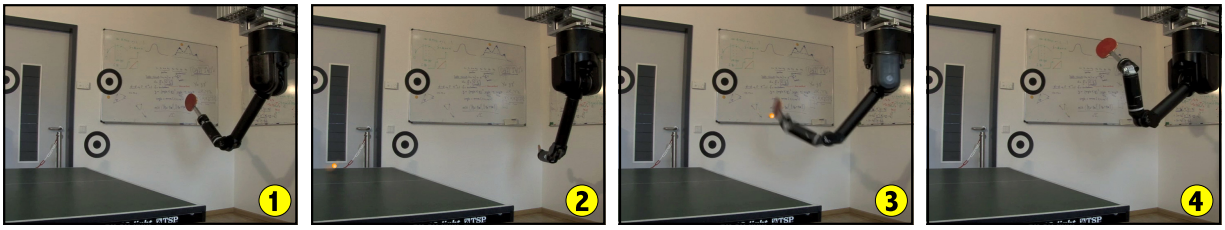


Figure 5. The mixture of motor primitives is used for the selection and generalization of motor primitives in a table tennis setup.

that this can be achieved using a cost regularized Gaussian process regression. The details are described in [12].

5.2 Select & Generalize Motor Primitives

Selection of motor primitives as well as generalization between motor primitives can be achieved using a mixture of motor primitives approach. In such an approach, we have a gating or localization network λ, similar to that in a mixture of experts [14] as part of the supervisor system and activates the right motor primitives. As a result, we obtain a task policy $\mathbf{u} = \pi(\mathbf{x}, t)$ that is composed of the n primitives such that

$$\mathbf{u} = \pi(\mathbf{x}, t) = \frac{\sum_{i=1}^{n} \lambda_i(\mathbf{x}_0)\pi_i(\mathbf{x}, t)}{\sum_{j=1}^{n} \lambda_j(\mathbf{x}_0)}, \tag{2}$$

where $\lambda_i(\mathbf{x}_0)$ denotes the activation of the motor primitive i represented by π_i and $\mathbf{x}_0$ denotes the initial state based upon which of the primitives are activated. A project currently in progress is the learning of table tennis [15] using a mixture of motor primitives (see Figure 5). Here, we currently have achieved already a success rate of 78% of the learned table tennis control law in a ball gun setup and we hope to have a significantly improved setup in the near future.

Using the example of table tennis, we can straightforwardly explain how the mixture of motor primitives is able to generalize between motor primitives. Assume that the system has successfully learned n primitives by imitation observed with different external states $\mathbf{x}_0^i$ (such as a ball position and velocity) and a gating network λ has been obtained. In this case, if a ball is observed at a new initial state $\mathbf{x}_0$, the motor primitives, that resulted in a successful responses to the most similar input, will also be activated and the resulting movement will be a convex combination of the previously successful ones. Selection can be understood in a similar fashion, i.e., if there are both forehands and backhands in the data set, these will be responses to drastically different ball trajectories if viewed in the robot coordinates. Hence, the gating network will discriminate between such motor primitives. For a detailed description see [15].

6 Conclusion

In this paper, we have presented recent progress towards a robot skill learning framework based on [17, 5, 6, 7, 8, 11, 12, 13, 15]. An earlier version of the progress up to 2009 appeared as [18]. While an overview paper in its nature, we have given a detailed outline of a general framework for motor skill. In *learning to control*, we have reviewed our work on learning models using local GPs and on learning operational space control. When learning motor primitives, we have discussed both *imitation learning* approaches as well as our progress in *reinforcement learning* for robotics starting from policy gradients and moving towards reward-weighted self-imitation. Progress towards learning the supervisory layer for complex tasks is briefly discussed with a focus on adjusting primitives to goals as well as generalizing and selecting primitives. Successful implementations on real robots underline the applicability of the presented approaches. This paper summarizes our successes between 2008 to 2012.

REFERENCES

[1] Sciavicco, L. and B. Siciliano. Modeling and control of robot manipulators. MacGraw-Hill, Heidelberg, Germany, 2007.

[2] S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparameteric statistics for real-time robot learning, *Applied Intelligence*, pp. 49–60, 2002.

[3] Schaal, S., A. J. Ijspeert, and A.Billard. Computational approaches to motor learning by imitation. In *The Neuroscience of Social Interaction*, C. D. Frith and D. Wolpert, Eds., Oxford, UK: Oxford University Press, 2004, pp. 199–218.

[4] Ijspeert, A. J., J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems*, volume 15, pages 1547–1554, Cambridge, MA, 2003. MIT Press.

[5] Nguyen-Tuong, D., M. Seeger and J. Peters. Model Learning with Local Gaussian Process Regression. *Advanced Robotics, 23(15), pp.2015-2034*, 2009.

[6] Nguyen-Tuong, D. and J. Peters. Semi-parametric regression in learning inverse dynamics. In *International Conference on Robotics & Automation (ICRA)*, 2010.

[7] Chiappa, S and J. Peters. Motion segmentation by detecting in continuous time-series. In *Advances in Neural Information Processing Systems 23 (NIPS'10), Cambridge, MA: MIT Press*, 2010

[8] Chiappa, S., J. Kober and J. Peters. Using Bayesian Dynamical Systems for Motion Template Libraries *Advances in Neural Information Processing Systems 21 (NIPS'08), Cambridge, MA: MIT Press*, 2009.

[9] Dayan, P. and G. E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

[10] Peters, J. and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks, 21(4), pages 682-697 (2008)* .

[11] Kober, J. and J. Peters. Policy Search for Motor Primitives in Robotics. *Machine Learning, 84(1), pp.171203*, 2011.

[12] Kober, J., A. Wilhelm, E. Oztop and J. Peters. Reinforcement Learning to Adjust Parametrized Motor Primitives to New Situations. *Autonomous Robots*, 2012.

[13] Kober, J. and J. Peters. Imitation and Reinforcement Learning Practical Algorithms for Motor Primitive Learning in Robotics. *IEEE Robotics and Automation Magazine, 17(2), pp. 55-62*, 2010.

[14] Jordan, M. and R. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6: 181–214, 1994.

[15] Muelling, K., J.Kober, O.Kroemer, and J.Peters. Learning Table Tennis with a Mixture of Motor Primitives. submitted to *International Journal of Robotics Research*

[16] Peters, J., 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.

[17] Peters, J. and S.Schaal. Learning to Control in Operational Space. *The International Journal of Robotics Research*, 27(2), 197-212, 2008.

[18] Peters, J., J.Kober, K.Muelling, D.Nguyen-Tuong, and O.Kroemer Towards robot skill learning for Robotics. *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2009.

[19] Sentis, L. and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4):505-518, 2005