2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Dynamic Optimality in Real-Time: A Learning Framework for Near-Optimal Robot Motions

Roman Weitschat, Sami Haddadin, Felix Huber and Alin Albu-Schäffer

Abstract— **Elastic robots have a distinct feature that makes them especially interesting to optimal control: their ability to mechanically store and release potential energy. However, solving any kind of optimal control problem for such highly nonlinear dynamics is feasible only numerically, i.e. offline. In turn, optimal solutions would only contribute a clear benefit for dynamic environments/tasks (apart from rather general insights), if they would be accessible/generalizable in real-time. In this paper, we propose a framework for executing near-optimal motions for elastic arms in real-time. We approach the problem as follows. First, we define a set of prototypical optimal control problems. These represent a reasonable set of motions that an intrinsically elastic robot arm is sought to execute. Exemplary, we solve the optimal control problem for some of these prototypes in a roughly covered task space. Then, we encode the resulting optimal trajectories in a dynamical system via Dynamic Movement Primitives (DMPs). Finally, a distance and cost function based metric forms the basis to generalize from the learned parameterizations to a new unsolved optimal control problem in real-time. In short, we intend to overcome the well known problems of optimal control and learning with associated generalization: being offline and being suboptimal, respectively.**

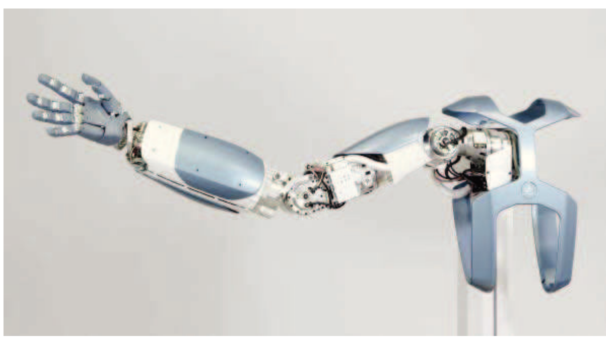

Fig. 1. The DLR Hand-Arm system (HASy) that is fully equipped with Variable Stiffness actuation in each joint [1].

Generating robot trajectories is one of the largest fields of robotics research. Initially, it started from purely geometric (mostly linear interpolation, circular interpolation or polynomial splines) approaches in the early days for calculating *point-to-point* or *N-via-point* motions without taking into consideration the environment or the robot dynamics. Since rigid industrial robots were supposed to execute purely geometric tasks, the low-level controller ensures accurate tracking behavior of the device, while trajectory generation remains on kinematic level. A recent very nice overview on these techniques can be found in [2]. In order to solve not only the free space problem but also the general motion planning problem, a vast range of methods for planning global motions in complex, however mostly static environments were developed. They ensure that the robot can be brought collision free from an initial configuration to a desired final configuration [3], [4], [5], [6], given a solution exists. Typically, such planning methods generate a statically valid path in a timeframe of seconds and are usually suboptimal (in particular in the dynamic sense). However, dynamic optimality e.g. in terms of minimal joint velocity or energy consumption is certainly a desirable property, which can be expressed on kinematic or dynamic level, respectively. In fact, for intrinsically elastic systems it offers entirely new ways to control and will certainly affect even global motion planning at some point. In industrial robotics, time optimal motions are of primary concern. For generating kinematically optimal trajectories in real-time, several efficient algorithms became available [7]. However, as all aforementioned schemes work on kinematic level only (some also up to kinodynamics), they can never fully exploit the inherent dynamic capabilities of a robot. Strong insights in this respect were e.g. obtained in [8], [9] by solving the time optimal tracking problem while taking into account the rigid body dynamics and the constrained motor torques. These results allow to execute dynamically optimal tracking in real-time for non-redundant rigid manipulators. Further work on solving optimal control (OC) problems for manipulators, however purely offline, can e.g. be found in [10], [11]. Discussions and results on full body motions are given in [12], where the problem of optimal running is solved with direct methods. Particularly challenging and interesting becomes the achievement of dynamic optimality if being faced with intrinsically elastic robots such the DLR Hand-Arm System (Hasy) [1], see Fig. 1. This is due to the fact that these systems are able to store and release potential elastic energy, which is especially helpful for carrying out explosive motions such as e.g. throwing. The effect was thoroughly analyzed for 1DoF variable stiffness systems (VSA) in [13], [14], [15], where analytic solutions could still be derived for some cases. For multiple degrees of freedom, however, only numerical solutions can be found due to the strongly nonlinear and coupled behavior [16]. Nonetheless, as elastic systems are designed for acting in dynamic environments at close proximity to humans, it is clear that pure offline solutions are neither applicable nor sufficient.

An entirely different approach to generate task trajectories originates from trajectory learning and generalization, where considerable effort was put into the development of the *learning-by-demonstration* (LbD) paradigm. Typically, a desired motion is taught to an actively or passively backdrivable robot [17], [18] kinesthetically [19], [20]. Alternatively, human motion tracking is used for generating trajectories, which are transferred to the robot and

All authors are with Robotics and Mechatronics Center, DLR - German Aerospace Center, Wessling, Germany `sami.haddadin@dlr.de`

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

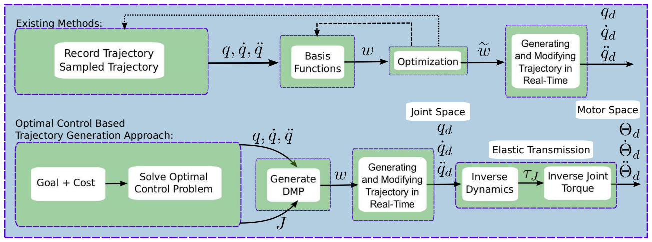

Fig. 2. Comparison of the overall structure of classical learning approached (upper) to our OMF (lower).

can then also be refined via kinesthetic teaching [21] or other iterative learning/optimization techniques. In order to generalize the demonstrated trajectories, one has to find a suitable representation that builds on the acquired data. A particularly interesting one is a dynamical forced system in the form of a Dynamic Movement Primitive (DMP) [22]. As the trajectories are generated by means of a second, or higher order differential equation, it is relatively simple to incorporate disturbance forces for collision avoidance or retraction [23], [24]. LbD is a very flexible framework, which can be applied to various types of problems [25], [26]. Nonetheless, there are also natural limitations with existing schemes, as the generated motions are certainly suboptimal (both kinematically and dynamically).

This brings us to the main idea of our paper. We intend to overcome the limitations of each approach by combining them into a single Optimal Motion Framework (OMF) that utilizes the advantages of both:

- Implicit representation of the robot dynamics in optimal control solutions
- Complexity reduction, generalization, and real-time capability of learning dynamical systems

The paper is organized as follows. Section II introduces our approach. Sec. III describes the concept of prototypical optimal control problems. In the following, Sec. IV provides an overview on how we embed optimal trajectories into DMPs and then generalize with the help of a distance and/or cost function based metric. Section V describes various simulations and experiments that underline the validity and "near-optimality" of our approach. Section VI concludes the paper.

II. APPROACH

As one is generally not interested in the solution of any possible OC problem, we first define a reasonable set of representative optimal control problems that can be roughly divided into *reaching type motions* and *tracking type motions*[1]. The former represent a rather abstract task, which concrete joint space trajectory for optimally solving the task (also in operational space) is to be found. The latter problem is specified by a full (joint or operational space) trajectory that shall be optimally tracked, i.e. we need to find the joint space motion that ensures both, minimal tracking error and possibly other costs, such as energy related ones. However, for both motion types we encode a set of optimal joint space trajectories for each task into a modified version of the original DMP formulation, whereas for the latter case a two-staged DMP approach is utilized, one for generalizing reference motions (in joint or task space) and one for generalizing the optimal solutions for tracking this operational space movement. Please note that for sake of clarity, we focus on *reaching type motions* in this paper. We show that the embedding of optimal motions that inherently capture the robot dynamics leads to the ability to generate near-optimal motions in real-time for each of the considered problems. Figure 2 depicts the overall framework target structure in comparison to the LbD approach. Specifically, we

1) use **dynamically** optimal trajectories coming from solving complex nonlinear optimal control problems as learning input,
2) encode them into an optimized dynamical system,
3) and use a metric, based on the cost function and/or geometric distance, for selecting a near-optimal parameter set in real-time for the generalizing step.

A somewhat related approach to ours can be found in [27], where kinematically optimal trajectories were used as template trajectories that are subject to situation related costs for collision avoidance. Together with a situation descriptor these are subsequently generalized via another offline stage, which, however, does not optimize from scratch and is therefore considerably faster than the generation of the original training data. First work on using energy optimal solutions for catching tasks and applying different learning approaches for generalization can be found in [28]. There, optimal trajectories are encoded in B-splines or trapezoidal functions, which are then generalized with different state-of-the-art machine learning techniques as e.g. Nearest-Neighbor (NN) or Support Vector Machines (SVM).

Our approach discriminates from these existing techniques in the sense that we also consider elastic systems (several additional constraints and nonlinearities have to be considered). Furthermore, we combine optimal control (which offers, in contrast to pure nonlinear optimization, the tools to give stronger hints for global optimality of a solution) with generalization algorithms by means of dynamical systems (DMPs), and also use the cost function of the according optimal control problem as an underlying generalization metric. Furthermore, instead of treating only a single motion control problem, we make an attempt to give a rather general methodology for a set of control problems that cover most typical robot motion tasks, which, however, can be easily extended if needed.

In the following section, we shortly introduce some basics on optimal control for VSA[2] and then describe our selection of prototypical optimal control problems.

[1]Please note that we do not claim completeness. It is straight forward to extend the approach to new problems as well. In this paper we focus on the methodology.

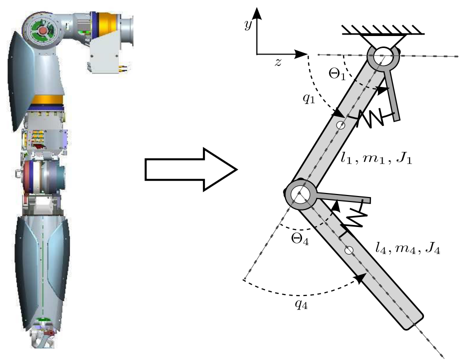

Fig. 3. 2-DoF intrinsically compliant robot model (right). The used parameters are taken from the DLR Hand-Arm system (left). Link 1: mass m_1 = 4.6 kg, inertia $J_1 = 0.0453$ kgm^2, length l_1 = 0.34 m. Link 4: mass m_4 = 5 kg, inertia $J_4 = 0.0492$ kgm^2, length l_4 = 0.34 m.

III. OPTIMAL CONTROL

A. Optimal control for elastic joints

Let us consider the class of systems that is described by a set of first order differential equations $\dot{\boldsymbol{\vartheta}}(t) = f(\boldsymbol{\vartheta}(t), \mathbf{u}(t))$, where $\boldsymbol{\vartheta}$ denotes the state vector and $\mathbf{u}$ the control input, respectively. The initial state is denoted by $\boldsymbol{\vartheta}(t_0) = \boldsymbol{\vartheta}_0$, the final constraints are $\boldsymbol{\phi}(\boldsymbol{\vartheta}(t_f), t_f) = \mathbf{0}$, and the set of path constraints is $\mathbf{c}(\boldsymbol{\vartheta}(t), \mathbf{u}(t), t) \leq \mathbf{0}$. Solving an optimal control problems aims at finding the control input $\mathbf{u}^*$ that minimizes a given cost function

$$\min_{\mathbf{u}(t)} \mathcal{J} = h(\boldsymbol{\vartheta}(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{\vartheta}(t), \mathbf{u}(\mathbf{t}), t)dt, \quad (1)$$

with $h(\boldsymbol{\vartheta}(t_f), t_f)$ being the terminal and $\int_{t_0}^{t_f} g(\boldsymbol{\vartheta}(t), \mathbf{u}(\mathbf{t}), t)dt$ the running cost. This cost function basically denotes the task to be accomplished and may take various forms. The dynamics of a full VSA arm can be described by

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_J \quad (2)$$

$$B\ddot{\boldsymbol{\Theta}} + \boldsymbol{\tau}_J = \boldsymbol{\tau}_m \quad (3)$$

$$\boldsymbol{\tau}_J = \boldsymbol{\tau}_J(\boldsymbol{\Theta}, \mathbf{q}, \boldsymbol{\sigma}), \quad (4)$$

where $\boldsymbol{\Theta}$ and $\mathbf{q}$ are the motor and link side position, respectively. The position of the stiffness actuator is denoted $\boldsymbol{\sigma}$, which is treated as a constant parameter in this paper. $M(\mathbf{q})$, $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$, $\mathbf{g}(\mathbf{q})$, and B are the link side inertia matrix, the centrifugal and Coriolis vector, the gravity torque, and the motor inertia. $\boldsymbol{\tau}_m$ denotes the torque acting through the positioning motor[3]. The vector $\boldsymbol{\tau}_J$ represents the elastic joint torque, which is a nonlinear function of the elastic deflection $\boldsymbol{\varphi} = \boldsymbol{\Theta} - \mathbf{q}$ and $\boldsymbol{\sigma}$. Please note that we assume that for a given $\boldsymbol{\sigma}$ it is possible to solve (4) for $\boldsymbol{\Theta}$. The optimal trajectories in this paper are obtained with the nonlinear optimal control solver GPOPS [29] available in MATLAB. It uses the hp-adaptive Legendre-Gauss-Radau quadrature integral pseudospectral method for solving general nonlinear optimal control problems.

The considered concrete nonlinear dynamics we intend to solve is a 2 DoF dynamic HASy model, see Figure 3. It serves as a benchmark problem for an intrinsically elastic robot with nonlinear torque deflection characteristics. Please note that in this work we do not make use of the possibility to change this relation via adjusting $\boldsymbol{\sigma}$, i.e. we assume the robot to be intrinsically elastic but the stiffness adjuster remains constant. Subsequently, we discuss several problems, which we consider to be important for elastic systems.

B. Prototypical optimal control problems

By nature, the motion generation problem is infinitely large and generally poorly defined in the sense of what a desired motion should exactly look like. Therefore, it is rather hard to find a general optimal control problem that inherently contains all possible instantiations and thus captures the essence of motion. Therefore, we pragmatically resolve this dilemma by introducing *prototypical optimal control problems*. These are sought to find an optimal control input $\mathbf{u}^*$ that generates a distinct type of motion for a given robotic system and potentially a secondary system the robot is associated to (e.g. an object to be manipulated with another target dynamics). The following classification aims at grouping motion behaviors according to their "higher-level" target. We coarsely distinguish between following main problems:

1) reaching type motion
2) tracking type motion

In the following classification of prototypical OC problems (see Fig. 4), $\mathcal{N}(\boldsymbol{\vartheta})$ denotes the neighborhood of the system state $\boldsymbol{\vartheta}$ according to a suitable metric (typically spatially). $\boldsymbol{\vartheta}_0$, $\dot{\boldsymbol{\vartheta}}_0$, $\boldsymbol{\vartheta}_f$, $\dot{\boldsymbol{\vartheta}}_f$ denote initial state and velocity, and final state and velocity.

- Reaching type motions
  1) Reaching motion (grasping): $\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0 = \mathbf{0} \rightarrow \boldsymbol{\vartheta}_f, \dot{\boldsymbol{\vartheta}}_f = \mathbf{0}$, considering minimum final error and minimum energy
  2) Explosive motion (throwing motion, catching) $\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0 = \mathbf{0} \rightarrow \mathcal{N}(\boldsymbol{\vartheta}_f), ||\dot{\boldsymbol{\vartheta}}_{\max}||\mathbf{e}$, considering minimum final error (and minimum energy)
  3) Explosive target motion (throwing motion, boxing, hitting) (dynamic grasping) $\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0 = \mathbf{0} \rightarrow \mathcal{N}(\boldsymbol{\vartheta}_f), \dot{\boldsymbol{\vartheta}}_f$, considering minimum final error
  4) Implicit target motion (object manipulation, ballistic throw e.g. bucket throw) $\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0 = \mathbf{0} \rightarrow \mathcal{N}(\boldsymbol{\vartheta}_f)$ while, considering minimum final error and minimum energy
  5) Implosive motion (braking motion, stopping catching) $\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0 \rightarrow \mathcal{N}(\boldsymbol{\vartheta}_f), \dot{\boldsymbol{\vartheta}}_f = \mathbf{0}$, considering minimum final error and minimum energy

  Please note that we intentionally excluded time optimal problems, as we leave them for future work.
- Tracking type motions (not necessarily optimal control problems[4])

[2]Please note that we introduce the problem formulation for VSA in general and not for the elastic case only. The subsquent learning and generalization, however, is then done for the case without explicit stiffness control input.

[3]We neglect the dynamics of the usually significantly smaller and therefore faster stiffness adjustment motor.

[4]Depending on the available degrees of freedom, there might be only a single solution to the problem and no solution space to be explored.

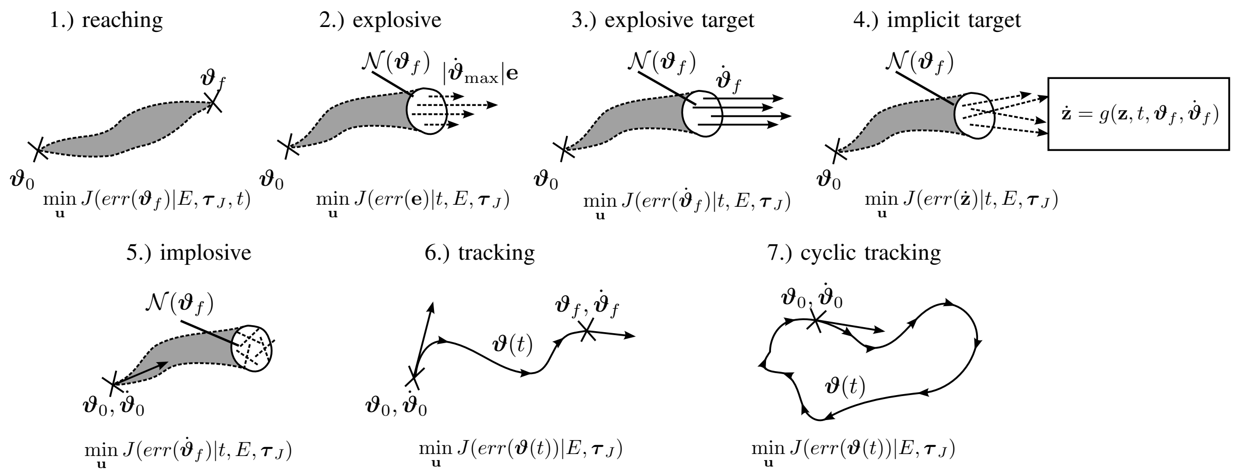

Fig. 4. Grouping of motion behaviors into *reaching type motions* (1-5) and *tracking type motions* (6-7).

1) Tracking (gestures, constrained motion primitives): $(\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0) \rightarrow \boldsymbol{\vartheta}(t) \rightarrow (\boldsymbol{\vartheta}_f, \dot{\boldsymbol{\vartheta}}_f)$,
2) Cyclic tracking (stirring, cranking, shaking): $(\boldsymbol{\vartheta}_0, \dot{\boldsymbol{\vartheta}}_0) \rightarrow \boldsymbol{\vartheta}(t)$ with $\boldsymbol{\vartheta}(t) = \boldsymbol{\vartheta}(t - pT)$, $T =$ const., and $p \in \mathbb{N}^+$,

Next, we discuss the embedding of the optimal trajectories into the DMP formulation and how the cost function might be exploited for generalization.

IV. Optimal Motion Framework

input : motion type, $\mathbf{c}(\mathbf{q})$, parameters $\{\boldsymbol{\xi}_k\}$
output: $\mathbf{w}^*, \boldsymbol{\Phi}^*$

for $k \leftarrow 1$ **to** m **do**
  $[\mathbf{q}_k^*, \dot{\mathbf{q}}_k^*, \ddot{\mathbf{q}}_k^*] = \min_{\mathbf{u}} J(\text{motion type}, \mathbf{c}(\mathbf{q}), \boldsymbol{\xi}_k)$;
  for $i \leftarrow 1$ **to** n **do**
    $\mathbf{f}_i^*(t_i) = -\tau^2 \ddot{\mathbf{q}}^*(t_i) + \kappa(t_i)(\mathbf{q}^*(\tau) - \mathbf{q}^*(t_i)) - D\tau\dot{\mathbf{q}}^*(t_i)$;
    $x_i = \exp\left\{-\frac{\alpha_x}{\tau} t_i\right\}$;
    $\mathbf{x}_i = [x_i; \cdots; x_i]_{dim=M\times 1}$;
    $\mathbf{F}_k^* = [\mathbf{F}_k^*;\ \mathbf{f}_i^{*T}(t_i)]$;
    $\mathbf{X} = [\mathbf{X};\ \mathbf{x}_i^T]$;
  end
  $[\mathbf{w}_k^*, \boldsymbol{\Phi}^*] = \min \Gamma[(\boldsymbol{\Phi}^j, \mathbf{F}_k^*, \mathbf{X}) \rightarrow \mathbf{w}^j \rightarrow \mathbf{f}_\approx^j]$
end

Algorithm 1: DMP generation

Algorithm 1 depicts the overall steps of our OMF. First, we solve a prototypical OC problem for a certain grid on the task space (e.g. goal configurations for reaching motions). The motion type, the constraints $\mathbf{c}(\mathbf{q})$, and an ensemble of task relevant parameters $\{\boldsymbol{\xi}_k\}$ (such as m goal locations) define the optimal control problem to be solved, see Sec. III-A. Its solution in turn initializes Algorithm 1. For generalizing a specified task, we need m optimal results of length n (number of trajectory samples) that represent a roughly covered grid over the task space. The grid of this task space of optimized and learned trajectories is experimentally determined for obtaining reasonable results. Solving the optimal control problem yields trajectories defined by optimal link side position $\mathbf{q}_k^*(t) \in \mathbb{R}^M$, velocity $\dot{\mathbf{q}}_k^*(t) \in \mathbb{R}^M$, and acceleration $\ddot{\mathbf{q}}_k^*(t) \in \mathbb{R}^M$ in joint or task space of dimension M, described in the outer loop of Algorithm 1. Deploying these trajectories into a second order differential equation (inner loop) with a total duration of the movement τ, a stiffness factor κ and a damping term D, we obtain a force based trajectory $\mathbf{f}_i^*(t_i) \in \mathbb{R}^M$. The dynamical force is then expressed as a function of the canonical system x, which can be interpreted as a path parameter. x is designed such that it continuously decreases from 1 to 0, being only parameterized by its slope parameter α_x. The according force function is then approximated by a Gaussian basis:

$$\mathbf{f}^*(t) \approx \mathbf{f}_\approx(x) = \left[\frac{\sum_{i=1}^{N} w_{i,l}^* \psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} x\right]_{dim=M\times 1} \tag{5}$$

It is parametrized by a set of optimal weights $\mathbf{w}_l^* \in \mathbb{R}^N$ for each dimension $l \in M$ and an optimal parameter vector for the Gaussian kernels. The Gaussian basis is defined as $\psi_i(x) = e^{-h_i(x-c_i)^2}$ with the center points and widths being $c_i = e^{-\alpha_x \frac{i-1}{N-i}}$ and $h_i = \frac{1}{(c_{i+1}-c_i)^2}$, respectively. The resulting linear regression problem $\mathbf{X}\mathbf{w}_l^* - \mathbf{f}_l^* = \mathbf{0}$ (with $\mathbf{f}_l^* := \mathbf{F}^*(1:n, l)$), which needs to be set up for every l and k (k is omitted for brevity), can be solved with efficient existing approaches [30]. Despite this being already a reasonably well working approach, this system choice leaves some issues unresolved. In particular, if choosing κ to be constant, as is typically done, the initial force $\kappa(\mathbf{q}_0 - \mathbf{g})$ produces a very large acceleration force $\mathbf{f}^*(t=0) \neq 0$, which is clearly not capturing the essence of a point-to-point trajectory that is initially at rest. This can be solved by introducing a time-varying continuously increasing, however, bounded stiffness $\kappa(t)$. A possible choice that preserves stability and convergence is

$$\kappa(t) = g_\kappa \left(\frac{1}{1 + e^{-g_\kappa k_\kappa t\left(\frac{g_\kappa}{k_0} - 1\right)}}\right), \tag{6}$$

where g_κ is the final value of the function, k_0 is the initial value ($k_0 > 0$), and k_κ is the slope. Of course, the particular implementation may be chosen differently.

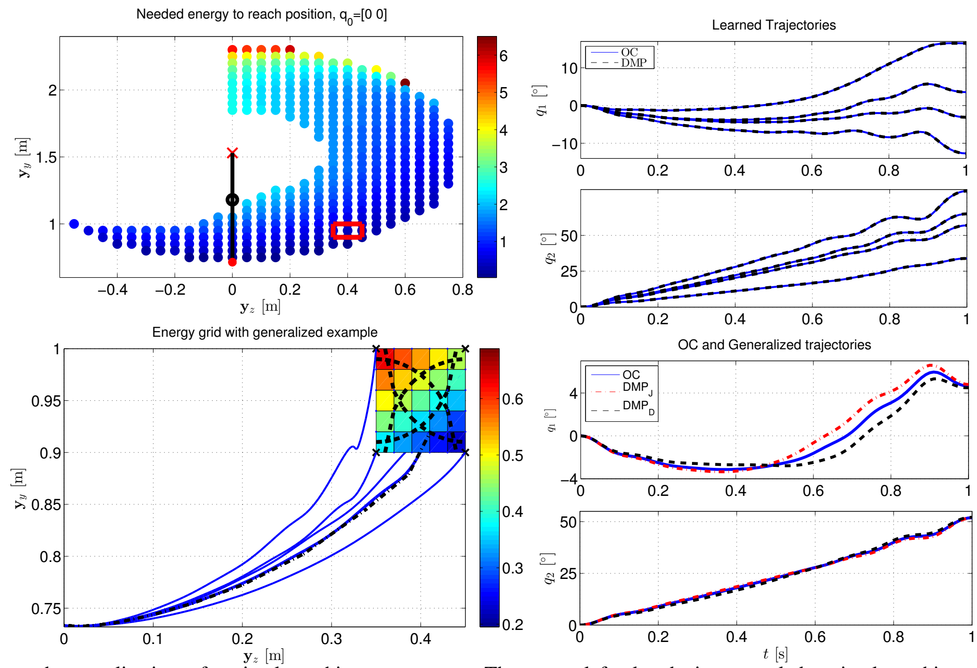

Fig. 5. Learning and generalization of optimal reaching movements. The upper left plot depicts sampled optimal reaching movements for the robot workspace. The 2DoF robot in stretched out configuration is indicated in black. The color value indicates the value of J for every trajectory, i.e. the consumed energy for the motion. The lower left plot shows a close up around $(\mathbf{y}_z, \mathbf{y}_y) = (0.4, 0.95)$. The optimal trajectories that are used for the learning step are indicated by the crosses in the corner. The according generalization DMP motion towards the new goal within the rectangle is compared against the according optimal motion, which was **not** used for learning. It was generalized inside this section by weighting kernel parameters with inverse distance weighting (9). The two upper right plots show optimal and DMP trajectories. The randomly generalized trajectory in the section are compared with the optimal trajectory (checked afterwards). Furthermore, the lower two plots show the difference between purely geometric and cost based weighting.

A further issue when using an original DMP design is the dependency of number of weights and accuracy-reproduction of the original input data. To minimize computation time and disk space, we want to use only few Gaussian basis functions. As described in [30], this can be solved by parameterizing the Gaussian widths as $h_i = \frac{\beta_x}{c_{i+1}-c_i} + \gamma_x$, and finding the optimal parameter vector $\mathbf{\Phi} = [\alpha_x \ \beta_x \ \gamma_x]$ by suitable minimization. For this, the error cost metric $\Gamma = \frac{1}{T}\sum_{i=1}^{T} ||\mathbf{f}_i - \mathbf{f}_{\approx,i}||$ is to be minimized (an alternative choice would be $\Gamma = \frac{1}{T}\sum_{i=1}^{T} ||\mathbf{y}_i - \mathbf{y}_{\approx,i}||$), see the last step of Algorithm 1[5]. This problem can be solved with standard SQP solvers. This optimization is carried out for increasing N and finally, the minimal N that produces a negligible force error only is selected. After having obtained the optimal parameter set $[\mathbf{w}^*, \mathbf{\Phi}^*]$, one may then generalize the motion along $t \in [0 \cdots t_f]$ for different goals $\mathbf{g}$ by solving

$$\mathbf{q}(t) = \frac{1}{\tau^2} \iint_0^{t_f} \mathbf{f}_{\approx}(\mathbf{x}) + \kappa(t)(\mathbf{g} - \mathbf{q}) - D\tau\dot{\mathbf{q}} \; \mathrm{d}t \; \mathrm{d}t + \mathbf{q}(0). \tag{7}$$

From (7) we obtain the link position, velocity, and acceleration which can then be inserted into (2). The motor trajectories $\Theta(t)$ are obtained by solving (2) for $\boldsymbol{\tau}_J$, and then use (4).

(7) can be implemented in real-time, leading to a scheme that is capable of producing near-optimal solutions for different start and end configurations based on a finite set of optimal trajectories. We decided to use DMPs for generating optimal real-time trajectories because we want to use force generated trajectories to have the ability to affect the desired trajectory online, for example collision avoidance or constraint adherence (this is, however, not part of this paper). Such requirements cannot be easily complied with a two staged optimization approach, where the optimal control problem is solved offline first and generalization is done via linear optimal control approaches for sufficient closeness to the offline solutions. Important to notice is that the tracking type motions (motion type 6.+7. from Fig. 4) can also be covered with the DMP approach (see [22] for the original formulation). Our extensions for matching the special needs of elastic robots, however, are out of the scope of the paper and therefore omitted for brevity.

Next we discuss our approach to generalization.

A. Generalization for DMP motions

Generalization of DMPs has e.g. been addressed in [31]. In order to make use of the fact that "close" trajectories presumably encode more relevant information for a new goal, distance based weighting is applied for the extrapolation step. For the reaching movement generalization, we may apply the method from [32]

$$\mathbf{w}_l^*(\mathbf{y}_g) = \frac{\sum\limits_{\forall k:\sigma_k \le \delta} \mathbf{w}_l^*(\mathbf{y}_k)\sigma_k^{-1}}{\sum\limits_{\forall k:\sigma_k \le \delta} \sigma_k^{-1}}, \tag{8}$$

Equation (8) is a sum of kernel weights multiplied with the inverse of the geometric or energetic distances between the

[5]For brevity, the index j indicates the required minimization loop.

previously learned and the new goal. σ_k is defined as

$$\sigma_k = \begin{cases} ||\mathbf{y}_g - \mathbf{y}_k|| + \epsilon, & ||\mathbf{y}_g - \mathbf{y}_k|| + \epsilon < \delta \\ \epsilon & ||\mathbf{y}_g - \mathbf{y}_k|| + \epsilon \geq \delta, \end{cases} \tag{9}$$

where $\mathbf{y}_g$ denotes the new goal position or goal cost function value, respectively. $\mathbf{y}_k$ represents the surrounding sample goal k, δ is a distance limitation and ϵ prevents division by zero. δ is a chosen metric for selecting close Gaussian bases of interest and is large enough that at least one Gaussian basis is used. The introduced metric in (9) is discussed in Sec. V in more detail.

After having introduced all necessary tools to solve some exemplary task, we now present simulations and experiments obtained with the OMF.

V. Simulation and Experiments

First, the generalization of energy-like optimal point-to-point motions with the HASy is shown and different generalization approached compared to each other. Then, optimal throwing movements are investigated and experimentally verified on the HASy robot.

A. Learning and generalization of optimal reaching movements

The mostly used movement in robotics is a point-to-point task. Typically, such reaching motions are generated by common interpolation schemes. The results are, however suboptimal (e.g. in the energetic sense). With the OMF, our solution to the problem is as follows. First, a grid of optimal solutions with minimal required motor energy is calculated over the entire workspace, see Fig. 5. The cost function to solve minimum input energy point-to-point motions is defined as

$$\min_{\dot{\mathbf{\Theta}}(t)} J = \int_{t_0}^{t_f} \left(\frac{1}{2} w_{\Theta_1} \dot{\Theta}_1^2 + \frac{1}{2} w_{\Theta_4} \dot{\Theta}_4^2 \right) \, dt, \tag{10}$$

with $w_{\Theta_1} = 1$, $w_{\Theta_2} = 1$ and $\dot{\Theta}_i$ being the motor velocity of joint $i = [1, 4]$. Ideally, one would choose $\boldsymbol{\tau}_m^T \dot{\mathbf{\Theta}}$ to be the running cost. However, due to safety reasons the input to the system is practically $\dot{\Theta}$, i.e. τ_m is not controlled directly. The virtual input is therefore chosen to be $\dot{\Theta}_d$. The equality constraint $\mathbf{q}(t_f) - \mathbf{q}_d = 0$ is considered to guarantee reaching $\mathbf{q}_d(t_f)$. The arising information regarding the cost function is stored into a database.

Using inverse kinematics, $\mathbf{q}_d(t_f)$ and $\mathbf{q}(t_f)$ are obtained. A few of the obtained trajectories from (10) are then encoded into DMPs, see Sec. IV. To represent $m = 4$ optimal input trajectories, $n = 40$ Gaussian bases are used. Further parameters are $g_k = 80$, $k_k = 20$, $k_0 = 2 * 10^{-4}$, and $\alpha_x = 2.8$.

Figure 5 depicts the according results over the entire 2-DoF workspace (upper left plot). A more detailed resolution of the red square is depicted in the bottom left plot. The trajectories of moving to the 4 "corners" of the square are optimal solutions from (10). The accuracy of DMP trajectories is shown in the upper right plot as a comparison between OC and real-time generated DMP trajectories. Finally, the bottom right plot depicts a comparison between DMP generalized trajectories and the optimal ones.

	1-NN J	1-NN Y	2-NN J	2-NN Y
J_{err} %	10.33	12.48	5.03	4.68
J_{max} rad^2s^{-2}	0.16	0.22	0.10	0.07
	3-NN J	3-NN Y	δ=0.21 J	$\delta = 0.09$ Y
J_{err} %	4.61	3.57	5.03	**2.66**
J_{max} rad^2s^{-2}	0.08	0.03	0.05	**0.02**

TABLE I

Comparison of OC to optimal DMP generalized with Nearest Neighbors (NN) cost J and geometric distance Y based

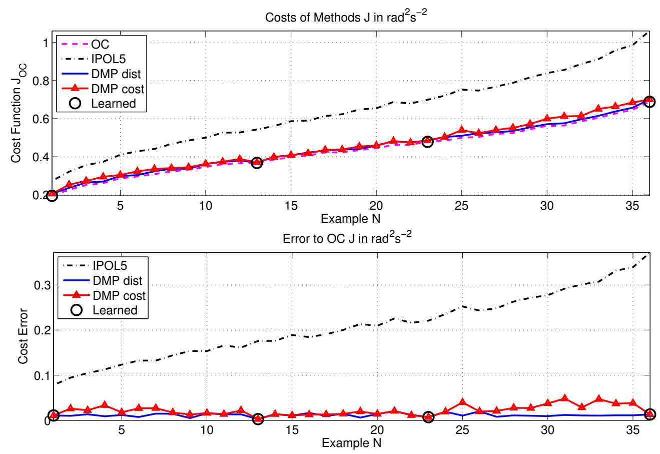

Fig. 6. Comparison of cost functions calculated for OC, DMP and IPOL.

To validate the performance of our approach, we give some comparison to other solutions. A straight forward one is the application of geometry based Nearest Neighbor (NN). On the other hand, it is possible to interpolate weights as explained in Sec. IV-A. The according numerical comparisons to the offline generated ground truth OC trajectories are listed in Tab. I[6]. As a meaningful error metric the relative error with respect to the ground truth cost is considered:

$$J_{err} = \frac{\sum_{i=1}^{N} J_{DMP_i} - J_{OC_i}}{\sum_{i=1}^{N} J_{OC_i}} \tag{11}$$

N is the number of optimized and simulated samples. J_{max} is the maximum error of $J_{DMP_i} - J_{OC_i}$. The grid in the experiment was chosen as point-to-point motion from $y = 0.9 : 1.0$ and $z = 0.35 : 0.45$ with a step size of 0.02 m along each direction, see Fig. 5 (bottom left). J_{OC} is the result of solving the optimal control from (10) and J_{DMP} is obtained with the OMF and calculating the cost with $J = \sum_{i=1}^{j} \int_{t_0}^{t_f} \dot{\Theta}_i^2$. In this case, the best generalization results are obtained with the distance based metric with $\delta = 0.09$ [7]. To compare with standard trajectory generation, a 5th order polynomial trajectory was used to reach the goal configuration as well [33]. The according Ipol5 routine is given by

$$\mathbf{q} = \mathbf{a}_0 + \mathbf{a}_1 t^3 + \mathbf{a}_2 t^4 + \mathbf{a}_3 t^5, \tag{12}$$

[6] 2-NN Y denotes the interpolation between the first and second NN geometric distance based.

[7] This value was found by minimization of the relative error using standard SQP.

with $t_f = 1s$, $\mathbf{a}_0 = \mathbf{q}_0$, $\mathbf{a}_1 = \frac{20\mathbf{q}_f - 20\mathbf{q}_0}{2t_f^3}$, $\mathbf{a}_2 = \frac{30\mathbf{q}_0 - 30\mathbf{q}_f}{2t_f^4}$, $\mathbf{a}_3 = \frac{12\mathbf{q}_f - 120\mathbf{q}_0}{2t_f^5}$. $\mathbf{q}_0$ denotes the start and $\mathbf{q}_f$ the final configuration. Figure 6 depicts the overall comparison between OC generated trajectories, DMPs with best distance and cost based generalization, and polynomial interpolation (Average J_{err} = 48.03 %). To sum up, the distance based generalization yields the best results for the examined task.

Next, we discuss how the OMF is able to generate optimal explosive motions for elastic robots in real-time.

B. Generalized throwing with the DLR Hand-Arm system

The throwing task is an illustrative example for explosive and implosive movements, see Fig. 4. The procedure, for executing the throwing task is again to encode optimal control trajectories as learning references for DMPs. The cost function for the throwing task (more specifically, the task is to throw a ball into a bin) is chosen to be

$$\min_{\dot{\Theta}(t)} J = \tfrac{1}{2} w_1 e_{dis}^2 + \int_{t_0}^{t_f} \left(\tfrac{1}{2} w_{u1} \dot{\Theta}_1^2 + \tfrac{1}{2} w_{u2} \dot{\Theta}_4^2 \right) \, dt, \tag{13}$$

with $e_{dis} = b_d - b_c$ being the throwing error. b_d is the actual bin position and b_c the achieved throwing distance. The weights are chosen to be $w_1 = 100$, $w_{u1} = 0.1$, and $w_{u2} = 0.1$. For carrying out experiments on the real system, we had to ensure that no constraints of the system are violated. For this, the generalization was checked in simulation for a given range prior to the experiments. After releasing the ball the end effector velocity is very high. In order to not damage the system thereafter, also the deceleration phase has to be optimized. For this, the cost function is chosen to be

$$\begin{aligned} \min_{\dot{\Theta}(t)} J = & \\ & + \tfrac{1}{2} \left(w_{x2,f} e_{x_2,f}^2 + w_{x3,f} e_{x_3,f}^2 + w_{x5,f} e_{x_5,f}^2 w_{x6,f} e_{x_6,f}^2 \right) \\ & + \int_{t_0}^{t_f} \tfrac{1}{2} \left(w_{x2} e_{x_2}^2 + w_{x3} e_{x_3}^2 + w_{x5} e_{x_5}^2 + w_{x6} e_{x_6}^2 + w_{\Theta-1} \dot{\Theta}_1^2 + w_{\Theta_2} \dot{\Theta}_4^2 \right) \, dt, \end{aligned} \tag{14}$$

with $w_{x2,f} = 1$, $w_{x3,f} = 1.1$, $w_{x5,f} = 1$, $w_{x6,f} = 1.1$, $w_{x2} = 1$, $w_{x3} = 1.1$, $w_{x5} = 1$, $w_{x6} = 1.1$ and $w_{\Theta_1} = w_{\Theta_2} = 0.5$. The parameters $e_{x_i,f}$, $i = [2, 3, 5, 6]$ are the final joint positions and velocities $\mathbf{q}(t_f) = \dot{\mathbf{q}}(t_f) = \mathbf{0}$ of joints 1 and 4. e_{x_i} ensures convergence to the desired position and velocity over time.

In the considered example, $m = 3$ trajectories for different throwing distances were optimized and generalized with $n = 200$ Gaussian bases. Furthermore, non-optimized goals are simulated for a comparison between the desired and resulting goal positions, see Fig. 7.

The experimental setup is as follows. A bin is placed at a desired position along the throwing plane. With a Microsoft Kinect the distance d_{Bin} to the bin is measured and then, the OMF is able to generate the correct trajectory for hitting the bin in real-time (trajectory planning runs at 1 kHz). Figure 8 depicts a photo series of three different bin distances being experimentally validated.

Figure 7 shows the entire movement, consisting of the throwing task and the subsequent stopping motion. These trajectories visualize the comparison between online OMF trajectories and optimal ones. Again, the generalized motions are very close to the true optimal solution. Opening the hand for releasing the ball is triggered at $t = 1$s. The repetitive accuracy to hit the bin is very high (in average > 80 %), if the ball can be placed repetitively in the hand, which needs some training. In another experiment, we simply let HASy throw the ball to our ball catching demonstrator Justin, whose location is again measured. Justin is then able to track the incoming ball and catch it accordingly, see Fig. 9. Important to notice is that the systems are not communicating with each other.

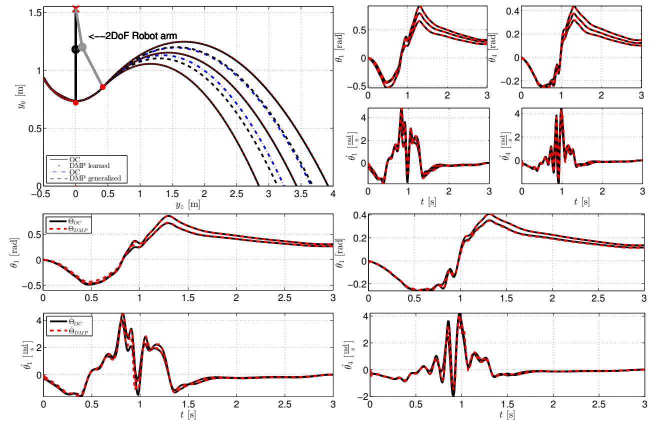

Fig. 7. Learning and generalization of optimal throwing movements. Top left: Optimal and generalized throwing trajectories in Cartesian space with flight trajectory of the ball. Four upper right: Learned trajectories and velocities in joint space with resulting optimal motor trajectories and velocities. Lower four: Generalized throwing trajectories and velocities in joint space with resulting motor trajectories in comparison with optimal trajectories.

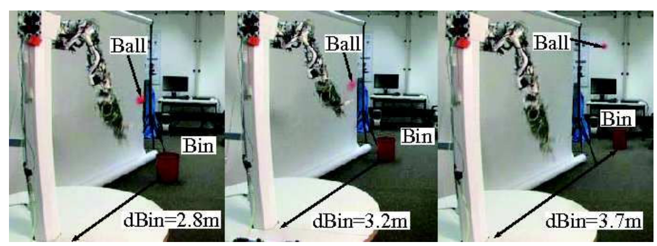

Fig. 8. Throwing sequence for varying target (bin). The bin is detected with a Kinect sensor and the distance d_{Bin} parameterizes $\mathbf{g}$. All trials were successful scores.

VI. Conclusion

In this paper, we presented an approach for generalizing optimal motions in real-time based on optimal sample trajectories that are then encoded into a DMP system for generalization. The developed optimal motion framework (OMF) performs a variety of optimal motions and was validated in simulation and experimentally on the anthropomorphic robot HASy. Our results demonstrate the possibility of combining optimality and generality even for its use in real-time. Learned optimal trajectories can be exactly reconstructed in real-time and even the extrapolation to other tasks show near-optimal behavior. The comparison of common movement generators and generalized point-to-point movements shows that the generalized trajectories produce significantly better results in terms of the considered cost function.

Fig. 9. Sequence for throwing to Justin. Justin is catching the ball.

ACKNOWLEDGMENT

This work has been partially funded by the European Commission's Sixth Framework Programme as part of the project SAPHARI under grant no. 287513.

REFERENCES

[1] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Hoppner, S. Jörg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimböck, S. Wolf, T. Wusthoff, and G. Hirzinger, "The DLR hand arm system," 2011, pp. 3175–3182.

[2] L. Biagiotti and C. Melchiorri, *Trajectory Planning for Automatic Machines and Robots*. Springer Berlin Heidelberg, 2008.

[3] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[4] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," 1999.

[5] S. M. LaValle and J. J. K. Jr., "Randomized kinodynamic planning," in *IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 473–479.

[6] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, S. Thrun, and L. Kavraki, *Principles of Robot Motion: Theory, Algroithms, and Implementation*. Cambridge: MIT Press, 2005.

[7] T. Kröger, "On-line trajectory generation: Straight-line trajectories," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 1010–1016, October 2011.

[8] K. Shin and N. D. McKay, "Minimum-time control of robotics manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, June 1985.

[9] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control if robotic manipulators along specified paths," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[10] O. v. Stryk and M. Schlemmer, "Optimal control of the industrial robot manutec r3," *Computational Optimal Control. International Series of Numerical Mathematics 115*.

[11] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 300–306.

[12] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783 –792, 2010.

[13] S. Haddadin, T. Laue, U. Frese, S. Wolf, A. Albu-Schäffer, and G. Hirzinger, "Kick it with elasticity: Requirements for 2050," *Robotics and Autonomous Systems*, vol. 57, pp. 761–775, 2009.

[14] S. Haddadin, M. Weis, S. Wolf, and A. Albu-Schäffer, "Optimal control for maximizing link velocity of robotic variable stiffness joints," *2011 IFAC World Congress (IFAC2011), Milano, Italy*, pp. 6863–6871.

[15] M. Garabini, A. Passaglia, F. A. W. Belo, P. Salaris, and A. Bicchi, "Optimality principles in variable stiffness control: the vsa hammer," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2011), San Francisco, USA*, pp. 3770 – 3775, 2011.

[16] S. Haddadin, F. Huber, and A. Albu-Schäffer, "Optimal control for exploiting the natural dynamics of variable stiffness robots," in *IEEE Int. Conf. on Robotics and Automation*, 2012.

[17] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR lightweight robot - lightweight design and soft robotics control concepts for robots in human environments," *Industrial Robot Journal*, vol. 34, no. 5, pp. 376–385, 2007.

[18] W. Townsend and J. Salisbury, "Mechanical design for whole-arm manipulation," in *Robots and Biological Systems: Towards a New Bionics?*, P. Dario, G. Sandini, and P. Aebischer, Eds. Springer, 1993, pp. 153–164.

[19] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," Handbook of Robotics, . chapter 59, 2008, 2008.

[20] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," *Computer Science and Neuroscience*.

[21] D. Lee, C. Ott, and Y. Nakamura, "Mimetic communication model with compliant physical contact in humanhumanoid interaction," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1684–1704, 2010.

[22] J. A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002, pp. 958–963.

[23] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Realtime reactive motion generation based on variable attractor dynamics and shaped velocities," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2010), Taipeh, Taiwan*, 2010.

[24] F. Stulp, E. Oztop, P. Pastor, M. Beetz, and S. Schaal, "compact models of motor primitive variations for predictible reaching and obstacle avoidance," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2009.

[25] K. Muelling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behavior Journal*, vol. 19, no. 5, pp. 359–376, 2011.

[26] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 763–768.

[27] N. Jetchev and M. Toussaint, "Trajectory prediction: learning to map situations to robot trajectories," in *International Conference on Machine Learning*, 2009, pp. 449–456.

[28] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 3719 –3726.

[29] A. Rao, D. Benson, C. Darby, M. Patterson, C. Francolin, I. Sanders, and G. Huntington, "Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method," *ACM Transactions on Mathematical Software*, vol. 22-60, no. 2, p. 39, 2010.

[30] A. Gams, T. Petric, L. Zlajpah, and A. Ude, "Optimizing parameters of trajectory representation for movement generalization: robotic throwing," *IEEE Journal of Basic Engineering*, pp. 161–166, June 2010.

[31] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and autonomous Systems*, pp. 1327–1339, 2012.

[32] A. Stark, "A study of business decisions under uncertainty: The probability of the improbable," Ph.D. dissertation, Boca Racon, USA Florida, 2010.

[33] J. J. Craig, *Introduction To Robotics*. Pearson Education International, 2005.