Optimization of sequential attractor-based movement for compact behaviour generation

Marc Toussaint, Michael Gienger, and Christian Goerick

***Abstract*— In this paper, we propose a novel method to generate optimal robot motion based on a sequence of attractor dynamics in task space. This is motivated by the biological evidence that movements in the motor cortex of animals are encoded in a similar fashion [1], [2] – and by the need for compact movement representations on which efficient optimization can be performed. We represent the motion as a sequence of attractor points acting in the task space of the motion. Based on this compact and robust representation, we present a scheme to generate optimal movements. Unlike traditional optimization techniques, this optimization is performed on the low-dimensional representation of the attractor points and includes the underlying control loop itself as subject to optimization. We incorporate optimality criteria such as e.g. the smoothness of the motion, collision distance measures, or joint limit avoidance. The optimization problem is solved efficiently employing the analytic equations of the overall system. Due to the fast convergence, the method is suited for dynamic environments, including the interaction with humans. We will present the details of the optimization scheme, and give a description of the chosen optimization criteria. Simulation and experimental results on the humanoid robot *ASIMO* will underline the potential of the proposed approach.**

I. INTRODUCTION

In real world environments movements of robotic systems have to fulfill many constraints such as collision avoidance, joint limit avoidance, task constraints, and other optimality criteria. When interacting with dynamic environments (e.g., a human user) the movement preparation process should be as short as possible. One approach to circumvent expensive trajectory planning in such dynamic environments is to use reactive feed-forward controllers and include additional potential functions to implement collision or joint limit avoidance. However, the resulting movement of such controllers is not optimal w.r.t. some global cost function. Further, reactive controllers can easily get trapped in deadlocks, it is difficult to avoid the limits that are imposed by the joint ranges and to deal with collisions. Many relevant tasks can only be solved when integrating some planning mechanism in the movement generation procedure. A key to efficient movement planning and optimization is an appropriate *movement representation*.

A. Trajectory planning and optimization

Standard trajectory planning or optimization approaches can solve many problems. A direct representation of the trajectory comprising the full robot state for each time slice bears a very high dimensional problem. Hence, existing research has focused, e.g., on using spline-encoding as a more compact representation for optimization. This is particularly the case in the field of industrial robot trajectory optimization. Examples for such systems utilize cost functions that are formulated in terms of dynamics [3], collision [4] or minimum jerk [5]. The splines are usually not formulated in the task space and the controller itself is left out of the optimization procedure: the tracking of the trajectories is realized with a different controller (see points (1) and (2) below). Zhang and Knoll [6] propose a representation based on *subgoals*. B-splines on joint level are optimized between these subgoals in order to generate a collision-free point-to-point motion. Nakamura and Hanafusa [7] propose an optimization scheme in which the end effector tracks a given path, while parameters for the redundant nullspace movement are optimized. The optimization problem is formulated as a 2 point boundary value problem. Other than in the approach presented in this paper, the end effector movement is pre-defined. General techniques like RRT [8], or randomized road maps [9], have been shown to solve difficult planning problems like the alpha puzzle, generating a complex balanced reaching trajectory for a humanoid robot, or plan footstep trajectories. These techniques consider a direct representation of the trajectory and focus on finding a feasible solution rather than optimizing the trajectory w.r.t. additional criteria (like minimal joint space length or maximal distance to joint limits).

When dealing with robots with a large number of degrees of freedom and low-level control of the order of a few milliseconds, a direct representation of the trajectory in this resolution is inefficient for optimization. However, also more compact representations like splines do not address the following issues: (1) Once an optimal trajectory was computed a low-level controller must be designed to follow this trajectory. This will inevitably lead to a deviation between the planned and executed movement, which has however not been considered as a part of the optimization problem. The alternative is to optimize a cost function which evaluates the trajectory that a given low-level controller actually generates when trying to follow the targets, rather than a cost function evaluating the target trajectory itself. (2) Many tasks can be described in a lower-dimensional task space (endeffector space). Performing optimization within this low-dimensional space is another opportunity to make the representation significantly more compact, especially for high-dimensional robotic systems like humanoids. We will propose to perform

Marc Toussaint is with the Machine Learning group at Technical University Berlin, Franklinstr. 28/29, 10587 Berlin, Germany; Michael Gienger and Christian Goerick are with the Honda Research Institute Europe, Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany.
mtoussai@cs.tu-berlin.de
michael.gienger@honda-ri.de
christian.goerick@honda-ri.de

optimization within this lower-dimensional space, which previous approaches have not yet addressed.

B. Motor primitives

An alternative view on efficient movement representation is motivated from previous work on motor primitives in animals [1], [2]. Inspired by these biological findings several researchers have adopted the concept of motor primitives to the realm of robotic movement generation. For instance, Amit & Mataric [10] propose a model in which a reactive controller is learnt. Its output is the attractor parameter of an underlying motor primitive. Williams et al. [11] analyze human handwriting data for motor primitives using a factored Hidden Markov Model to represent multiple concurrent, but non-reactive motor primitives. Ijspeert & Schaal et al. [12], [13], [14], [15], [16] focus on non-linear attractors and learning the non-linearities, e.g., in order to imitate observed movements. These approaches optimize the parameters of a single attractor system, e.g., such that this single motor primitive imitates as best as possible a teacher's movement. Data generated from exploratory trials is used to train the attractor dynamics. General optimization under redundancy is not considered.

C. Series of attractor dynamics

In this paper we consider sequences of attractors in task space as a compact movement representation. We select this representation, since it is closely related to the biological concept of motor primitives, and yields further advantages like robustness against perturbations and dynamical environments [12]. Furthermore, attractor points only represent an attracting point, whereas for splines, both initial and final conditions have to be defined. This makes them particularly suited in terms of their reuseability. Our approach may be viewed as a combination of classical optimization approaches with the idea of motor primitives which (1) optimizes a sequence of attractors rather than a single attractor, (2) bases the optimization on analytic robot kinematics rather than exploration, and (3) derives an optimization scheme which accounts for the redundant inverse kinematics used in the control architecture and thereby allows us to define arbitrary task spaces as a more compact representation of the movement. Unlike with spline optimization this representation is equivalently used during optimization and movement execution, i.e., the true movement generation process (the control attractor) is considered as part of the representation on which the optimization operates.

The outline of the paper is as follows. We first define the movement generation process based on a series of control attractors in section II. Based on this we derive in section III the gradient of a global cost function w.r.t. arbitrary parameters of these attractor. In this paper we focus on optimizing the location of attractors. Section IV will define precisely the global cost function we optimize, and section V explains some implementation issues. Finally, in section VI we apply our technique to several robotic movement problems under collision and joint limit avoidance constraints.

II. Movement representation with a sequence of task space attractors

We first describe the movement generation process given a sequence of control points $x^*_{1:K}$, with $x^*_k \in \mathbb{R}^m$. This is based on a previously presented whole body motion architecture [17].

Let $q_t \in \mathbb{R}^n$ be the robot's joint configuration state. For brevity we call the joint configuration state space the q-space. We assume we have a task space definition as given by a kinematic mapping $\phi : \mathbb{R}^n \to \mathbb{R}^m$. For instance, $\phi(q_t)$ may be the humanoid robot's left hand position (for $m = 3$). In our experiments we will consider also task spaces comprising the right hand's position and hand orientations.

If the duration of the total trajectory is T, each control point in the sequence $x^*_{1:K}$ is active for a time interval of length T/K. During this interval, the adjoining control points x^*_k and x^*_{k+1}, with $k = \lfloor tK/T \rfloor$, define some attractor dynamics in task space with the attractor point x^*_{k+1}. In general, any kind of attractor dynamics is admissible. In our implementation we use a non-oscillatory 2nd order dynamics superposed with a continuous shift. Details are given in appendix A.

The attractor dynamics generate a task space trajectory $x_t \in \mathbb{R}^m$. The current task space point x_t is translated to a joint control command via redundant inverse kinematic control,

$$q_{t+1} = q_t + J_t^{\#}(x_{t+1} - \phi(q_t)) - \alpha(I - J_t^{\#} J_t)\ W^{-1}(\partial_q H_t)^T \ . \tag{1}$$

Here, $J_t = \partial_q \phi(q_t)$ is the task space Jacobian at the joint configuration q_t, and H_t is a potential function over the q-space. This inverse kinematic control realizes a step in q-space that (up to first order approximation) results in a step $(x_{t+1} - \phi(q_t))$ in task space and additionally adds movement in the so-called nullspace following the gradient of H_t. We use $\alpha = 1/\Delta t$ as the weighting of the nullspace movement, where Δt is the real time span between to steps t and $t+1$.

The generalized pseudo-inverse $J^{\#}$ of a Matrix J is defined as

$$J^{\#} = W^{-1} J^T (J W^{-1} J^T)^{-1} \ , \tag{2}$$

and depends on a q-space metric W. In appendix B we derive the derivative $\partial_q J^{\#}$ of the generalized pseudo-inverse which we need during the optimization procedure.

III. Efficient Gradient computation

Since we want to find optimal solutions within only a few seconds – a time frame within which a fluent interaction with humans is possible – the key is a very efficient gradient-based optimization technique on the compact representation. In this section we derive an algorithm for the exact gradient computation of the global cost function with complexity $O(T)$. This primarily involves the propagation of gradient through the dependencies similar to backprop in neural networks.

The movement generation process can be summarized by equations (28), (29), and (1). During online execution,

all these equations are iterated forwardly. To derive analytic gradients, the whole movement generation process can be captured in the diagram at the top of table I as a network of functional (i.e., deterministic) dependencies between variables. This is similar to a Bayesian network, but based on deterministic rather than probabilistic dependencies. The diagram tells us how to compute global gradients since the chain rule implies that for any global functional C the *total* derivative w.r.t. some arbitrary variable z is generally

$$\frac{dC}{dz} = \sum_{\text{children } y_i \text{ of } z} \frac{\partial y_i}{\partial z} \frac{dC}{dy_i} . \tag{3}$$

For example, part c) in Table I lists all the chain rules for our particular network. This implies that we can backpropagate gradients through the whole control architecture using the local *partial* derivatives in order to derive analytic gradients for the control points $x^*_{1:K}$.

Table I summarizes all equations necessary to specify the controller and for the gradient computation. We assumed that the cost C is a function of the resulting robot trajectory $q_{0:T}$, which we introduce in detail in section IV. The chain rules following the network dependencies and equation (3) are summarized in part c) of Table I. These involve partial derivatives of the equations (A.1-A.5), which are summarized in part d) of the Table.

Based on this, the analytic gradient of the global cost function w.r.t. the control points $x^*_{1:K}$ can be computed with linear time complexity as follows.

In the *forward propagation step* we start with a given set of current control points $x^*_{1:K}$, then compute the ramp trajectory $r_{0:T}$ for these points, then compute the task trajectory $x_{0:T}$, then the $q_{0:T}$-trajectory, and finally the global cost C. All of this is done internally (without executing the movement on the robot).

In the *backward propagation step* we propagate the cost function gradients backward through the network using the chain rules. This involves first computing all the gradients dC/dq_t, then dC/dx_t, then dC/dr_t, and finally $dC/dx^*_{1:K}$. Since all computations in the forward and backward propagation are local, the overall complexity is $O(T)$.

IV. Optimality Criteria

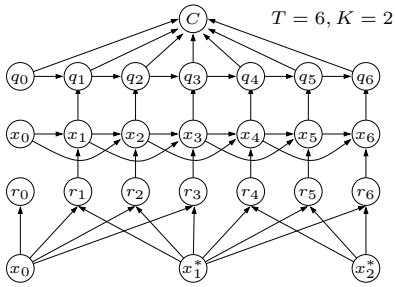
We consider a cost function of the general form

$$C = \sum_{t=0}^{T} g(q_t) + \sum_{t=0}^{T-1} h(q_t, q_{t+1}) . \tag{18}$$

Here, g is some function that subsumes cost criteria in the q-space such as collision or joint limit avoidance, and h is a function that subsumes costs for transitions in q-space, e.g., for penalizing long movements. We describe these terms in detail in the following.

In our applications we will define the cost term h to include

1) costs $c_1 = \sum_{t=1}^{T} (q_t - q_{t-1})^T W (q_t - q_{t-1})$ for the global length of the trajectory in q-space,



a) cost function:

$$C = \sum_{t=0}^{T} g(q_t) + \sum_{t=0}^{T-1} h(q_t, q_{t+1}) , \tag{A.1}$$

b) movement generation:

$$q_{t+1} = q_t + J_t^{\#}(x_{t+1} - \phi(q_t)) - \alpha \, (I - J_t^{\#} J_t) \, W^{-1} \, (\partial_q H_t)^T \tag{A.2}$$

$$x_{t+1} = x_t + \pi(x_t, x_{t-1}, r_{t+1}) \tag{A.3}$$

$$\pi(x_t, x_{t-1}, r_{t+1}) = a(r_{t+1} - x_t) + b(x_t - x_{t-1}) \tag{A.4}$$

$$r_t = (1-\tau) x^*_k + \tau x^*_{k+1} , \quad k = \lfloor tK/T \rfloor , \quad \tau = \frac{t - kT/K}{T/K} \tag{A.5}$$

c) chain rules following (3):

$$\frac{dC}{dq_t} = \frac{\partial C}{\partial q_t} + \frac{\partial q_{t+1}}{\partial q_t} \frac{dC}{dq_{t+1}} \tag{A.6}$$

$$\frac{dC}{dx_t} = \frac{\partial q_t}{\partial x_t} \frac{\partial C}{\partial q_t} + \frac{\partial x_{t+1}}{\partial x_t} \frac{dC}{dx_{t+1}} + \frac{\partial x_{t+2}}{\partial x_t} \frac{dC}{dx_{t+2}} \tag{A.7}$$

$$\frac{dC}{dr_t} = \frac{\partial x_t}{\partial r_t} \frac{dC}{dx_t} \tag{A.8}$$

$$\frac{dC}{dx^*_l} = \sum_t \frac{\partial r_t}{\partial x^*_l} \frac{dC}{dr_t} \tag{A.9}$$

d) partial derivatives:

$$\frac{\partial C}{\partial q_t} = g'(q_t) + h'^1(q_t, q_{t+1}) + h'^2(q_{t-1}, q_t) \tag{A.10}$$

$$\frac{\partial q_{t+1}}{\partial q_t} = I - J_t^{\#} J_t + (\partial_q J_t^{\#})(x_{t+1} - \phi(q_t)) - \alpha \, (I - J_t^{\#} J_t) \, W^{-1} \, (\partial_q^2 H_t)^T + \alpha \, \partial_q (J_t^{\#} J_t) \, W^{-1} \, (\partial_q H_t)^T \tag{A.11}$$

$$\frac{\partial q_t}{\partial x_t} = J_{t-1}^{\#} \tag{A.12}$$

$$\frac{\partial x_{t+1}}{\partial x_t} = 1 + \pi'^1(x_t, x_{t-1}, r_{t+1}) \tag{A.13}$$

$$\frac{\partial x_{t+2}}{\partial x_t} = \pi'^2(x_{t+1}, x_t, r_{t+2}) \tag{A.14}$$

$$\pi'^1(x_t, x_{t-1}, r_{t+1}) = -a + b , \quad \pi'^2(x_t, x_{t-1}, r_{t+1}) = -b \tag{A.15}$$

$$\frac{\partial x_t}{\partial r_t} = \pi'^3(x_{t-1}, x_{t-2}, r_t) \tag{A.16}$$

$$\frac{\partial r_t}{\partial x^*_l} = (1-\tau)\delta_{l=k} + \tau \delta_{l=k+1} , \quad \tau \text{ and } k \text{ depend on } t \text{ as above} \tag{A.17}$$

TABLE I

Functional network of the control architecture.

2) costs $c_2 = |\tilde{\phi}(q_T) - \tilde{\phi}(q_{T-1})|^2$ for the endeffector velocity at the end of the trajectory.

Further, we will define the cost term g to include

3) costs $c_3 = |\tilde{\phi}(q_T) - \hat{x}|^2$ for the offset of the final endeffector state to a target $\hat{x}$,
4) costs $c_4 = \sum_{t=0}^{T} Q(q_t)$ for collisions and proximities between collidable objects throughout the trajectory,
5) costs $c_5 = \sum_{t=0}^{T} H(q_t)$ for joint limit proximities.

Note that again W plays the role of a metric in q-space in the first cost term. Both, W and H, play a double (but consistent) role as part of the low-level controller (1) and as part of the global cost function. The "endeffector kinematics" $\tilde{\phi}$ that enters the cost terms 2 and 3 may by only a subspace of the task space defined by ϕ. This is clarified for each experiment. The global cost function C is the linear combination of these terms, $C = \sum_{i=1}^{5} c_i$. In the following we will more precisely define the collision cost function $Q(q)$ and the joint limit potential $H(q)$.

A. Collision avoidance

To obtain the collision cost and gradient, we loop through all collision-relevant pairs of bodies and sum their cost contributions. Each body is represented as a rigid primitive shape. Currently we use capped cylinders and sphere swept rectangles [18].

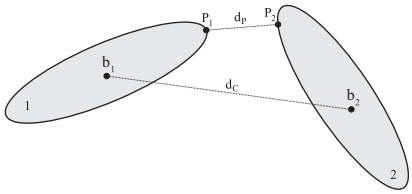


Fig. 1. Collision description

1) Collision cost: The cost associated with a pair of bodies is composed of two terms, one related to the distance between the closest points $d_p = |p_1 - p_2|$ and one related to the distance between their centers $d_c = |b_1 - b_2|$, see Figure 1.

To compute the closest point cost g_p, we set up three zones that are defined by the closest point distance $d_p = |p_1 - p_2|$ between two collision primitives. Figure 2 shows the linear, the parabolic and the zero cost zones, respectively. More formally, the closest point cost is

$$g_p = \begin{cases} s d_B (d_B - 2 d_p) & \text{for } d_p < 0 \\ s (d_p - d_B)^2 & \text{for } 0 \le d_p \le d_B \\ 0 & \text{for } d_p > d_B \end{cases} \tag{19}$$

with s defining the inclination of the gradient when penetrating. Similarly, the center point cost g_c shall only be active if the link distance has dropped below the distance d_B. The cost function will continuously be scaled with a factor being zero at $d_p = d_B$ and one if $d_p = 0$.

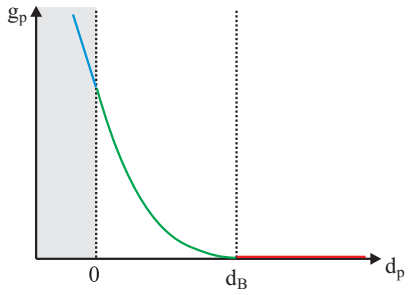


Fig. 2. Zones for the collision cost function determination.

$$g_c = \begin{cases} e^{-d_c} & \text{for } d_p < 0 \\ \left(1 - \frac{d_p}{d_B}\right) e^{-d_c} & \text{for } 0 \le d_p \le d_B \\ 0 & \text{for } d_p > d_B \end{cases} \tag{20}$$

The overall collision cost function is

$$Q(q) = \sum_{i}^{pairs} g_p(d_{p,i}) + g_c(d_{p,i}, d_{c,i}) \tag{21}$$

2) Distance gradient: Let us derive the gradient of the distance $d_p = |p_1 - p_2|$ w.r.t. the joint configuration q. Differentiating the distance $d_p = \sqrt{(p_2 - p_1)^T (p_2 - p_1)}$ with respect to the closest points p_1 and p_2 leads to

$$\frac{\partial d_p}{\partial p_1} = -\frac{1}{d_p} (p_2 - p_1)^T \qquad \frac{\partial d_p}{\partial p_2} = \frac{1}{d_p} (p_2 - p_1)^T \ . \tag{22}$$

If the collidable object is fixed to the environment, the partial derivative of the points with respect to the state is a $3 \times n$ zero matrix. If it corresponds to a body part or is attached to the robot's body (e.g. held in the hand), we use the closest point Jacobians $\frac{\partial p_1}{\partial q} = J_{p_1}$ and $\frac{\partial p_2}{\partial q} = J_{p_2}$. With (22) we get

$$\frac{\partial d_p}{\partial q} = \frac{1}{d} (p_2 - p_1)^T (J_{p_2} - J_{p_1}) \ . \tag{23}$$

Analogously we can compute the gradient of $d_c = |b_1 - b_2|$.

3) Closest point gradient: Differentiating eq. (19) with respect to the distance d_p, and inserting the distance gradient (23) leads to

$$\left(\frac{\partial g_p}{\partial q}\right)^T = \begin{cases} -2 s d_B (J_{p_2} - J_{p_1})^T (p_2 - p_1) & \text{for } d_p < 0 \\ 0 & \text{for } d_p > d_B \\ 2 s (d_p - d_B)(J_{p_2} - J_{p_1})^T (p_2 - p_1) & \text{else} \end{cases} \tag{24}$$

4) Center point gradient: Since the cost function g_c depends on the distance of the body centers d_c and on the closest point distance d_p, we need to apply the chain rule to get the gradient:

$$\frac{\partial g_c}{\partial q} = \frac{\partial g_c}{\partial d_c} \frac{\partial d_c}{\partial q} + \frac{\partial g_c}{\partial d_p} \frac{\partial d_p}{\partial q} \tag{25}$$

where

$$\frac{\partial g_c}{\partial d_c} = -\frac{d_B - d_p}{d_B} e^{-d_c} \qquad \frac{\partial g_c}{\partial d_p} = -\frac{1}{d_B} e^{-d_c} \tag{26}$$

and the respective distance gradient is given in eq. (23).

B. Joint limit avoidance

The joint limit avoidance cost function penalizes the weighted squared sum of the deviations of the joint angles q from their center position q_0.

$$H(q) = \frac{1}{2} \sum_{i=1}^{\text{dof}} w_i (q_i - q_{0,i})^2 \tag{27}$$

The weighting factors w_i are chosen such that the contributions of each individual joint is normalized with respect to its joint range.

V. Implementation decisions

We tested Conjugate Gradient, Levenberg-Marquard, and Rprop [19] optimization algorithms. Rprop turned out to be faster than Conjugate Gradient and Levenberg-Marquard, and therefore was used in the experiments.

Further, the control points $x^*_{1:K}$ are initialized as linear interpolation between the start position and the target position. In all our experiments, this initialization produces colliding movements.

The software is implemented in C/C++. All optimizations were carried out on a standard PC (Intel P4, 2.0GHz). After convergence, the control points were commanded to the humanoid robot *ASIMO* as a time-synchronized sequence. The real-time onboard controller generates the task-level trajectory with the identical attractor-based approach as utilized in the optimization. The Inverse Kinematics was implemented according to eq. (1), so that the resulting robot motion should be identical to the motion resulting from the optimization. Minor differences are due to the following points:

- To obtain faster convergence, the sampling time interval within the optimization procedure is selected somewhat larger than the sampling time on the real-time controller. The resulting discretization errors were negligible.
- The robot's upper body lateral position has been assumed to be constant, while the real robot uses these degrees of freedom to compensate momentum effects.
- The timing of the control point commands is not absolutely accurate, since they are transmitted over a network.

VI. Experiments

We considered three scenarios in the experiments. All of these involve the generation of movement where a reactive feed-forward controller would fail. Video recordings of the experiments accompany this paper. For all experiments we analyze the optimization performance by monitoring the cost decay during optimization. Figures 4, 6, and 8 display the cost decays for all weighted cost contributions and the total cost (which is the sum of all contributions). For further evaluation of the technique we analyze the time to find a first feasible (i.e., collision free and low proximity cost) solution, and the time to final convergence of the optimization w.r.t. the total cost C, see Figure 10. Table II summarizes the parameter settings used in the experiments.

Settings for experiments	A	B	C
K	8	8	4
steps	40	80	80
T	4sec	3sec	3sec
Collision avoidance cost	0.1	0.1	0.2
q-space path length cost	100	100	100
Joint limit avoidance cost	0.1	0.1	0.1
Target reaching cost	1000	1000	1000
Target velocity cost	10000	10000	10000

TABLE II
PARAMETER SETTINGS

A. Bimanual coordination

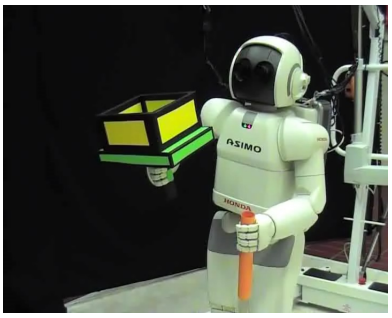


Fig. 3. Scenario of experiment A.

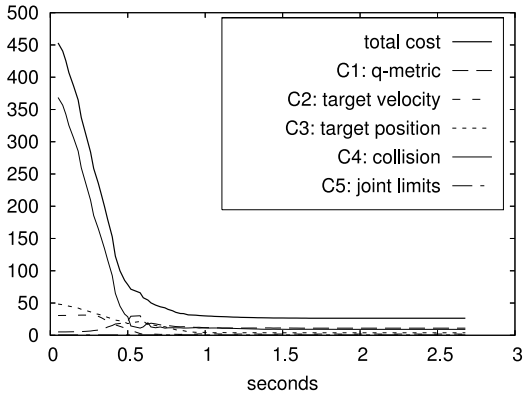


Fig. 4. Optimization performance in experiment A.

In the first experiment the robot holds a "bottle" in the left hand and a "box" (or tablet with very high rim) in the right hand, see Figure 3. The initial position of the bottle is low and of the box is high. The target is to place the bottle into the box, which involves moving both, the bottle and the box, in a coordinated way without collision. The solution found by the robot is to move the bottle in an arc upwards and into the box while at the same time moving the box with the right hand downwards below the bottle. A snapshot sequence of the motion is given in Figure 9, middle row, and a recording can be found in the accompanying video. The video also includes real time screen captures of the optimization procedure. The green points in the simulation environment illustrate the control points (only positions, not orientations). Initially they form a linear interpolation in

target space from the start position to the target. The way they move reflects their adaptation during the optimization procedure.

The task space ϕ ($= \tilde{\phi}$) in this experiment was defined 10-dimensional, comprising the positions of the left and right hand and the 2D polar orientation of the hand aligned axis for both hands.

Figure 4 displays the cost decay during optimization. From table 10 we see that a first solution is found already after 0.52 seconds.

B. Reaching over a wall

In the second experiment the robot has to reach for a balloon on a table hidden behind a front wall, see Figure 5. The initial position of the robot is such that the hands are low beside the body. In order to reach the balloon the robot must lift the arms above the wall and lower them again behind it, avoiding collision with the wall. A snapshot sequence of the motion is given in the bottom row of Figure 9, and a recording is found in the accompanying video.

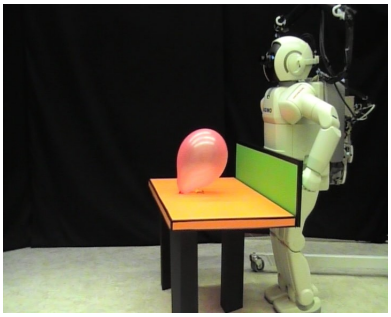


Fig. 5. Scenario of experiment B.

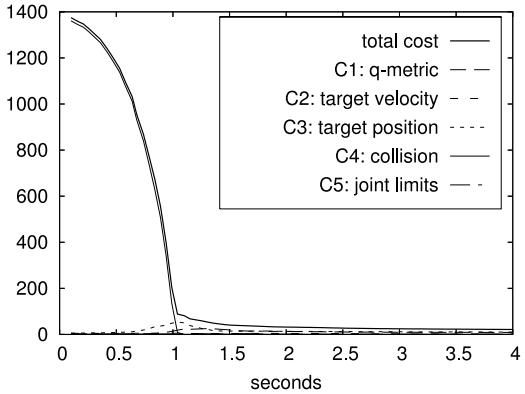


Fig. 6. Optimization performance in experiment B.

The task space ϕ was defined to be 10D as above. However, the target costs were only assigned to the final hand positions, not their orientations. That is, $\tilde{\phi}$ only comprises the 6D space of both hand positions. Figure 6 displays the cost decay during optimization and table 10 the performance times.

C. Reaching through a hole

In the third experiment we build a wall with a hole in it. The hole was low such that the robot has to lower its upper body in order to reach through the hole. The task was to reach a balloon placed behind the hole with the left hand, see Figure 7. A snapshot sequence of the motion is given in the top row of Figure 9. Interestingly, in the early phase of the movement the role of the control points is to impose attractors on the left hand that let the robot lower its body. Only the last two control points induce the forward movement through the hole.

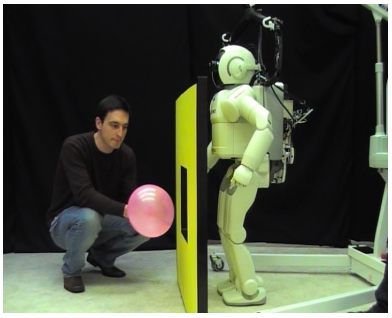


Fig. 7. Scenario of experiment C.

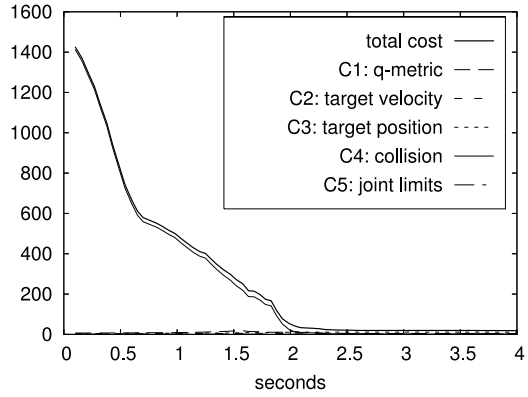


Fig. 8. Optimization performance in experiment C.

The task space ϕ was defined as 5D, comprising the position and orientation of the left hand. Including the orientation in the task space allows the control points to impose attractors to align the hand relative to the hole. However, the final hand orientation was not included in the target cost term ($\tilde{\phi}$ was 3D comprising only the hand position). Figure 8 displays the cost decay during optimization and table 10 the performance times.

VII. Conclusion

We introduced a representation of movements based on a sequence of attractor dynamics providing a compact and efficient representation for movement optimization and execution. This approach is inspired by biological findings in the field of motor primitives. The attractor dynamics act on the low-dimensional task space and are defined by a series

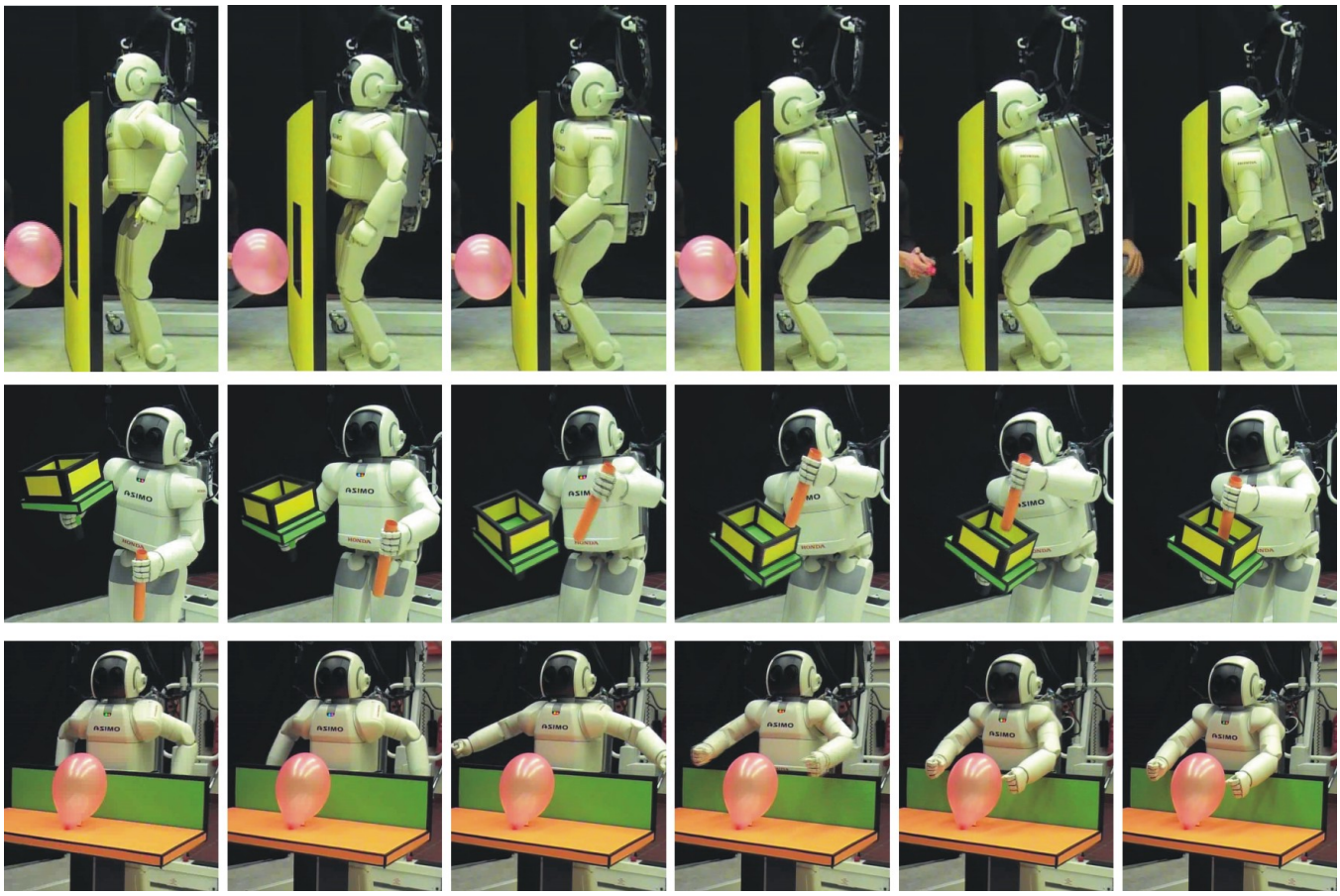


Fig. 9. Top row: Reaching through a hole (C). Middle row: Putting a bottle into a box (A). Bottom row: Reaching over a wall (B).

experiment	time to feasible solution	time to convergence
A	0.52 sec	1.47 sec
B	1.04 sec	6.73 sec
C	2.04 sec	5.07 sec

Fig. 10. Times for finding the first feasible (i.e., collision free and low proximity cost) solution and for final convergence ($\pm.1$) of the optimization w.r.t. the total cost C.

of control points in task space. Based on this description, we developed an optimization scheme that

- allows us to combine multiple criteria like collision avoidance, joint limit avoidance, trajectory length, and goal constraints in a global cost function,
- allows us to compose the task vector in a very flexible way, ranging from single handed tasks to coordinated bi-manual movements,
- finds feasible solutions within the range of 0.5 to 2 seconds – which is below the critical "patience" threshold for the interaction with humans, and
- incorporates the reactive control law into the optimization scheme.

The result is a sequence of attractor points in task space that lead to a robot movement accounting for the incorporated criteria in an optimal way.

The method is particularly useful for human-robot interaction in complex environments, e.g. when the robot has to reach around an obstacle that the human has just placed on the table. It is such scenarios that we are aiming for in future research.

Unlike previous approaches, the outcome of optimization is not a trajectory which needs an additional controller to be followed, but a series of control points which directly defines the reactive controller generating the movement on the real system. Since the underlying control law is incorporated into the optimization process, the resulting movement is always realizeable. This is different in many classical approaches, where optimized trajectories not necessarily can be tracked. The origin of this approach was the idea to combine previous work on motor primitives as model of reactive movement representation in animals with traditional optimization approaches.

The experiments corroborate the efficiency of the approach for movement generation on the humanoid robot *ASIMO*. In the considered problems a standard feed-forward controller would get trapped due to the collision constraints.

Finally, the current system has to two limitations: The number K of control points has to be fixed in advance and their timing is constrained to equal time intervals. Both points can be addressed using heuristics, e.g., for the addition

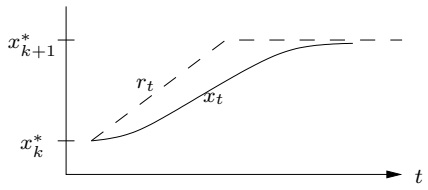


Fig. 11. Step response of the attractor system

of a new control point during the optimization procedure at intermediate time points. Future research will address these issues.

VIII. ACKNOWLEDGMENTS

MT is grateful to Honda RI Europe for their hospitality as a guest scientist. MT also acknowledges support by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3.

APPENDIX

A. 2nd order attractor dynamics

Given two adjoining control points x^*_k and x^*_{k+1} we shift the attractor point continuously from one to the other. This is captured by the linear interpolated trajectory $r_t \in \mathbb{R}^m$,

$$r_t = (1-\tau)x^*_k + \tau x^*_{k+1} \ , \ k = \lfloor tK/T \rfloor \ , \ \tau = \frac{t - kT/K}{T/K} \ . \tag{28}$$

In Figure 11 this is illustrated by the dashed line.

Point r_t is taken as attractor point to a second order dynamics which generates the task trajectory $x_t \in \mathbb{R}^m$,

$$x_{t+1} = x_t + \pi(x_t, x_{t-1}, r_{t+1}) \tag{29}$$

$$\pi(x_t, x_{t-1}, r_{t+1}) = a(r_{t+1} - x_t) + b(x_t - x_{t-1}) \ . \tag{30}$$

The step response of the scheme is depicted as solid line in Figure 11. We choose the coefficients a and b according to

$$a = \frac{\Delta t^2}{T^2_{\rm mc} + 2T_{\rm mc}\Delta t\xi + \Delta t^2} b \quad = \frac{T^2_{\rm mc}}{T^2_{\rm mc} + 2T_{\rm mc}\Delta t\xi + \Delta t^2} \tag{31}$$

with the relaxation time scale $T_{\rm mc} = 0.2$sec, the oscillation parameter $\xi = 1$, and the time span $\Delta t = T_{\rm real}/T$ between two steps t and $t+1$. This leads to a smooth non-overshooting approach.

B. Derivative of the pseudo-inverse

Using the matrix identity $\partial_q(A^{-1}) = -A^{-1}(\partial_q A)A^{-1}$, the partial derivative of the pseudo-inverse is

$$\begin{aligned}
\partial_q J^\# &= W^{-1}(\partial_q J^T)(JW^{-1}J^T)^{-1} \\
&\quad + W^{-1}J^T\Big[-(JW^{-1}J^T)^{-1}\Big((\partial_q J)W^{-1}J^T \cdots \\
&\qquad + JW^{-1}(\partial_q J^T)\Big)(JW^{-1}J^T)^{-1}\Big] \\
&= W^{-1}(\partial_q J^T)(JW^{-1}J^T)^{-1} - J^\#(\partial_q J)J^\# \\
&\quad - J^\# JW^{-1}(\partial_q J^T)(JW^{-1}J^T)^{-1} \\
&= -J^\#(\partial_q J)J^\# + (1 - J^\# J)\ W^{-1}(\partial_q J^T)\ (JW^{-1}J^T)^{-1}
\end{aligned} \tag{32}$$

REFERENCES

[1] F. A. Mussa-Ivaldi, S. F. Giszter, and E. Bizzi, "Linear combinations of primitives in vertebrate motor control," *Neurobiology*, vol. 91, pp. 7534–7538, 1994.

[2] E. Bizzi, A. d'Avella, P. Saltiel, and M. Tresch, "Modular organization of spinal motors systems," *The Neuroscientist*, vol. 8, pp. 437–442, 2002.

[3] A. Heim and O. v. Stryk, "Trajectory optimization of industrial robots with application to computer-aided robotics and robot controllers," *Optimization*, vol. 47, pp. 407–420, 1999.

[4] K. Schlemmer and G. Gruebel, "Real-time collision-free trajectory optimization of robot manipulators via semi-infinite parameter optimization," *International Journal of Robotics Research*, vol. 17, no. 9, pp. 1013–1021, 1998.

[5] K. Abdel-Malek, Z. Mi, J. Yang, and K. Nebel, "Optimization-based trajectory planning of the human upper body," *Robotica*, vol. 24, no. 6, pp. 683–696, 2006.

[6] J. Zhang and A. Knoll, "An enhanced optimization approach for generating smooth robot trajectories in the presence of obstacles," in *Proc. of the 1995 European Chinese Automation Conference*, London, 1995, pp. 263–268.

[7] Y. Nakamura and H. Hanafusa, "Optimal redundancy control of robot manipulators," *International Journal of Robotics Research*, vol. 6, 1987.

[8] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of IEEE Int'l Conf. on Robotics and Automation*, 2000.

[9] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[10] R. Amit and M. J. Matarić, "Parametric primitives for motor representation and control," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2002, pp. 863–868.

[11] B. Williams, M. Toussaint, and A. Storkey, "A primitive based generative model to infer timing information in unpartitioned handwriting data," in *Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, 2007.

[12] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2002.

[13] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, Cambridge, 2003, pp. 1523–1530.

[14] S. Schaal, "Movement planning and imitation by shaping nonlinear attractors," in *Proc. of the 12th Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT, 2003.

[15] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE Int. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, 2003.

[16] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and K. M., "Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives," in *Workshop on Robot Learning by Demonstration, IEEE Int. Conf. on Intelligent Robots and Systems*, 2003.

[17] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Proceedings of the 2005 5th IEEE-RAS International Conference on Humanoid Robots*, Los Angeles, USA, 2005, pp. 238–244.

[18] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time self collision avoidance for humanoids by means of nullspace criteria and task intervals," in *Proceedings of the 2006 5th IEEE-RAS International Conference on Humanoid Robots*, Genova, Italy, 2006, pp. 575–580.

[19] C. Igel and M. Hüsken, "Empirical evaluation of the improved Rprop learning algorithm," *Neurocomputing*, vol. 50(C),, pp. 105–123, 2003.