Whole-body Motion Planning – Building Blocks for Intelligent Systems

Michael Gienger, Mark Toussaint, Christian Goerick

2010

Preprint:

This is an accepted article published in Motion Planning for Humanoid Robots. The final authenticated version is available online at: https://doi.org/[DOI not available]

Chapter 1
Whole-body Motion Planning – Building Blocks for Intelligent Systems

M. Gienger, M. Toussaint and C. Goerick

Abstract Humanoid robots have become increasingly sophisticated, both in terms of their movement as well as their sensorial capabilities. This allows one to target for more challenging problems, eventually leading to robotic systems that can perform useful tasks in everyday environments. In this paper, we review some elements we consider to be important for a movement control and planning architecture. We first explain the whole-body control concept, which is the underlying basis for the subsequent elements. We then present a method to determine optimal stance locations with respect to a given task. This is a key element in an action selection scheme that evaluates a set of controllers within a parallel prediction architecture. It allows the robot to quickly react to changing environments. We then review a more global movement planning approach which casts the overall robot movement into an integral optimization problem, and leads to smooth and collision-free movements within interaction time. This scheme is then extended to cover the problem of grasping simple objects.

1.1 Introduction

While in its beginning, humanoid robotics research focused on individual aspects like walking, current systems have become increasingly sophisticated. Many humanoid robots are already equipped with full-body control concepts and advanced sensorial capabilities like stereo vision, auditory and tactile sensor systems. This is the prerequisite to tackle complex problems, such as walking and grasping in dy-

Dr.-Ing. Michael Gienger, Dr.-Ing. Christian Goerick
Honda Research Institute Europe GmbH, Carl-Legien-Strasse 30, 63073 Offenbach, Germany, e-mail: michael.gienger@honda-ri.de

Dr.rer.nat. Marc Toussaint
Technical University of Berlin, Franklinstr. 28/29, 10587 Berlin, Germany e-mail: mtoussai@cs.tu-berlin.de

namically changing environments. Motion planning seems to be a promising way to deal with this class of problem. State of the art planning methods allow one to flexibly account for different criteria to be satisfied. Further, many computationally efficient methods have been proposed (see [38–40] for a comprehensive overview), so that fast planning and replanning can be achieved in real-world, real-time problems.

In general, two problem fields in humanoid robot motion planning have emerged. One recent research focus is centered around solving the gait [9, 27] and footstep planning problem in dynamic environments [12, 45]. This is complemented by efforts to plan collision-free arm movements for reaching and grasping [4, 41], and to incorporate the dynamics of the objects to be manipulated [63].

However, there seems to be no approach to address all problem domains within a consistent architecture. In this article, we will present some steps in this direction. We start out with the whole-body control concept applied to our humanoid robot ASIMO in Section 1.2. We will explain the underlying robot model and derive the kinematics for the task and null space control. Based on the proposed model, we present a method to efficiently estimate an optimal stance location in Section 1.3. Reactive prediction and action selection is added with an architecture described in Section 1.4. It compares a set of different controller instances and selects the most suitable one according to their prediction result. However, this scheme has a limited time horizon. To generate movements that satisfy criteria throughout the whole trajectory, we present a controller-based optimization scheme in Section 1.5 in which we determine the attractor points describing the trajectory. The elements share a common basis, the whole-body control concept. The contribution finishes with the concept of *task maps* in Section 1.6. The fundamental idea is that there exists a space of feasible solutions for grasp problems that can be represented in a map. We will show how to generate and seamlessly integrate such maps into movement optimization, addressing the coupled problem of reaching and grasping in an integrated framework.

1.2 Models for Movement Control and Planning

While many movement planning approaches deal with navigation problems, this work will focus mainly on the problem of reaching and grasping with humanoid robots. Comprehensive kinematic models are particularly suited to describe the robot's end effector movement in an efficient and flexible way. In this section we briefly review the chosen redundant control concept: the general definition of task spaces, inverse kinematics and attractor dynamics to generate whole-body motion for high-dimensional humanoid robots.

Findings from the field of biology impressively reveal how efficiently movement is represented in living beings. Besides the well-known principle of movement primitives, it is widely recognized that movement is represented in various frames of reference, such as in eye centered, reach and grasp centered or object centered

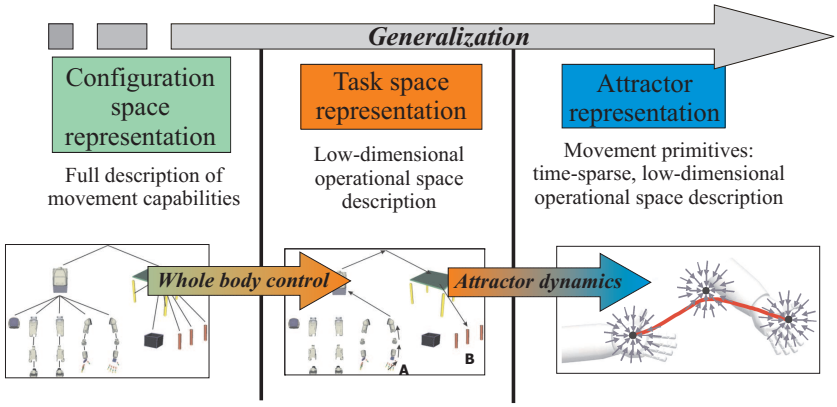

Figure 1.1 Task description

ones [15]. Egocentric frames describe movements with respect to the own body, and are a powerful representation when it comes to introducing invariance to a task. We borrow the above principle and use it as the underlying description of our control and planning models.

This work will focus on the large class of kinematically controlled robots. They differ from computed torque concepts such as [36] in that on the lowest level, the movement is represented in positions or velocities instead of torques. For this class of robots, the projection from a configuration space into task spaces is often done with redundant control schemes (e. g. *resolved motion rate control*, see [17,42,49]). Proper choice of the task description is a crucial element in humanoid robot control. Other than in other robotics domains, tasks may be carried out with two effectors.

Among the well-known trajectory generation methods, we chose a dynamical systems approach. This is closely related to the biological findings, and yields further advantages like robustness against perturbations and dynamical environments [32, 48]. The overall control architecture is summarized in Figure 1.1.

1.2.1 Control System

Kinematic control of redundant robots has been subject to extensive research. A popular method that allows one to split the control objective into a task and null space goes back to Liégeois [42] in the 1970s. Others have extended this approach towards introducing hierarchical task spaces [3, 16, 26, 53], to deal with collisions, singularities and ill-defined configurations [14,43,44,61] and have formulated criteria to map into the null space of such systems [11]. References [17, 49] give a good overview on these approaches.

We employ a motion control system that is based on [42]. The task space trajectories are projected into the joint space using a weighted generalized pseudo-inverse of the task Jacobian. Redundancies are resolved by mapping the gradient of a joint limit avoidance criterion into the null space of the motion. Details on the whole-body control algorithm are given in [18, 19]. The whole-body controller is coupled with a walking and balancing controller [31], which stabilizes the motion. Further, a real-time collision avoidance algorithm [61] protects the robot against self-collisions.

Setting up the controller equations is done by flexibly augmenting a task Jacobian J_{task} holding row-wise the Jacobians of the desired task descriptors that we derive in Section 1.2.1.1 (see also [18]).

$$\dot{q} = J^{\#}\,\dot{x}_{task} - \alpha\, N W^{-1} \left(\frac{\partial H}{\partial q}\right)^{T}. \tag{1.1}$$

Matrix $J^{\#}$ is a weighted generalized pseudo-inverse of J with metric W and null space projector N:

$$J^{\#} = W^{-1} J^{T} (J W^{-1} J^{T})^{-1} \qquad N = E - J^{\#} J. \tag{1.2}$$

Matrix E is an identity matrix. We chose a diagonal matrix W with elements proportional to the range of the corresponding joint. Scalar H is an arbitrary optimization criterion. Its gradient is mapped into the null space with projection matrix N and scalar α defining the step width. Vector $\dot{x}_{task}$ comprises a feedback term to minimize the tracking error (closed loop inverse kinematics or "CLIK").

1.2.1.1 Task Kinematics

The robot's kinematics and dynamics are described in the form of a tree structure depicted in Figure 1.2. The individual links are connected by degrees of freedom (joints) or fixed transformations. Further, the tree may also comprise objects from the environment. This allows derivation of the inverse kinematics equations not only with respect to a heel or world reference frame, but also to formulate task descriptors accounting for robot–object relations. In the forward kinematics pass (left), the transformations of all bodies are computed according to the current configuration space vector. The connectivity of the tree is such that the computation can be carried out starting from a root node and descends the branches of the tree. In the inverse kinematics pass (right), the desired Jacobians are computed. Since they depend only on the degrees of freedom that connect the respective body to the root, the connectivity in this direction is the shortest path towards the root node. We employ an efficient algorithm that allows one to compute the Jacobians by re-using the results of the forward kinematics pass. We will skip the details for brevity and refer the interested reader to [10].

In the following, a task is defined as the movement of one body with respect to any other belonging to the tree. This allows, for instance, one to describe the position

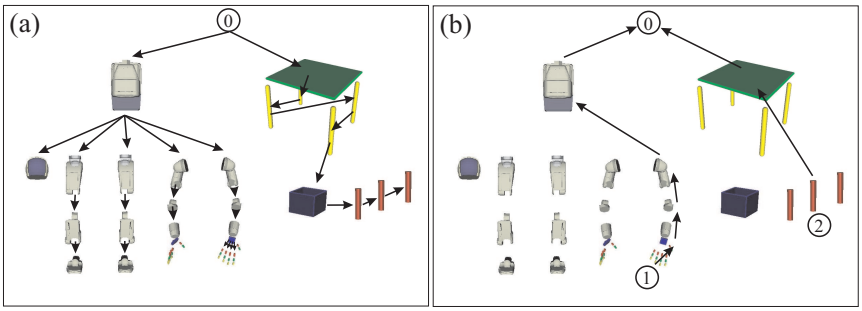

Figure 1.2 (a) Forward kinematics loop; and (b) loop for Jacobian computation

of one hand with respect to the other, the orientation of the head to the body, etc. It is also possible to describe robot link transformations with respect to objects in the environment, such as the position of the hand with respect to an object, or the direction of the gaze axis with respect to an object.

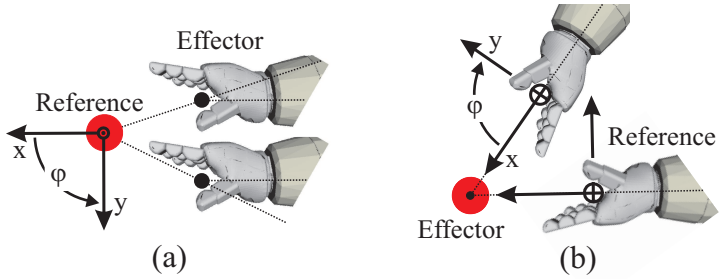

Figure 1.3 Relative hand–object task description: (a) with respect to the cylinder; and (b) with respect to the hand

The choice of the order of the relative coordinates yields some interesting aspects. This is illustrated in Figure 1.3 for a simple planar example. Representing the movement of the hand with respect to the cylinder results in Figure 1.3 (a). A coordinated hand–object movement has to consider three task variables $(x\ y\ \varphi)$. Switching the frame of reference and representing the object movement with respect to the hand, such as depicted in Figure 1.3 (b), leads to a description of the movement in hand coordinates. In this example, this might be advantageous, since the object is symmetric and can be approached from any side. While in the first case the task variables are dependent, in the second case φ and y are invariant and can be set to zero. There are many other examples, such as representing a gazing controller as an object in head-

centered coordinates which is “pointed” to by the focal axis, or a pointing controller in a similar way.

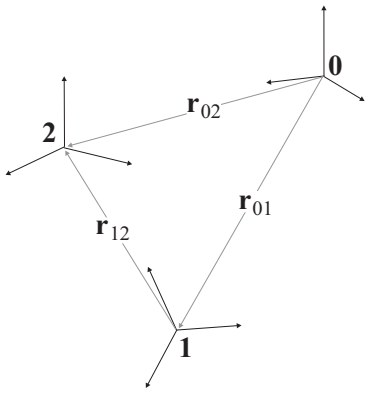

Figure 1.4 Relative effector kinematics

To mathematically formalize this concept, we look at the relative kinematics of an articulated chain, such as depicted in Figure 1.4. Coordinate frame 0 denotes its origin. Frame 1 is an arbitrary body which is connected to 0 through a set of joints. Body 2 shall be represented relative to body 1 with vector r_{12}. We now can write the (coordinate-free) kinematic equations as follows:

$$r_{12} = r_{02} - r_{01} \qquad \dot{r}_{12} = \dot{r}_{02} - \dot{r}_{01} + \omega_1 \times r_{12} \quad . \tag{1.3}$$

The last term of Equation 1.3 right is due to the rotation of body 1. Introducing the coordinate system in which the respective vector is represented as the left subindex and projecting the velocities into the state space with the respective Jacobians $\dot{r}_i = J_{T,i}\,\dot{q}$ and $\omega_i = J_{R,i}\,\dot{q}$, the differential kinematics becomes

$${}_1\dot{r}_{12} = A_{10}\left({}_0J_{T,2} - {}_0J_{T,1} + {}_0\tilde{r}_{12}^T\,{}_0J_{R,1}\right)\dot{q} = J_{T,rel}\,\dot{q}, \tag{1.4}$$

with J_T and J_R being the translational and rotational Jacobians, respectively, expression $\tilde{r} = r\times$ being a skew-symmetric matrix representing the outer product, and A_{10} being a rotation matrix from frame 0 to frame 1. If the reference (“1”) body corresponds to a fixed frame, it has no velocity and the corresponding Jacobian is zero. In this case, we get the standard differential end effector kinematics ${}_1\dot{r}_{12} = A_{10}\,{}_0J_{T,2}\dot{q}$.

The task descriptors for attitude parameters are computed slightly differently. This is due to the fact that many parametric descriptions such as Euler angles have discontinuities and ill-defined configurations (gimbal lock). We therefore project the tracking error directly on the Jacobians for the angular velocities:

$${}_1\omega_{12} = A_{10}\left({}_0J_{R,2} - {}_0J_{R,1}\right)\dot{q} = J_{R,rel}\,\dot{q}\,, \tag{1.5}$$

using the formulation of [13] for Euler angles. It is particularly elegant and avoids gimbal-locks. Similar feedback errors are formulated for 1D and 2D orientation task descriptors, for details see [18].

A well-investigated task descriptor is the overall linear momentum, which corresponds to the center of gravity velocity [34, 60]. It computes as

$$r_{cog} = \frac{1}{m} \sum_{i=1}^{bodies} m_i r_{cog,i} \qquad \dot{r}_{cog} = \frac{1}{m} \left\{ \sum_{i=1}^{bodies} m_i J_{T,cog,i} \right\} \dot{q} = J_{cog}\, \dot{q}\,. \tag{1.6}$$

Similarly, we can derive the task descriptor for the overall angular momentum L with respect to a non-accelerated reference. Tensor I denotes the inertia tensor of the respective body link:

$$L = \sum_{i=1}^{bodies} m_i r_{cog,i} \times \dot{r}_{cog,i} + I\omega = \left\{ \sum_{i=1}^{bodies} m_i\, \tilde{r}_{cog,i} J_{T,cog,i} + I_i J_{R,i} \right\} \dot{q} = J_{am}\, \dot{q}. \tag{1.7}$$

Another very simple but useful task descriptor is the movement of a single joint. The task variable is simply the joint angle, and the single joint Jacobian is a zero row vector with a single "one" entry at the column of the corresponding joint.

1.2.1.2 Null Space Control

In the following we present two of the null space criteria employed, namely a well-known joint limit avoidance criterion [42] and a collision avoidance criterion. The joint limit cost computes as

$$H_{jl}(q) = \frac{1}{2} \sum_{i=1}^{\mathrm{dof}} \left(\frac{q_i - q_{0,i}}{q_{max,i} - q_{min,i}} \right)^2 . \tag{1.8}$$

The contribution of each individual joint is normalized with respect to its joint range. To avoid collisions, we use the formulation in [62] and loop through all collision-relevant pairs of bodies, summing up their cost contributions. Each body is represented as a rigid primitive shape. Currently we use capped cylinders and sphere swept rectangles [61]. 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 $d_c = |C_1 - C_2|$, see Figure 1.5 (a). To compute the closest point cost g_p, we set up three zones that are defined by the closest point distance d_p between two collision primitives. Figure 1.5 (b) shows the linear, the parabolic and the zero cost zones, respectively. In the region between contact ($d_p = 0$) and a given distance boundary d_B, the closest point cost g_p is determined as a parabolic function, being zero at $d_p = d_B$ and having the slope s for $d_p = 0$. It progresses linearly for $d_p < 0$, and for $d_p > d_B$, it is zero.

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 be scaled continuously with

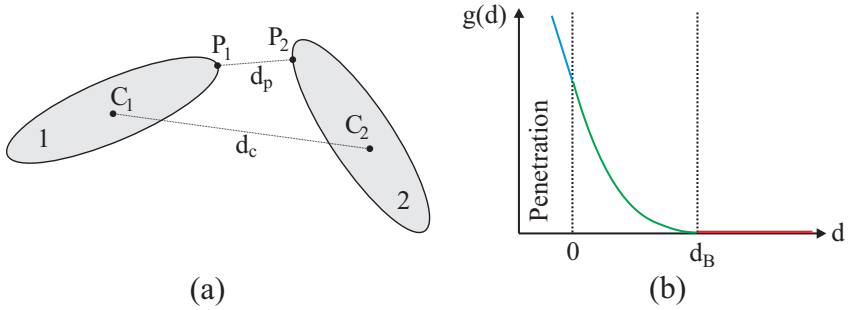

Figure 1.5 Zones for the collision cost function determination: (a) distance terms; and (b) cost zones

a factor zero at $d_p = d_B$ and one if $d_p = 0$. This cost adds an additional approximate avoidance direction, which is useful when the bodies are in deep penetration and the closest point direction is not well defined. Putting this together, the costs for one body pair become

$$g_p = \begin{cases} sd_B(d_B - 2d_p) & \text{for } d_p < 0 \\ s(d_p - d_B)^2 & \text{for } 0 \le d_p \le d_B \\ 0 & \text{for } d_p > d_B \end{cases} \quad g_c = \begin{cases} e^{-d_c} & \text{for } d_p < 0 \\ \left(1 - \frac{d_p}{d_B}\right) e^{-d_c} & \text{for } 0 \le d_p \le d_B \\ 0 & \text{for } d_p > d_B \end{cases} \tag{1.9}$$

with s defining the inclination of the gradient when penetrating. The overall collision cost is summed over all relevant body pairs as

$$H_{coll}(q) = \sum_{i}^{pairs} g_p(d_{p,i}) + g_c(d_{p,i}, d_{c,i}). \tag{1.10}$$

To derive the overall collision gradient, let us first derive the gradient of the distance $d_p = |p_1 - p_2|$ w.r.t. the joint configuration q. Differentiating with respect to the closest points p_1 and p_2 leads to

$$\frac{\partial d_p}{\partial p_1} = -\frac{1}{d_p}(p_2 - p_1)^T \qquad \frac{\partial d_p}{\partial p_2} = \frac{1}{d_p}(p_2 - p_1)^T. \tag{1.11}$$

If the collidable object is fixed to the environment, the partial derivative of the points with respect to the state is a $3 \times dof$ 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 Equation 1.11 we get

$$\frac{\partial d_p}{\partial q} = \frac{1}{d_p}(p_2 - p_1)^T (J_{p_2} - J_{p_1}) \,. \tag{1.12}$$

Analogously we can compute the gradient of $d_c = |C_1 - C_2|$. Differentiating Equation 1.9 with respect to the distance d_p, and inserting the distance gradient (Equation 1.12) leads to the closest point gradient

$$\left(\frac{\partial g_p}{\partial q}\right)^T = \begin{cases} -2s\frac{d_B}{d_p}(J_{p_2} - J_{p_1})^T(p_2 - p_1) & \text{for } d_p < 0, \\ 0 & \text{for } d_p > d_B, \\ 2s\frac{(d_p - d_B)}{d_p}(J_{p_2} - J_{p_1})^T(p_2 - p_1) & \text{else.} \end{cases} \tag{1.13}$$

The cost function g_c depends on the distance between the body centers d_c and on the closest point distance d_p, so we need to apply the chain rule to get the center point gradient:

$$\frac{\partial g_c}{\partial q} = \frac{\partial g_c}{\partial d_c}\frac{\partial d_c}{\partial q} + \frac{\partial g_c}{\partial d_p}\frac{\partial d_p}{\partial q} \tag{1.14}$$

where

$$\frac{\partial g_c}{\partial d_c} = -\frac{d_B - d_p}{d_B}e^{-d_c} \qquad \frac{\partial g_c}{\partial d_p} = -\frac{1}{d_B}e^{-d_c} \tag{1.15}$$

and the respective distance gradient is given in Equation 1.12. The overall collision gradient is

$$\frac{\partial H_{coll}}{\partial q} = \sum_i^{pairs} \frac{\partial g_d(i)}{\partial q} + \frac{\partial g_c(i)}{\partial q}. \tag{1.16}$$

1.2.2 Trajectory Generation

The quality of robot movements is very much related to the underlying trajectory generation. Popular trajectory generation methods use higher-order polynomials (splines) [9,51], time-optimal profiles [29], or employ attractor dynamics [32]. Polynomials are particularly suited for movements that require precise timing, such as generating step patterns, etc. Dynamical systems represent the time implicitly and can form attractor systems or periodic solutions. They are closely related to the biological findings, and yield further advantages like robustness against perturbations and dynamic environments. We apply a simple attractor system [18, 62] to the task elements to be controlled. The same attractor dynamics are applied to other controllers that are not related to the inverse kinematics, such as "closing the fingers to a power grasp", etc.

Given two 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 $r_t \in \mathbb{R}^m$. In Figure 1.6 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 $x_t \in \mathbb{R}^m$:

$$x_{t+1} = x_t + \pi(x_t, x_{t-1}, r_{t+1}) \tag{1.17}$$

$$\pi(x_t, x_{t-1}, r_{t+1}) = a(r_{t+1} - x_t) + b(x_t - x_{t-1})\ . \tag{1.18}$$

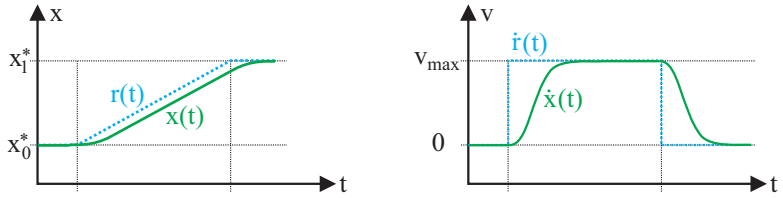

Figure 1.6 Step response of attractor system

The step response of the scheme is depicted as the solid line in Figure 1.6. We choose the coefficients a and b according to

$$a = \frac{\Delta t^2}{T_{\text{mc}}^2 + 2T_{\text{mc}}\Delta t\xi + \Delta t^2} \qquad b = \frac{T_{\text{mc}}^2}{T_{\text{mc}}^2 + 2T_{\text{mc}}\Delta t\xi + \Delta t^2}, \tag{1.19}$$

with a relaxation time scale T_{mc}, the oscillation parameter ξ, and the sampling time Δt. We select $\xi = 1$, which leads to a smooth non-overshooting trajectory and an approximately bell-shaped velocity profile.

1.2.3 Task Relaxation: Displacement Intervals

In common classical motion control algorithms the tracking of the desired trajectories is very accurate. In many cases the tracking accuracy of a reference trajectory is not very critical, or there are at least some phases where the accuracy may be lower than in others. For example, "reaching" or "pointing" a humanoid robot to an object does not necessarily have to precisely follow the commanded trajectory. A certain impreciseness is permitted, and sometimes even desired, since machine-like motions look somewhat strange when performed by humanoid or animal robots.

In this section, we introduce the concept of displacement intervals [19] in task space. These intervals describe regions around a given task variable, in which the tracking has to be realized. Analogous to the null space motion, the displacement intervals are exploited to satisfy one or several cost functions. By choosing appropriate criteria, the motion can be influenced in almost arbitrary manners, e.g., with respect to joint limit or collision avoidance, energy etc. In the following, the gradient of the joint limit avoidance cost function (Equation 1.8) is projected into the task interval. Its gradient with respect to the task space is

$$\frac{\partial H}{\partial x} = \frac{\partial H}{\partial q}\,\frac{\partial q}{\partial x} = \nabla H^T\, J^{\#}. \tag{1.20}$$

To describe the displacement interval in position coordinates, many solutions are thinkable: ellipsoids, cuboids or other 3D shapes. We choose a cuboid because the

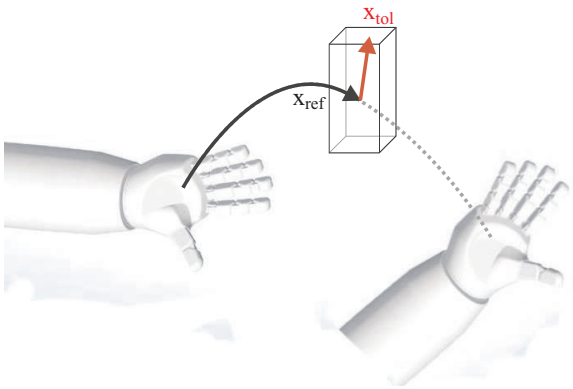

Figure 1.7 Displacement interval

computational complexity is low and the interval can be described in a physically transparent way. The cuboid can be conceived as a virtual box around the reference point, in which the effector is allowed to move (see Figure 1.7). If one dimension of this box is set to zero, the effector may move on a plane. Similarly, setting two box-dimensions to zero, the effector may move on a straight line in the third, free direction. Setting all interval dimensions to zero leads to the standard motion control tracking the reference trajectory exactly. Therefore, the proposed approach can be seen as an extension to common trajectory generation methods. Figure 1.8 left illustrates the computation of the linear displacement in each iteration. It computes as

$$\delta x_{disp} = -\alpha_{pos} \left(\frac{\partial H}{\partial x} \right)^T . \tag{1.21}$$

Displacement x_{disp} is superposed with the reference trajectory, and it is checked if the updated effector command lies within the permitted boundary. If the boundary is exceeded, the displacement vector x_{disp} is clipped to stay within the permitted region. Figure 1.8 (a) illustrates this for a 2D example.

An interval formulation for the effector axis direction is depicted in Figure 1.8 (b). The commanded effector axis a_{cmd} is allowed to move within a cone with symmetry axis being the reference axis and opening angle φ being the displacement boundary. The cone edge is of unit length, so that the depicted circumference is the intersection of the cone and a unit sphere. The tangential displacement on the unit sphere results from the gradients $\frac{\partial H}{\partial \omega_x}$ and $\frac{\partial H}{\partial \omega_y}$:

$$\delta a = -\alpha_{att} \begin{pmatrix} \frac{\partial H}{\partial \omega_x} \\ \frac{\partial H}{\partial \omega_y} \\ 0 \end{pmatrix} \times a_{cmd} . \tag{1.22}$$

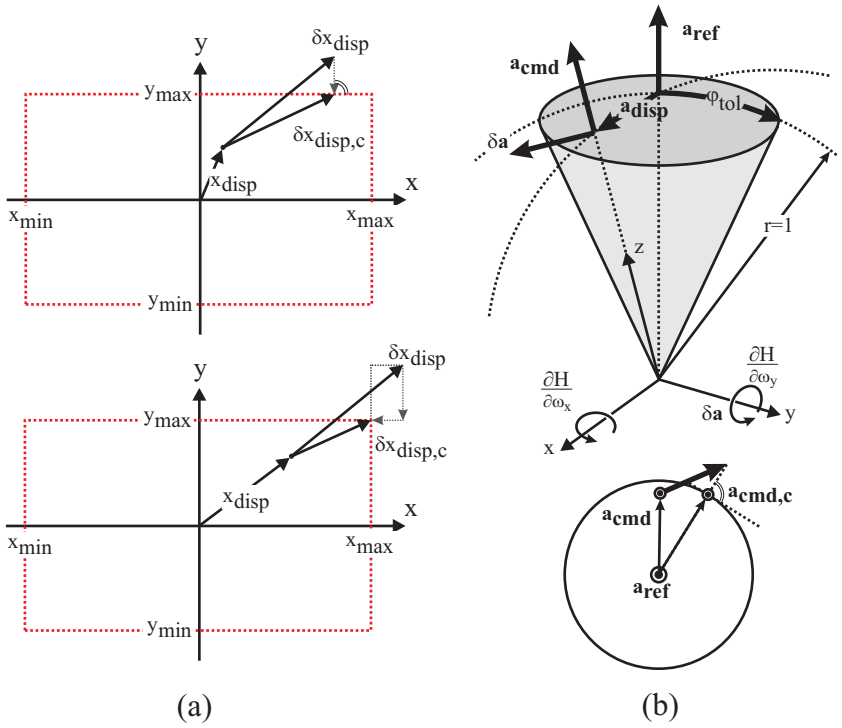

Figure 1.8 (a) Position and (b) attitude intervals

If the propagated command axis $a_{cmd} = a_{ref} + a_{disp}$ lies within the tolerance cone, no clipping has to be carried out. Otherwise, the command axis has to be clipped according to the lower parts of Figure 1.8.

1.3 Stance Point Planning

When carrying out a task with a humanoid robot, it is crucial to determine a good stance position with respect to the object to grasp or manipulate. There exist some interesting approaches, which sample and evaluate a reachable space for feasible solutions [24, 25, 64]. In this section, we will explain a potential-field-based method to determine an optimal stance. The underlying kinematic model is depicted in Figure 1.9. We introduce a stance coordinate system that is described by the translation and rotation in the ground plane, corresponding to the stance poses the robot can reach. The upper body (body frame) is described with respect to this stance frame, the successive links (arms, legs, head) are attached to the upper body. Now we set up the

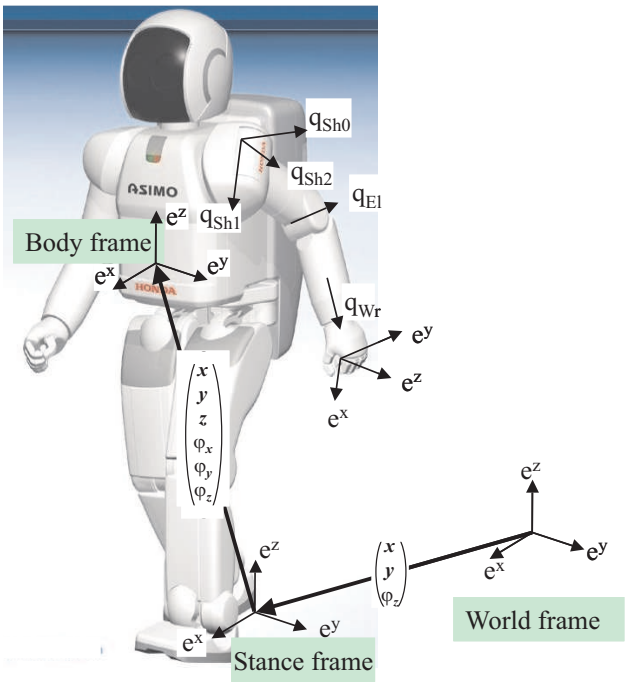

Figure 1.9 Kinematic model for stance pose optimization

controller equations according to Section 1.2. The important aspect is to unconstrain the three stance dofs, simply by assigning zeros to the corresponding column of the task Jacobian. This results in the stance dofs not being employed in the task space of the movement. However, they are still being utilized in the null space.

When assigning a target to the task vector, the controller equations will in each iteration make a step towards it, while shifting the stance coordinate system to a position and orientation that leads to a (local) minimum with respect to the employed null space criteria. A minimum can be found with regression techniques. Figure 1.10 illustrates this for the task of grasping a basket from a table. The task vector is composed of the following elements:

$$x_{task} = (x_{foot-l}^T \; \varphi_{Euler,foot-l}^T \; x_{foot-r}^T \; \varphi_{Euler,foot-r}^T \; x_{cog,xy}^T \; x_{hand-l}^T \; \varphi_{Polar,hand-l}^T)^T. \tag{1.23}$$

The tasks for the feet are chosen to be in a normal stance pose. The horizontal components of the center of gravity lie in the center of the stance polygon. The left hand position and orientation are aligned with the handle of the basket. The null space of the movement is characterized by a term to avoid joint limits (Equation

1.8), and another term to avoid collisions between the robot links and the table (Equation 1.16). The weight of the latter is increased in Figure 1.10 left to right. It can be seen that the resulting stance pose has a larger body-to-table distance for a higher collision weight. This scheme is very general as it can be applied to arbitrarily

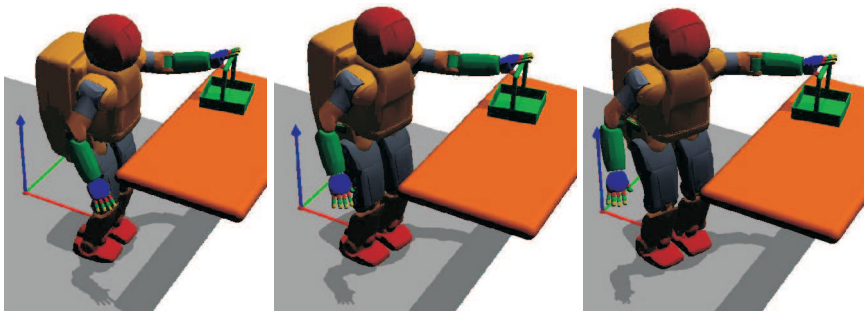

Figure 1.10 Optimal stance poses for differently weighted collision cost

composed task vectors. The resulting stance pose will always be a local optimum with respect to the null space criterion. Upon convergence, the resulting stance dofs can be commanded to a step pattern generator which generates a sequence of steps to reach the desired stance.

1.4 Prediction and Action Selection

With the control concept presented, ASIMO can walk around and safely reach towards objects while maintaining balance and avoiding self-collisions. However, the decision of *how to reach*, for instance, what hand to use or how to approach the object, is not tackled. In this section we will present an approach that solves this problem within a prediction and action selection architecture as depicted in Figure 1.11 (see [8, 22] for more details). The underlying idea is to connect the sensory (here visual) input to a set of predictors that correspond to simulated controllers of the robot. Each predictor solves the task associated with the sensory input in a different way. Within a strategy selection, these behavioral alternatives are continuously compared, and the command to the most suitable one is given to the physical robot.

First, we will explain the visual perception system employed which is based on a so called proto-object representation. Proto-objects are a concept originating from psychophysical modeling and can be thought of as coherent regions or groups of features in the field of view that are trackable and can be pointed or referred to without identification. Based on these stimuli, a prediction-based decision system selects

the best movement strategy and executes it in real time. The internal prediction as well as the executed movements incorporate the control system presented.

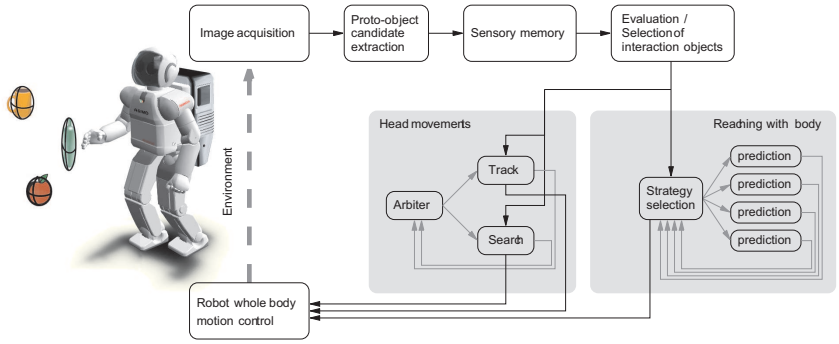

Figure 1.11 Overview of the system design

1.4.1 Visual Perception

To generate such proto-objects, we extract 3D ellipsoids from the visual input based on color segmentation and a disparity algorithm. The extracted blobs encode the position, metric size, and orientation of significant visual stimuli. They are stabilized and organized consistently as proto-objects in a *short term memory*. According to a set of extracted criteria, proto-objects are categorized into *found* if the object is seen in the visual scene, *memorized* if it has been found recently but is not seen currently, and *inaccurate* if it is only partially visible. Details of the chosen approach can be found in [8]. The 3D data and the above evaluation result are sent to the behaviors (search, track, reach). Each behavior can then extract the relevant information.

1.4.2 Behavior System

The output of the sensory memory is used to drive two different gazing behaviors: 1) searching for objects; and 2) tracking objects. Separate from these behaviors is a decision instance or *arbiter* [5] that decides which behavior should be active at any time. The decision of the arbiter is based on a scalar fitness value that describes how well a behavior can be executed. In this concrete case, tracking needs at least an inaccurate proto-object position to look at. Thus the tracking behavior will output a

fitness of 1 if any proto-object is present and a 0 otherwise. The search behavior has no prerequisites at all and thus its fitness is fixed to 1.

The search behavior is realized by means of an inhibition of return map with a simple relaxation dynamics. If the search behavior is active and new vision data is available it will increase the value of the current gaze direction in the map and select the lowest value in the map as the new gaze target. The tracking behavior is realized as a multi-tracking of 3D points. The behavior takes all relevant proto-objects and object hypotheses into account and calculates the pan/tilt angles for centering them in the field of view. The two visual interaction behaviors together with the arbiter switching mechanism show very short reaction times and have proven to be efficient to quickly find and track objects.

Similarly to the search and track behaviors, the reaching behavior is driven by the sensory memory. It is composed of a set of internal predictors and a strategy selection instance. Each predictor includes a whole-body motion controller described in Section 1.2.1 and a fitness function.

The key idea is to evaluate this set of predictors, each solving the given task in different ways. In the following, we look at the task of reaching towards an object and aligning the robot's palm with the object's longitudinal axis. This corresponds to a pre-grasp movement, which brings the hand in a suitable position to grasp an object. In a first step, the visual target is split up into different motion commands, with which the task can be achieved. Four commands are chosen: reaching towards the target with the left and right hand, both while standing and walking. While the strategies that reach while standing assume the robot model to have a fixed stance position, we apply an incremental version of the stance point planning scheme introduced in Section 1.3 to the strategies that involve walking. This leads to a very interesting property: the control algorithm will automatically find the optimal stance position and orientation with respect to the given target and the chosen null space criteria. If a walking strategy is selected, the best stance pose is commanded to a step pattern generator, which generates appropriate steps to reach the desired stance position and orientation. In each time step, the strategies compute their motion and an associated fitness according to the specific command. The fitness is composed of the following measures:

- **Reachability:** penalty if the reaching target cannot be reached with the respective strategy.
- **Postural discomfort:** penalizes the proximity to the joint limits when reaching towards the target.
- **"Laziness":** penalizes the strategies that make steps. This way, the robot prefers standing over walking.
- **Time to target:** penalizes the approximate number of steps that are required to reach the target. This makes the robot dynamically change the reaching hand also during walking.

The costs are continuously evaluated, and the strategy with the highest fitness is identified. The corresponding command is given to the physical robot. The robot is controlled with the identical whole-body motion controller that is employed for

the internal simulations. An interesting characteristic of the system is the temporal decoupling of real robot control and simulation. The strategies are sped up by a factor of 10 with respect to the real-time control, so that each strategy has converged to the target while the physical robot is still moving. From another point of view, the predictions could be seen as alternative results of a planning algorithm. A major difference is their incremental character. We use a set of predictors as continuously acting robots that each execute the task in a different way. The most appropriately acting virtual robot is mapped to the physical instance.

1.4.3 Experiments

The system as described above was tested many times with different people interacting with ASIMO with a variety of target objects. The scenario was always to have a human interaction partner who has an elongated object that was shown or hidden in various ways to ASIMO. The system is not restricted to only one object. If a number of objects are close to each other, the system will try to keep all objects in the field of view. If they are further apart, the objects leaving the field of view will be neglected after a short while and the system will track the remaining object(s).

Objects are quickly found and reliably tracked even when moved quickly. The robot will reach for any elongated object of appropriate size that is presented within a certain distance — from 20 cm to about 3 m. ASIMO switches between reaching with the right and left hand according to the relative object position with some hysteresis. It makes steps only when necessary. Figure 1.12 shows a series of snapshots taken from an experiment. From second 1–7, ASIMO is reaching for the bottle with its right hand. At second 8, the object becomes out of reach of the right hand, and the strategy selection mechanism selects the left hand reaching strategy, still while the robot is standing. At second 12, the object can not be reached with the left hand while standing. The strategy selection mechanism now selects to reach for the object with the left hand while walking towards it. The whole-body motion control generates smooth motions and is able to handle even extreme postures, which gives a very natural and human-like impression even to the casual observer. For more details of this system see [8].

1.5 Trajectory Optimization

The prediction architecture presented in the previous section allows the robot to dynamically act and react in a simple, but dynamically changing environment. However, it does not consider the overall movement throughout the trajectory, which is relevant when it comes to acting in a more difficult environment, with the potential danger to collide with objects, etc. In such cases more comprehensive planning and optimization schemes are required.

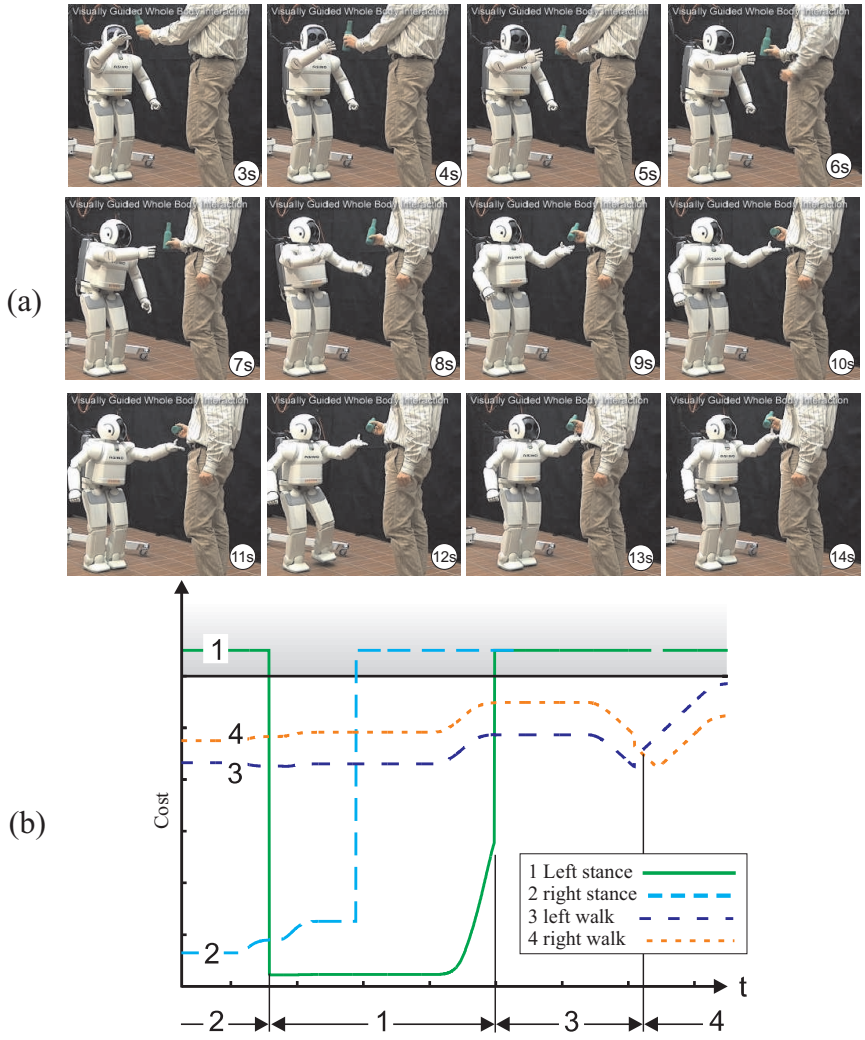

Figure 1.12 (a) Reaching towards a bottle; and (b) corresponding costs of predictor instances

A lot of research in this field has focused 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 of such systems utilize cost functions that are formulated in terms of dynamics [30, 50], collision [56] or minimum jerk [1].

General techniques like rapidly exploring random trees (RRTs) [37], or randomized road maps [35], 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.

An alternative view on efficient movement representation is motivated from previous work on motor primitives in animals [6, 48]. Inspired by these biological findings several researchers have adopted the concept of motor primitives to the realm of robotic movement generation. For instance, Ijspeert et al. and Schaal et al. [32, 33, 54, 55] 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.

In this section, we will review an attractor-based optimization scheme [62]. It incorporates the robot's whole-body controller of Section 1.2.1 into the optimization process, and finds a sequence of task space attractors from Section 1.2.2 describing the optimal movement. The key idea is to optimize a scalar cost function by finding an optimal sequence of task space attractor vectors which determines the robot's motion. We consider an integral cost function over the movement in the general form of Equation 1 of Table 1.1. It is split into two terms. Function h subsumes costs for transitions in joint space and depends on the current and the previous time steps. It is suited to formulate criteria like the global length of the trajectory in joint space and the end effector velocity at the end of the trajectory:

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

Function g subsumes cost criteria that depend on single time steps. It is suited to account for costs that depend on the posture of the robot. We formulate criteria to account for the offset of the final end effector state to a target, collisions and proximities between collidable objects throughout the trajectory, and joint limit proximities:

3. costs $c_3 = |\tilde{\phi}(q_T) - \hat{x}|^2$ for the offset of the final end effector state to a target $\hat{x}$;
4. costs $c_4 = \sum_{t=0}^{T} H_{coll}(q_t)$ for collisions and proximities between collidable objects throughout the trajectory, see Equation 1.10;
5. costs $c_5 = \sum_{t=0}^{T} H_{jl}(q_t)$ for joint limit proximities, see Equation 1.8.

The global cost function C is the linear combination of these terms, $C = \sum_{i=1}^{5} c_i$.

The movement generation process can be summarized by Equations 5, 4, and 2 in Table 1.1. To derive analytic gradients, the whole movement generation process can be captured in the diagram in Figure 1.13 as a network of functional 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,

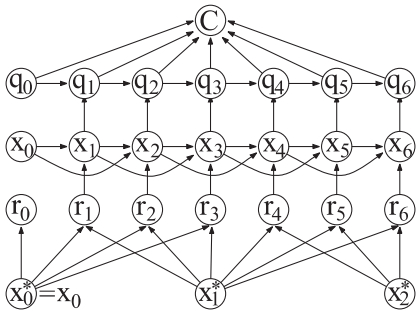

Figure 1.13 Functional network of the control architecture

$$\frac{dC}{dx^*} = \sum_{\text{children } y_i \text{ of } x^*} \frac{\partial y_i}{\partial x^*} \frac{dC}{dy_i} \,. \tag{1.24}$$

The gradient computation is carried out in a forward and a backward computation step. In the *forward propagation step* we start with a given set of current attractor points $x^*_{1:K}$, then compute the task space trajectory $x_{0:T}$, then the $q_{0:T}$-trajectory, and finally the global cost C. In the *backward propagation step* we propagate the cost function gradients backward through the network using the chain rules. This involves first computing gradients dC/dq_t, then dC/dx_t, and finally $dC/dx^*_{1:K}$. Since all computations in the forward and backward propagation are local, the overall complexity is $O(T)$.

Figure 1.14 (a) shows a snapshot series of an experiment. The scenario has been chosen such that a purely reactive controller would fail. The robot holds a cylinder in the left hand and a box in the right hand. 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. The task space in this experiment was defined 10D, comprising the positions of the left and right hand and the 2D polar orientation of the hand aligned axis for both hands. Figure 1.14 (b) displays the cost decay during optimization. A first collision-free solution is found after only 0.52 s, the final solution converged after 1.5 s. 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. More experiments are given in [62].

1.6 Planning Reaching and Grasping

In this section, we will build on the movement optimization scheme presented and present an integrative approach to solve the coupled problem of reaching and grasp-

Table 1.1 Costs and gradients underlying the optimization

cost function

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

movement generation

$$q_{t+1} = q_t + J_t^{\#}(x_{t+1} - \phi(q_t)) - \alpha\ (I - J_t^{\#} J_t)\ W^{-1}\ (\partial_q H_t)^T \tag{2}$$
$$x_{t+1} = x_t + \pi(x_t, x_{t-1}, r_{t+1}) \tag{3}$$
$$\pi(x_t, x_{t-1}, r_{t+1}) = a(r_{t+1} - x_t) + b(x_t - x_{t-1}) \tag{4}$$
$$r_t = (1-\tau)x_k^* + \tau x_{k+1}^* \,, \quad k = \lfloor tK/T \rfloor \,, \quad \tau = \frac{t - kT/K}{T/K} \tag{5}$$

chain rules following Equation 1.24

$$\frac{dC}{dq_t} = \frac{\partial C}{\partial q_t} + \frac{\partial q_{t+1}}{\partial q_t}\ \frac{dC}{dq_{t+1}} \tag{6}$$
$$\frac{dC}{dx_t} = \frac{\partial q_t}{\partial x_t}\ \frac{\partial C}{\partial q_t} + \frac{\partial x_{t+1}}{\partial x_t}\ \frac{dC}{dx_{t+1}} + \frac{\partial x_{t+2}}{\partial x_t}\ \frac{dC}{dx_{t+2}} \tag{7}$$
$$\frac{dC}{dr_t} = \frac{\partial x_t}{\partial r_t}\ \frac{dC}{dx_t} \tag{8}$$
$$\frac{dC}{dx_l^*} = \sum_t \frac{\partial r_t}{\partial x_l^*}\ \frac{dC}{dr_t} \tag{9}$$

partial derivatives

$$\frac{\partial C}{\partial q_t} = g'(q_t) + h'^1(q_t, q_{t+1}) + h'^2(q_{t-1}, q_t) \tag{10}$$
$$\frac{\partial q_{t+1}}{\partial q_t} = I - J_t^{\#} J_t + (\partial_q J_t^{\#})(x_{t+1} - \phi(q_t)) - \alpha\ (I - J_t^{\#} J_t)\ W^{-1}\ (\partial_q^2 H_t)^T + \alpha\ \partial_q (J_t^{\#} J_t)\ W^{-1}\ (\partial_q H_t)^T \tag{11}$$
$$\frac{\partial q_t}{\partial x_t} = J_{t-1}^{\#} \tag{12}$$
$$\frac{\partial x_{t+1}}{\partial x_t} = 1 + \pi'^1(x_t, x_{t-1}, r_{t+1}) \tag{13}$$
$$\frac{\partial x_{t+2}}{\partial x_t} = \pi'^2(x_{t+1}, x_t, r_{t+2}) \tag{14}$$
$$\pi'^1(x_t, x_{t-1}, r_{t+1}) = -a + b \,, \quad \pi'^2(x_t, x_{t-1}, r_{t+1}) = -b \tag{15}$$
$$\frac{\partial x_t}{\partial r_t} = \pi'^3(x_{t-1}, x_{t-2}, r_t) \tag{16}$$
$$\frac{\partial r_t}{\partial x_l^*} = (1-\tau)\delta_{l=k} + \tau \delta_{l=k+1} \,, \quad \tau \text{ and } k \text{ depend on } t \text{ as above} \tag{17}$$

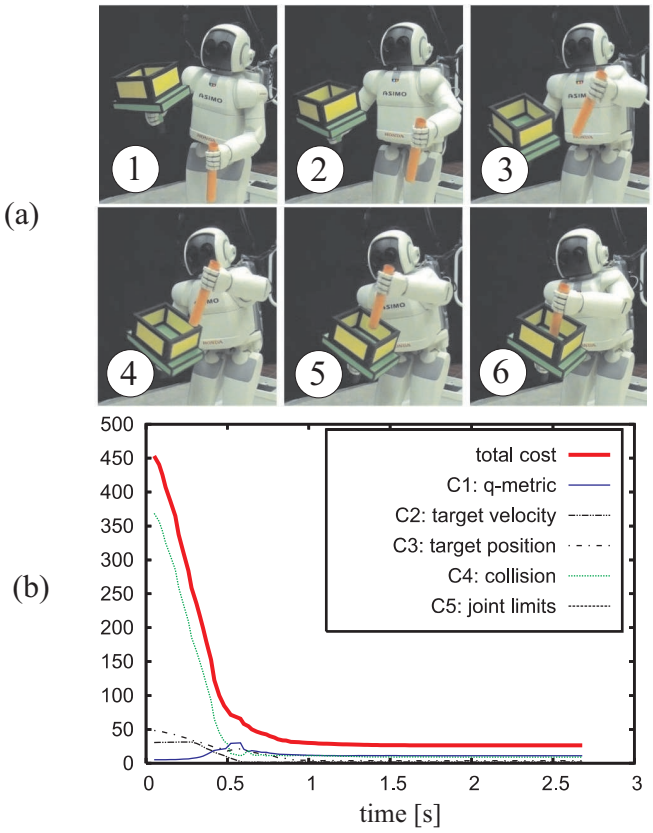

Figure 1.14 (a) Putting a cylinder into a box; and (b) cost decay

ing an object in a cluttered environment. While finding an optimal grasp is often treated independently from reaching to the object, in most situations it depends on how the robot can reach a pregrasp pose while avoiding obstacles. In essence, we are faced with the coupled problem of grasp choice and reaching motion planning.

Most literature on grasp optimization focuses on the grasp itself, isolated from the reaching movement. For instance, [59] review the various literature on defining grasp quality measures, [57] learn which grasp positions are feasible for various objects, [28] efficiently compute good grasps depending on how the objects shall be manipulated, and [46] simplify the grasp computation based on abstracting objects into shape primitives. The coupling to the problem of reaching motion optimization is rarely addressed. In [52], reaching and grasping is realized by reactive control primitives. A recent approach [4] makes a step towards solving the coupled problem by including an "environment clearance score" in the grasp evaluation measure. In that way, grasps are preferred which are not prohibited by immediate obstacles

directly opposing the grasp. Still, the full reaching motion is neglected in the grasp evaluation.

We approach this problem by proposing an object representation in terms of an object-specific task map which can be learnt from data and, during movement generation, efficiently coupled into a movement optimization process. Further, we generalise the optimization scheme presented in Section 1.5 to cope with such task maps [20, 21].

With the term *task map*, we refer to a map that comprises a set of sampled task coordinates, each associated with a scalar quality measure. In previous work [20], we proposed, for instance, to represent a set of hand–object pregrasp poses with respect to a failure/success criterion. These maps generally replace the concept of one explicit reaching target by the concept of a whole manifold of feasible targets in the task space. This relaxes the constraints imposed on the subsequent movement optimization process, which is particularly beneficial to improve other criteria governing the movement. If the chosen quality measure can be determined with the robot's sensors, it is further possible to build up or refine task maps in real experiments.

In the following, we will focus on simple "power grasps". However, the approach is not limited to a certain grasp. It is possible to represent different grasp types (e.g., precision grasps, etc.) in several task maps. The concept even holds for bi-manual grasp strategies.

1.6.1 Acquisition of Task Maps for Grasping

Learning a task map requires exploring many different grasps on a specific object. The first question is how different grasp trials are sampled. The second, how each trial is evaluated. Previous approaches consider an exhaustive sampling over a grid [4]. To reduce the number of samples, we proposed to use RRTs [20]. While this technique is very fast, it is hard to generate a very dense set of samples. Further, when the set of feasible grasps separates into disjoint clusters, RRTs typically explore only one of the clusters. Here we will focus on an approach similar to the heuristics proposed in [4] and [58]. We assume that the robot can (visually) acquire a rough estimate of the object volume. Using the approximate shape information we can sample a random point from the volume and compute a pregrasp posture. For this, we initialize the hand inside the object volume. The hand is then retracted so as to get in palm contact with the object. Subsequently the finger joints are closed until they contact the objects surface. For this grasp, we collect the following data: (1) the hand position and orientation in coordinates relative to the object frame – this 6D point $g \in \mathbb{R}^6$ will become the vector of control parameters in the task space; (2) the contact points, normals and penetration between the finger segments and the object – this information is used to compute a quality measure based on force closure, which is a well-known measure determining the ability of a grasp to resist external forces [47].

While this learning phase can be executed on a real robot, we use realistic simulations to speed up the process. The force closure is computed from simulated tactile sensor positions and normals, excluding those that have zero pressure. In that way we collect a data set consisting of control parameters in $\mathbb{R}^6$ and the corresponding force closure scalars.

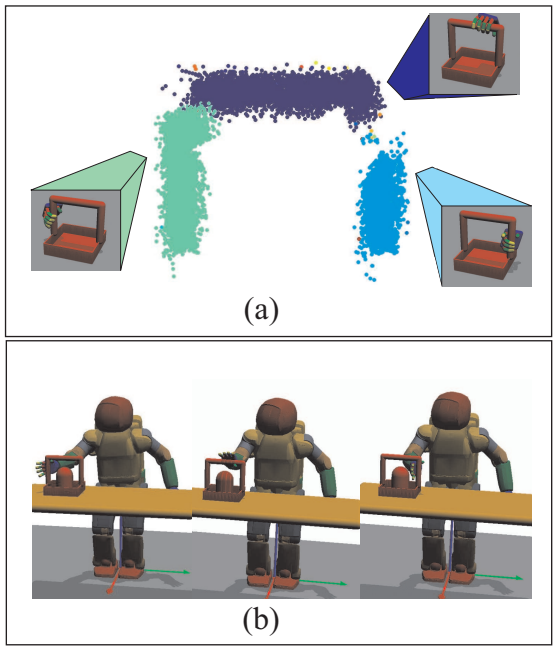

Figure 1.15 (a) Disjoint clusters in sampled task map. Solutions with the thumb and palm pointing upwards have been left out. (b) Initialization poses for different clusters.

For many realistic objects, the task map will consist of disjoint clusters which form qualitatively different solutions. Consider, for instance, a simple basket with handle that can be grasped on the top bar or on each of the side bars. The orientations and positions of feasible grasps change discontinuously from bar to bar, forming a set of clusters. We employ a Euclidean distance-based hierarchical clustering approach to extract a set of qualitatively different solutions. The chosen algorithm does not make any a priori assumption on the number of clusters.

Figure 1.15 (a) displays the extracted clusters for the given example. Clusters are formed by grasps applied to the left, right and top handle of the basket. In the figure, only the position elements of the 6D task map parameters are visualized. We only consider clusters of samples that have a significant size, eliminating outliers that comprise only a few samples.

1.6.2 Integration into Optimization Procedure

To find an appropriate initialization for the optimization problem, the target posture at the end of the movement is computed according to each cluster center, using inverse kinematics. The different target postures are then compared by an *end-state comfort* value, motivated from psychological studies. It is based on the joint limit and collision costs given in Equations 1.8 and 1.10. The best solution determines the target task vector to initialize the optimization problem. Figure 1.15 (b) illustrates the initializations for a given location of the basket on the table.

To incorporate the learnt task maps seamlessly into the optimization process, we formulate criteria to account for both the proximity to the nearest solution in the task map manifold and the quality of the resulting grasp: The *proximity criterion* enforces a smooth and collision-free movement into the manifold of feasible grasps represented in the map. It "pulls" the final grasp towards the manifold of valid preshape postures in the course of the optimization. The *quality criterion* evaluates the quality of the final grasp. It is based on the force-closure quality measure for each task map sample and guides the movement towards a preshape posture that leads to a high-quality grasp.

The proximity criterion will replace the distance of the target end-effector state and contribute to the cost function g in Equation 1 during motion optimization. Given the final task state at the last time step (e.g., the hand position and orientation relative to the object), we compute the nearest element x_{map} in the task map. Now we define a cost:

$$g_p = (x_t^{\text{rel}} - x_{\text{map}})^T W_{\text{map}} (x_t^{\text{rel}} - x_{\text{map}}) \; . \tag{1.18}$$

The metric W_{map} accounts for the difference in the linear and angular units. The nearest neighbor in the task map to the hand is computed with the approximate nearest neighbor algorithm described in [2]. For this, the hand position and orientation x_t^{rel} is represented in the reference frame of the object. The gradient is

$$\frac{\partial g_p}{\partial x_t^{\text{rel}}} = 2(x_t^{\text{rel}} - x_{\text{map}})^T W_{\text{map}}. \tag{1.19}$$

To account for the quality of the grasp that has been determined with the force closure measure, each sample of the task map is interpreted as a Gaussian:

$$f_i = |2\pi\Sigma_i^{-1}|^{-\frac{1}{2}} \exp(-\frac{1}{2} d_{\Sigma,i}), \tag{1.20}$$

with a mean vector μ and some small neighborhood $d_{\Sigma,i} = (x_i - \mu)^T \Sigma_i (x_i - \mu)$, determined by covariance matrix Σ^{-1}. The overall cost is computed as a weighted mixture of Gaussians considering the quality measure (force closure) w_i associated with each sample:

$$g_q = \frac{1}{\sum |f_i|} \sum w_i f_i. \tag{1.21}$$

We skip the gradient for brevity. This model has been chosen to account for the noise associated with the chosen force closure value. The noise is mainly due to the discontinuous number of contacts, the determination of the contact points and some other aspects. The mixture model smooths the criterion in a local region, assuring a smooth gradient.

While the proximity criterion tries to pull the final hand posture towards the task map manifold, the quality criterion becomes active only when the final hand posture is close to the manifold. It tries to level the target pose towards the best force closure. Figure 1.16 (b) shows the two cost terms over the time course of a trajectory. The proximity cost decreases as the final hand posture converges to the task manifold; the quality cost shows activity only when the hand posture is close to the manifold. To account for the formulated criteria during optimization, their gradient has to be derived with respect to the state vector according to term g' in Equation 10 of Table 1.1. For this, we can rewrite the differential kinematics as

$$\frac{\partial g_{\text{map}}}{\partial q_t} = \frac{\partial (g_p + g_q)}{\partial x_t^{\text{rel}}} \frac{\partial x_t^{\text{rel}}}{\partial q_t} = \frac{\partial (g_p + g_q)}{\partial x_t^{\text{rel}}} J_{\text{rel}}, \tag{1.22}$$

with J_{rel} being the Jacobian of the task vector relating to the relative hand–object coordinates of the task map. This means that the task map can be represented in different coordinates than we chose to control the system. We can, for instance, represent the task map in relative hand–object coordinates and control the movement in global coordinates.

Both quality and proximity terms are only evaluated in the last time step of the trajectory, since this corresponds to the hand's final grasp pose. Their effect is back-propagated in time and on the attractor point locations with Equations 11 ff of Table 1.1. The nearest neighbor query needs to be carried only out once in each optimization iteration, which is advantageous in terms of computational efficiency.

1.6.3 Experiments

We have set up an experiment comprising a model of the humanoid robot ASIMO, a table and a basket with a U-shaped handle, see Figure 1.15 (b). To account for hand–object proximities, we consider collisions for the hand and finger segments, the lower arms, the body, the thighs and the environment objects table and basket. The overall collision model comprises 30 body pairs. The following task variables are subject to the optimization: right hand position and orientation (2D, polar angles) between hand and basket. Further, we constrained the horizontal components of the center of gravity and the foot transformations. Another constraint has been added to the gaze direction vector: it will continuously travel towards the center of the top basket handle. The employed task map has been sampled with 1000 valid grasps. The best initialization of the trajectory is determined according to Section 1.6.2.

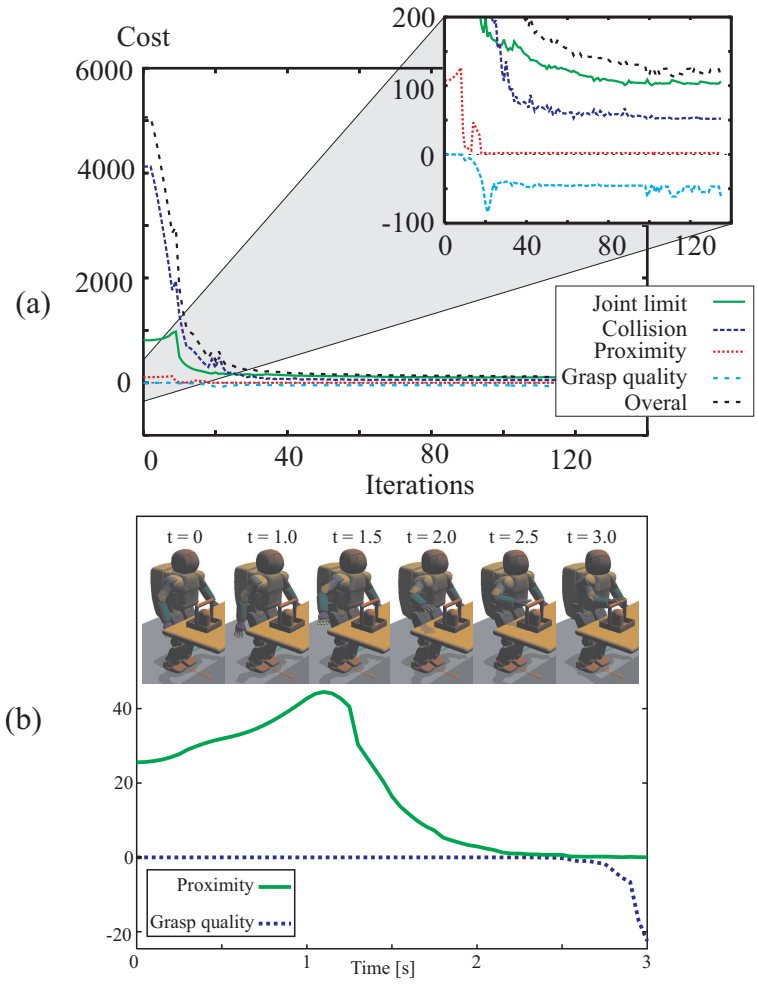

Figure 1.16 (a) Cost terms during optimization run; and (b) cost terms during movement

The progression of the cost terms during the simulation run is depicted in Figure 1.16 (a). The costs massively decrease in the beginning, since the initial trajectory leads to many collisions, violates some joint limits and doesn't exactly reach the task map manifold. After a few iterations, it rapidly converges to a minimum. The enlarged subimage in the figure shows that (1) the target task manifold is exactly reached after approximately 15 iterations and (2) the grasp quality converges after approximately 25 iterations.

Figure 1.16 (b) shows the proximity and quality cost over the optimal trajectory. It can be seen that initially, the hand moves away from the table edge, and then moves upward in an arc around the table. The finally chosen grasp position is on the

Figure 1.17 Movements for different basket locations

left side of the basket handle, which seems to lead to lower costs than grasping the handle at its center.

Figure 1.17 shows three representative movement sequences. In the upper row of the figure, the robot reaches and grasps for the inner handle of the basket. During reaching, it avoids to hit the other handles and finally shifts its hand to the lower part of the handle before closing the fingers. In the middle row, the basket has been moved away in the frontal direction. The optimal pregrasp was found on the right end of the basket's top handle. In the lower row, the basket was moved to the left side. In this location, the movement optimization was initialized with the right handle. In this case, the hand did not have to move under the top handle, which resulted in a more "relaxed" movement.

1.7 Conclusion

In this work we presented elements towards a consistent control, prediction and movement planning architecture for humanoid robots. Movement control is achieved

with a classical redundant control concept that has been extended with mechanisms to ensure balance stability and self-collision avoidance. This concept is the common basis of the subsequently presented methods that aim towards more movement intelligence. We presented a potential-field-based method to determine optimal stance poses for arbitrary end-effector tasks. It is an important element in the presented prediction and action selection architecture. This architecture predicts the outcome of a set of movement alternatives that solve a given task in different ways. The novel aspect is to continuously predict and evaluate the movement of a set of virtual controllers in parallel, these being able to quickly reorganize the movement behavior based on perceptual information. In order to cope with movements in more complex environments, a holistic trajectory optimization approach was presented. It operates on a somewhat slower time scale, but is still suited to be applied in interactive scenarios. Comparing this to the prediction architecture, it computes movements that satisfy criteria concerning the overall movement throughout a trajectory, such being able to reach towards objects while avoiding collisions and self-limits of the robot. This scheme has been extended to the coupled problem of reaching and grasping with the concept of object-specific *task maps*. These maps represent a functional characterization of objects in terms of their grasp affordances, i.e. a manifold of feasible pregrasp poses.

All elements have been verified in simulations and experiments with the humanoid robot ASIMO, and have been successfully integrated into large-scale systems such as [7, 8, 23].

Acknowledgments The authors thank A. Bendig, B. Bolder, M. Dunn, H. Janssen, N. Jetchev, J. Schmuedderich and H. Sugiura for their valuable contributions. Further, the authors thank the members of Honda's robot research teams in Japan for their support.

References

1. Abdel-Malek K, Mi Z, Yang J, Nebel K (2006) Optimization-based trajectory planning of the human upper body. Robotica 24(6):683–696
2. Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1998) An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J ACM 45(6):891–923
3. Baerlocher P, Boulic R (1998) Task-priority formulations for the kinematic control of highly redundant articulated structures. In Proceedings of the IEEE international conference on intelligent robots and systems (IROS), Canada
4. Berenson D, Diankov R, Nishiwaki K, Kagami S, Kuffner J (2007) Grasp planning in complex scenes. In Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots (humanoids), USA
5. Bergener T, Bruckhoff C, Dahm P, Janssen H, Joublin F, Menzner R, Steinhage A, von Seelen W (1999) Complex behavior by means of dynamical systems for an anthropomorphic robot. Neural Netw
6. Bizzi E, d'Avella A, P. Saltiel P, Tresch M (2002) Modular organization of spinal motors systems. Neuroscientist 8:437–442

7. Bolder B, Brandl H, Heracles M, Janssen H, Mikhailova I, Schmdderich J, Goerick C (2008) Expectation-driven autonomous learning and interaction system. In Proceedings of the IEEE-RAS conference on humanoid robots (humanoids), Korea
8. Bolder B, Dunn M, Gienger M, Janssen H, Sugiura H, Goerick C (2007) Visually guided whole body interaction. In Proceedings of the IEEE international conference on robotics and automation (ICRA), Italy
9. Buschmann T, Lohmeier S, Ulbrich H, Pfeiffer F (2005) Optimization based gait pattern generation for a biped robot. In Proceedings of the IEEE-RAS conference on humanoid robots (humanoids), USA
10. Buss SR. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Unpublished survey. http://math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/index.html
11. Chaumette F, Marchand E (2001) A redundancy-based iterative approach for avoiding joint limits: application to visual servoing. IEEE Trans Robotics Autom, 17(5):52–87
12. Chestnutt J, Nishiwaki K, Kuffner J, Kagami S (2007) An adaptive action model for legged navigation planning. In Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots (humanoids), USA
13. Chiaverini S (1997) Singularity-robust task priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Trans Robotics Autom, 13(3)
14. Choi SI, Kim BK (2000) Obstace avoidance control for redundant manipulators using collidability measure. Robotica 18:143–151
15. Colby CL (1998) Action-oriented spatial reference frames in cortex. Neuron 20(1):15–24
16. Dariush B, Gienger M, Arumbakkam A, Goerick C, Zhu Y, Fujimura K (2008) Online and markerless motion retargetting with kinematic constraints. In Proceedings of the IEEE international conference on intelligent robots and systems, France
17. English JD, Maciejewski AA (2000) On the implementation of velocity control for kinematically redundant manipulators. IEEE Trans Sys Man Cybern:233–237
18. Gienger M, Janssen H, Goerick C (2005) Task-oriented whole body motion for humanoid robots. In Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots (humanoids), USA
19. Gienger M, Janssen H, Goerick C (2006) Exploiting task intervals for whole body robot control. In Proceedings of the IEEE/RSJ international conference on intelligent robot and systems (IROS), China
20. Gienger M, Toussaint M, Goerick C (2008) Task maps in humanoid robot manipulation. In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), France
21. Gienger M, Toussaint M, Jetchev N, Bendig A, Goerick C (2008) Optimization of fluent approach and grasp motions. In Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots (humanoids), Korea
22. Gienger M, Bolder B, Dunn M, Sugiura H, Janssen H, Goerick C (2007) Predictive behavior generation - a sensor-based walking and reaching architecture for humanoid robots. Informatik Aktuell (AMS): 275–281, Germany, Springer (eds. Berns K, Luksch T)
23. Goerick C, Bolder B, Janssen H, Gienger M, Sugiura H, Dunn M, Mikhailova I, Rodemann T, Wersing H, Kirstein S (2007) Towards incremental hierarchical behavior generation for humanoids. In Proceedings of the IEEE-RAS international conference on humanoid robots (humanoids), USA
24. Guan Y, Yokoi K (2006) Reachable boundary of a humanoid robot with two feet fixed on the ground. In Proceedings of the international conference on robotics and automation (ICRA), USA, pp 1518 – 1523
25. Guan Y, Yokoi K, Xianmin Zhang X (2008) Numerical methods for reachable space generation of humanoid robots. Int J Robotics Res 27(8):935–950
26. Guilamo L, Kuffner J, Nishiwaki K, Kagami S (2005) Efficient prioritized inverse kinematic solutions for redundant manipulators. In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Canada

27. Harada K, Kajita S, Kaneko K, Hirukawa H (2004) An analytical method on real-time gait planning for a humanoid robot. In Proceedings of the IEEE-RAS international conference on humanoid robots (humanoids), USA
28. Haschke R, Steil JJ, Steuwer I, Ritter H (2005) Task-oriented quality measures for dextrous grasping. In Proceedings of the IEEE international symposium on computational intelligence in robotics and automation (CIRA), Finland
29. Haschke R, Weitnauer E, Ritter H (2008) On-line planning of time-optimal, jerk-limited trajectories. In Proceedings of the IEEE-RSJ international conference on intelligent robots and systems (IROS), France
30. Heim A, van Stryk O (1999) Trajectory optimization of industrial robots with application to computer-aided robotics and robot controllers. Optimization 47:407–420
31. Hirose M, Haikawa Y, Takenaka T, Hirai K (2001) Development of humanoid robot asimo. In Workshop of the IEEE/RSJ international conference on intelligent robots and systems (IROS), USA
32. Ijspeert A, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In Proceedings of the IEEE international conference on robotics and automation (ICRA), USA
33. Ijspeert A, Nakanishi J, Schaal S (2003) Learning attractor landscapes for learning motor primitives. Advances in Neural Information Processing Systems 15. MIT Press, Cambridge MA, USA, pp 1523–1530
34. Kajita S, Kanehiro F, Kaneko K, Fujiwara K, Harada K, Yokoi K, Hirukawa H (2003) Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), USA
35. Kavraki L, Svestka P, Latombe J, Overmars M (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robotics Automa (12):566–580
36. Khatib O (1987) A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE Int J Robotics Autom, RA-3(1):43–53
37. Kuffner J, LaValle S (2000) RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE international conference of robotics and automation (ICRA), USA
38. Latombe JC (1991) Robot motion planning. Kluwer, Boston, MA, USA
39. Laumond JP (1998) Robot motion planning and control. Springer-Verlag, Berlin, Germany. Available online at http://www.laas.fr/∼jpl/book.html
40. LaValle S (2006) Planning algorithms. Cambridge University Press, Cambridge, UK. Available at http://planning.cs.uiuc.edu
41. Lebedev D, Klanke S, Haschke R, Steil J, Ritter H (2006) Dynamic path planning for a 7-dof robot arm. In Proceedings of the IEEE-RSJ international conference on intelligent robots and systems (IROS), China
42. Liégeois A (1977) Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Trans Sys Man Cybern, SMC-7 (12)
43. Maciejewski AA (1990) Dealing with the ill-conditioned equations of motion for articulated figures. IEEE Computat Graphics Applic, 10(3):63–71
44. Marchand E, Chaumette F, Rizzo A (1996) Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Japan
45. Michel P, Chestnutt J, Kuffner J, Kanade T (2005) Vision-guided humanoid footstep planning for dynamic environments. In Proceedings of the IEEE-RAS conference on humanoid robots (humanoids), USA
46. Miller AT, Knoop S, Christensen HI, Allen PK (2003) Automatic grasp planning using shape primitives. In Proceedings of the IEEE international conference of robotics and automation (ICRA), Taiwan
47. Murray RM, Li Z, Sastry SS (1994) Robotic manipulation. CRC Press, FL, USA
48. Mussa-Ivaldi FA, Giszter SF, Bizzi E (1994) Linear combinations of primitives in vertebrate motor control. Neurobiology 91:7534–7538

49. Nakamura Y (1991) Advanced robotics: redundancy and optimization. Addison-Wesley
50. Nakamura Y, Hanafusa H (1987) Optimal redundancy control of robot manipulators. Int J Robotics Res (6)
51. Pfeiffer F (2008) Mechanical system dynamics. Springer Verlag
52. Platt R, Fagg AH, Grupen R (2006) Improving grasp skills using schema structured learning. In Proceedings of the international conference on development and learning (ICDL), India
53. Mas R, Boulic R, Thalmann D (1997) Interactive identification of the center of mass reachable space for an articulated manipulator. In Proceedings of the IEEE international conference of advanced robotics (ICAR), USA
54. Schaal S (2003) Movement planning and imitation by shaping nonlinear attractors. In Proceedings of the 12th Yale workshop on adaptive and learning systems, Yale University, USA
55. Schaal S, Peters J, Nakanishi J, Ijspeert AJ (2003) Control, planning, learning, and imitation with dynamic movement primitives. Workshop on bilateral paradigms on humans and humanoids, IEEE international conference on intelligent robots and systems (IROS), USA
56. Schlemmer K, Gruebel G (1998) Real-time collision-free trajectory optimization of robot manipulators via semi-infinite parameter optimization. Int J Robotics Res, 17(9):1013–1021
57. Schwammkrug D, Walter J, Ritter H (1999) Rapid learning of robot grasping positions. In Proceedings of the international symposion on intelligent robotics systems (SIRS), Portugal
58. Steil JJ, Roethling F, Haschke R, Ritter H (2004) Situated robot learning for multi-modal instruction and imitation of grasping. Robotics Autonomous Sys 47 (Special issue on robot learning by demonstration):129–141
59. Suárez R, Roa M, Cornellà J (2006) Grasp quality measures. Technical Report IOC-DT-P 2006-10, Universitat Politècnica de Catalunya, Institut d'Organització i Control de Sistemes Industrials
60. Sugihara T, Nakamura Y (2002) Whole-body cooperative balancing of humanoid robot using COG jacobian. In Proceedings of the IEEE/RSJ international conference on intelligent robot and systems (IROS), Switzerland
61. Sugiura H, Gienger M, Janssen H, Goerick C (2007) Real-time collision avoidance with whole body motion control for humanoid robots. In Proceedings of the IEEE-RSJ international conference on intelligent robots and systems (IROS), USA
62. Toussaint M, Gienger M, Goerick C (2007) Optimization of sequential attractor-based movement for compact movement representation. In Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots (humanoids), USA
63. Yoshida E, Belousov I, Esteves C, Laumond JP (2005) Humanoid motion planning for dynamic tasks. In Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots, USA
64. Zacharias F, Borst C, Beetz M, Hirzinger G (2008) Positioning mobile manipulators to perform constrained linear trajectories. In Proceedings of the IEEE-RSJ international conference on intelligent robots and systems (IROS), France