Task Level Motion Planning – Integrating Local Robot Control and Global Sampling

Matthias Behnisch

Diplom Informatiker Matthias Behnisch
AG Neuroinformatik
Research Institute for Cognition and Robotics (CoR-Lab)
Technische Fakultät, Universität Bielefeld
email: mbehnisc@techfak.uni-bielefeld.de

Abdruck der genehmigten Dissertation zur Erlangung
des akademischen Grades Doktor der Ingenieurwissenschaften.
Der Technischen Fakultät der Universität Bielefeld
am 20.06.2012 vorgelegt von Matthias Behnisch,
am 10.09.2012 verteidigt und genehmigt.

Gutachter:
Prof. Dr. Helge Ritter (Universität Bielefeld)
Prof. Dr. Oliver Brock (Technische Universität Berlin)
Dr.-Ing. Michael Gienger (Honda Reserach Institute Europe, Offenbach/Main)

Prüfungskommission:
Prof. Dr. Ralf Möller (Universität Bielefeld)
Prof. Dr. Helge Ritter (Universität Bielefeld)
Prof. Dr. Oliver Brock (Technische Universität Berlin)
Dr.-Ing. Michael Gienger (Honda Reserach Institute Europe, Offenbach/Main)
Dr. Felix Reinhart (Universität Bielefeld)

Gedruckt auf alterungsbeständigem Papier nach ISO 9706.

Task Level Motion Planning – Integrating Local Robot Control and Global Sampling

Der Technischen Fakultät der Universität Bielefeld vorgelegt von

Matthias Behnisch

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften.

Juni 2012

Abstract

This thesis addresses the problem of generating goal-directed movements for robots, a challenging problem especially in cases where the involved state spaces are high-dimensional and complex constraints, such as obstacles and joint-limits, are present. Local methods can be severely hindered in such situations and planning on a global level becomes necessary. Although an exhaustive search of the state space can generally solve these problems, complete planning methods do not scale well with the number of dimensions. To avoid the inherent complexity of complete state space planning, this work proposes the use of a condensed task-centered representation, focusing on the task to be achieved and neglecting all unnecessary details. The motion planning problem can then be effectively decomposed into two layers: A global search component that is restricted to the coarse-scale task representation and a local control method that generates fine-scale motions.

The ability of the approach to solve difficult planning problems is demonstrated with an application for redundant robotic manipulators. Due to the limitation of global planning to the task representation, completeness can no longer be guaranteed, but it can be shown that many relevant situations can be solved. In particular, the ability to solve planning tasks for a humanoid robot operating in human-centered environments is demonstrated in both simulation and laboratory experiments, providing evidence of a significant computational advantage compared to complete state space sampling methods.

Acknowledgments

A big thank you to Robert Haschke, Michael Gienger and Helge Ritter for supervising this work. Also to my friends and colleagues from the Neuroinformatics Group and from the CoR-Lab Graduate School. The lab crew was very helpful assisting with the robot experiments, thank you very much. To all my friends, cheering me up and reminding me of the bright side. This would not have been possible without the support of my family, thank you all for being so encouraging and helpful, you are awesome. Last but not least, I thank Theda for her passionate support and for being there for me with her love.

I gratefully acknowledge the financial support from Honda Research Institute Europe.

Contents

Chapter 1

Introduction

The world continues to be computerized at a fast pace. Driven by the integrating power of the globe-circling Internet, information technology enters our everyday lives and shapes our culture. Electronic entertainment, education and communication. To an even greater extend, our machines evolve, crammed with micro controllers and sensors, recording on all channels with high bandwidth. Smart automobiles, intelligent domestic appliances and interactive toys. On the frontier of this process, there are robots emerging in application areas outside of academia. Medical robots are assisting during operations, cars are able to park by themselves and unmanned drones replace civil and military planes. Factory automation and automated warehouse logistics are reaching ever higher levels, producing and distributing our global streams of goods.

Among many, some key scientific and technological advancements can be identified that drive this evolution. On one hand, a wide spectrum of powerful sensors becomes available and there is progress in integrating many of them for robust and accurate information gathering and perception. On the other hand, steps are taken that go beyond a hard-wired behavior and a blind following of pre-determined programs. Benefiting from sensory enhancements, it becomes possible to dynamically adapt behaviors and to react to unforeseen events. This work aims to utilize these capabilities to take a step towards more deliberative action generation. While small scale adaptive behavior can perform reactive mircomanagment, an enlarged perceptive horizon can be utilized to concentrate on the achievement of large scale objectives. The immediate course of actions can be reactively generated, leaving resources to reason about future states and to plan ahead in time.

The level of large scale objective addressed in this work is the goal-directed planning of non-trivial movements for redundant robotic manipulators. A large scale objective could for instance be to reach for a certain object in a obstructed and cluttered environment. While deliberative planning guides and monitors the progress towards this goal, small scale motion coordination handles the individual motion of the robots limbs based on sensory inputs. This devision of labor enables to tackle complex problems, as shown in an application of motion planning for highly redundant humanoid robots operating in human environments.

It is important to keep in mind that goal-directed planning as discussed here is not the ultimate level of planning, there exist higher levels not subject of this work. It is assumed that a goal is already given, the intention is already there. Determination of this goal is a different planning problem.

1.1 Integrating Control and Planning

The creation of smooth goal-directed movements is a central research field in robotics. In the area of *motion control*, various tools for a systematic development of control strategies can be found. They allow for an efficient solution for the problem of controlling a potentially very high number of variables in order to follow a certain defined motion behavior. Relying on explicit modeling of the involved systems, they can handle complex relationships while

being fast and accurate. If sensory information is incorporated, it is also possible to adapt the behavior to cope with varying conditions. However, since motion control heavily relies on explicit models of the application domain, it often lacks the flexibility to achieve more complex behaviors. For these cases, it might be possible to develop problem specific solutions, hand tuned for a certain objective, but this can quickly become a tedious task and is not universally applicable.

The ability to create any kind of movement is addressed in the field of *motion planning* in a general way. Typically, the problem is handled on a more abstract level, which allows for the development of methods that are independent of specific system models. Although some astonishing results in solving complex problems can be achieved, these methods are not very efficient and have difficulties in handling the requirements of real world applications. This is the case because task specific knowledge is neglected and a huge number of possible solutions is considered, even those irrelevant to the task at hand.

Combining both, the small scale motion control and the large scale motion planning approach, helps in overcoming these limitations. By augmenting control methods with motion planning capabilities, solving more complex problems in a general way becomes possible, without the need to develop specific solutions. Conversely, incorporating suitable control approaches into motion planning effectively narrows the huge amount of possibilities down to the most relevant portion for a given task.

This concept is implemented in our proposed *hybrid motion planning* framework. The gap between motion control and motion planning is bridged, combining the strengths of both approaches and overcoming their weaknesses. A key element is the shift of planning towards a high-level *task representation*, on which sub-goals can be specified on a coarse, task-specific scale, while actual execution of these task-level sub-goals is delegated to suitable motion control techniques.

Several questions and challenges have to be addressed towards a successful application of hybrid motion planning. One issue is to adapt motion planning towards searching the task representation. Can already known methods be re-used in this different context? What is the most efficient way to search on task level? Other questions are arising from the involvement of motion control. How autonomous in creating movements should the control method be and is it possible to effectively utilize this? Can additional constraints arising from motion control be handled? Further, with respect to the application to planning for redundant robots, can complex problems indeed be solved and is the effort to do so reduced? Is it possible to address real world planning problems? Also, are there limits on the problems that can be solved?

1.2 Outline

Since the project is located at the intersection of motion control and motion planning, these two areas are separately introduced first. Chapter 2 reviews the widely used resolved motion control method as a way to generate movements towards a given task target. Extensions for singularity robust control and redundancy resolution are presented and with the *whole body motion control* framework, an implementation of these methods for humanoid robots is described. Building on resolved motion control, a novel control method is developed in chapter 3. The strict task hierarchy of primary and secondary tasks, characteristic for the methods of chapter 2, is broken. Secondary tasks are allowed to influence the primary task and the influence can be dynamically varied. An approach to automatically adapt the secondary tasks influence to ensure primary task convergence is presented.

An introduction into the area of motion planning is given in chapter 4. Three basic

approaches are summarized, local techniques based on potential fields and optimization, globally valid combinatorial and sampling-based methods. In particular, probabilistically complete random tree search methods are described and their specific properties and challenges are highlighted.

Chapter 5 then develops the proposed concept of *hybrid motion planning*. The previously described areas of motion control and motion planning are joined in a hybrid motion planning approach. The conceptual division into a local planning and a global planning layer is formalized and conceptual properties are analyzed. The interplay of both layers is examined with the goal of devising good global exploration strategies for redundant robots.

An implementation of hybrid motion planning for redundant robots is presented in chapter 6, combining sampling-based global planning with local motion control, able to utilize redundancies to avoid joint-limits and obstacle constraints. A simulated humanoid robot is used as a test system for a planning study, comparing our novel approach against state of the art motion planning methods. In chapter 7, the feasibility of hybrid planning for real-world scenarios is demonstrated, using a humanoid robot performing a complex bi-manual object placement task. So far, the optimization abilities of local planning were limited to the redundant space. Chapter 8 integrates the relaxed motion control method developed in chapter 3 into the framework, allowing to modify task trajectories as well in order to enhance the constraint avoidance capabilities of local planning. A novel sampling heuristic exploiting the enhanced local planning abilities is developed and simulation results for a redundant planar manipulator are presented.

Finally, a conclusion and outlook is given in chapter 9.

Chapter 2

Motion Control

This chapter reviews methods for generating movements for robotic manipulators. Starting with a discussion of the question how tasks can be represented in section 2.1, the basic resolved motion rate control technique is introduced in section 2.2 and singularity robust motion rate control in section 2.3. If the kinematic structure of a manipulator is redundant, there remains some freedom in choosing the joint motions which can be utilized for secondary motion objectives, as explained in section 2.4. While the first parts of this chapter deal with general manipulators, a specific control framework for humanoid robots is described in section 2.5. Finally, section 2.6 examines constraints of the control system with the aim of characterizing the behavior of the resulting movements.

2.1 Task Representation

Different robots can move on many different ways. They have different actuation possibilities and different kinematic structures, normally selected to best fit their primary work objectives. Looking at different applications, one can differentiate several distinct areas with distinct movement characteristics. Some objectives exculsively address point-to-point movements, for example to reach for a certain object and to carry it to some fixed destination. Other types of movements need to perform an exact tracing of a pre-determined geometric path, like an industrial welding robot that has to precisely follow a weld seam. In contrast, other path following movements allow for a greater freedom in execution, for instance wiping movements for cleaning a surface or human like gestures. Another class of movements has an almost opposite objective, not to change the spatial position of the robot or a tool but to stabilize it on a fixed position. This is the case for balancing tasks, for example moving the hip and the legs of a humanoid robot in a way that keeps it in an upright position.

Once the type of movement needed is specified, the question arises how to encode the actual movement to be done, how to represent a certain *task*. Consider a mobile robot that is able to move in the plane. Its full state space consists of three variables, the two dimensional position on the plane and the orientation around its vertical axis. A possible objective of the robot could be to patrol between a number of waypoints, defined as two dimensional positions on the plane. The orientation at this waypoints remains undefined, since one is only concerned about the robot being at a certain place, with any orientation. The *task space* in which the waypoints are specified is thus a two dimensional sub-space of the three dimensional state space. Regarding the movement objective of the robot, only the position in the task space matters.

Transfered to robotic manipulators, similar observations can be made. Usually, the goal of a movement is described as moving one point of the structure, often the end-effector, from one position to another. Such a task is not defined by specifying complete postures of the manipulator in its state space, but is described in a different task dependent representation, capturing the movement goal more directly. For example, the task of sliding an object on a table surface between two positions is more easily described by planar coordinates on the

table than by full arm postures, involving a potentially large number of joints. The task can be fully encoded in planar table surface space, which is of a lower dimension than the full state space and for which a mapping between task positions and states of the robot exists.

The existence of such task centered representations can be confirmed in humans as well. Scholz and Schöner (1999) show that motion trajectories of a human performing a sit-to-stand movement can be decomposed into two orthogonal subspaces. While motions in one subspace affect the controlled variables, motions in the other subspace have no influence. For the sit-to-stand task, the position of the center of mass in the sagittal plane is controlled, while head and hand positions are controlled less stable, they are part of the *uncontrolled manifold*. Consequently, the motion objective can be described with a task space representation given by the center of mass position, while all other degrees of freedom are not of interest, they are redundant to the task at hand.

These task space representations can be found for other types of human movements as well using *machine learning* techniques. For instance, Maycock et al. (2010) examine kinematic data of humans grasping for spherical objects. Analyzing the data with principal component analysis reveal that a large amount of the postural variance can be described with few principal components. These components can be seen as a condensed representation of this specific grasping task. In (Steffen et al., 2008), human manipulation data is used to explicitly build a representation of the involved movements. It is possible to construct low dimensional manifolds that contain the hand postures specific for a certain task, like the unscrewing of a bottle. With a suitable structuring of these manifolds, meaningful motion parameters can be captured along distinct dimensions, like the size of a bottle cap in one dimension and the cap turning motion along another dimension. Calinon et al. (2007) use kinesthetic teaching to record a desired movement for a humanoid robot. By computing statistics over the data variance along different dimensions, a probabilistic representation of the movement can be constructed on a low dimensional latent space.

Thus, the first step in finding the right task representation is to identify in which space the task should be encoded. In this work, the idea of decomposing the state space into a task specific and a task independent subspace or manifold is used, but unlike the previously presented machine learning methods, the task space is explicitly modeled for the task at hand. Sections 2.2 and 2.3 introduce the *resolved motion rate control* method as a well known and versatile approach for motion control of robotic manipulators. Motions can be executed fast and accurate due to a precise modeling of the kinematics. In addition, redundancies can be effectively utilized to perform additional motion objectives with the *gradient projection method*, as shown in section 2.4.

The second step is then to encode the actual movement to be done, using the previously found task representation. A movement can be defined as a geometrical path, for example a pair of a starting and a target point with a finite sequence of points in-between. Often a smooth movement between these points is desired, for which a time law over the path is specified, a rule that assigns a velocity and acceleration to each spatial point. Techniques to produce such smooth trajectories with polynomial interpolation are for example given by Sciavicco and Siciliano (2000, chapter 5).

This approach is feasible if the objective is primarily about following pre-defined paths, but often it is desired to generalize tasks away from specific paths towards whole sets of paths. Many tasks can be solved by following different paths that equivalently solve the problem. For example, the only condition for the task of reaching for a fixed object is that the path ends at the object. The shape of the path before the object is reached is not restricted. A suitable representation allowing this generalization are dynamical systems, as demonstrated by works of Schöner et al. (1995) or with the *dynamic movement primitives* (DMP) (Ijspeert et al., 2002; Schaal, 2006). An advantage of dynamical systems is the ability

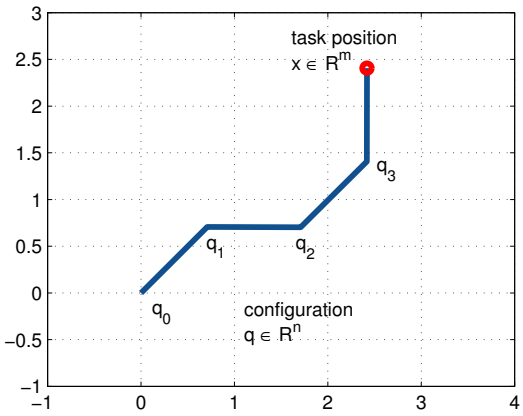

Figure 2.1: Simple example of a robot manipulator with 4 rotational joints q_0 to q_3. The robot is fixed at the base $(0, 0)$ and the end-effector is marked by the red circle. Task space is the 2 dimensional end-effector position.

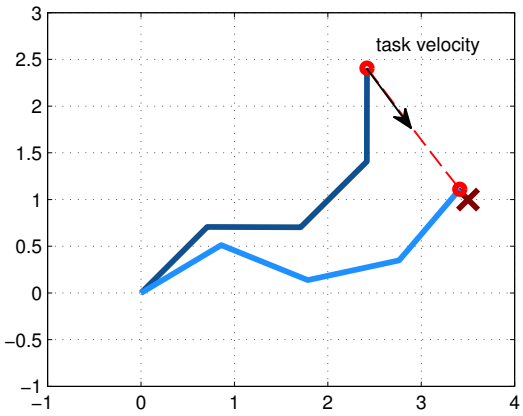

Figure 2.2: To reach a task space target (red cross), suitable task velocities have to be specified. Based on the velocities, motion rate control is used to compute a matching motion of the joints.

to maintain stable behaviors over a large space of possibilities. For example, if the dynamics follow the slope of a potential well with a single global extremum, the convergence can be guaranteed under all circumstances. Another strength is the flexibility to include multiple modalities, like vision and position, and multiple tasks, like simultaneous target reaching and obstacle avoidance, as shown in chapter 3.

In this work, the dynamical system concept is followed for the representation of desired movements. More specifically, the second order attractor dynamic approach introduced by Gienger et al. (2005) is used and detailed description is given in section 2.5.1.

2.2 Jacobian Pseudoinverse Motion Rate Control

Figure 2.1 shows a schematic model of planar manipulator. While the posture of the manipulator is given by a vector of joint-angles $q \in \mathbb{R}^n$, the task is defined by a second vector $x \in \mathbb{R}^m$, representing for example the tool or end-effector position. Note that the space where the task is defined, the task-space, does not have to coincide with the workspace of the manipulator, here the two dimensional plane. For instance, the task space could encompass three dimensions given by the tool position and orientation. The connection between joint-angles and task representation is given by the kinematic mapping of a manipulator, determining where a point of interest on the robot is located, given an exact specification of the posture. Formally, this mapping can be expressed by a nonlinear mapping $x = f(q)$.

A fundamental problem for the control of a manipulator is the computation of the right joint angles in order to reach a desired position of the end-effector. This problem is called inverse kinematics and can be realized by inverting the kinematic equation. Although there exist algebraic solutions for some simpler structures there is no general closed form solution. For general robotic manipulators, a common approach is to restrain to a linearization of the equation and to use the differential relationship between the joint velocities and end-effector velocities. As shown in figure 2.2, the problem is then to compute a suitable instantaneous joint motion for every time step, given a desired task velocity. With this method, called resolved motion rate control, it is then possible to solve the inverse kinematics problem.

Following Nakamura (1991), differentiation of the kinematic mapping with respect to time

yields

$$\dot{x} = J(q)\dot{q} \tag{2.1}$$

with the Jacobian matrix $J(q) \in \mathbb{R}^{m \times n}$ relating joint velocities $\dot{q} \in \mathbb{R}^n$ to the resulting end-effector velocities $\dot{x} \in \mathbb{R}^m$. This linear equation can be inverted to solve for the joint velocities given desired end-effector velocities, under the assumption that the Jacobian can be inverted. If $m = n$, the Jacobian is square and is invertible if it has full rank. If $m < n$, the system is redundant, meaning that there are more degrees of freedom than end-effector variables, leading to an under determined equation system and thus potentially to an infinite variety of solutions. In this case, the solution for the inverse Jacobian can be obtained using the generalized Jacobian pseudoinverse $J^{\#}(q) \in \mathbb{R}^{n \times m}$:

$$\dot{q} = J^{\#}(q)\dot{x} \tag{2.2}$$

The pseudoinverse can be seen as the least-squares solution of equation 2.1, minimizing the norm $\|\dot{x} - J(q)\,\dot{q}\|$ and choosing the smallest possible $\|\dot{q}\|$ among all possible solutions. It can be computed as follows:[1]

$$J^{\#} = J^T(JJ^T)^{-1} \tag{2.3}$$

A numerically stable way of computing the pseudoinverse is to use the singular value decomposition (SVD), see e.g. (Nakamura, 1991).

2.3 Singularity Robust Motion Rate Control

A problem with resolved motion rate control is how to deal with singularities. Singularities are configurations of a manipulator where the mobility is reduced and can be classified into two cases (Sciavicco and Siciliano, 2000): Boundary singularities are at the boundary of the workspace when the manipulator is either outstretched or retracted. These singularities can be avoided since these regions are relatively easy to identify and the manipulator can be prevented to go there. The second class of singularities are internal singularities that can occur everywhere inside the workspace. They appear if degrees of freedom are aligned in an unfortunate way and are harder to counteract because they are not as easily located.

Regarding the Jacobian pseudoinverse approach, there are also computational problems caused by singularities (Nakamura, 1991): Although the pseudoinverse in equation 2.3 gives approximate solutions at singularities, the switch from an accurate solution to an approximated one causes discontinuities. Also the joint-rates become prohibitively large in a neighborhood around singularities. In order to compute continuous and feasible solutions at or in the neighborhood of singularities, a singularity robust inverse is proposed by Nakamura (1991) as one particular method from the family of damped least-squares approaches, see for example (Chiaverini et al., 2008). It minimizes an error norm $||e||$ composed of the exactness of the solution and the magnitude of the joint velocities

$$e = \begin{pmatrix} \dot{x} - J(q)\dot{q} \\ \dot{q} \end{pmatrix}, \tag{2.4}$$

in order to find a solution that is both accurate and feasible with small joint rates. The singularity robust pseudoinverse can be computed as:

$$J^* = J^T(JJ^T + kI)^{-1} \tag{2.5}$$

[1] If not explicitly stated, the Jacobian $J(q)$ is always dependent on q.

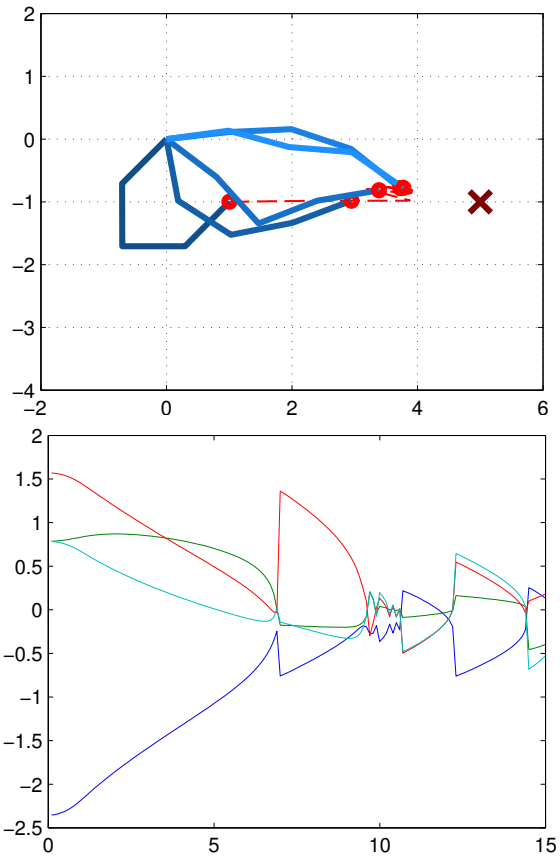

Figure 2.3: Example of a four joint manipulator approaching a singularity (top). Near the singularity, the joint velocities are discontinuous, leading to a discontinuous joint space trajectory as well (bottom).

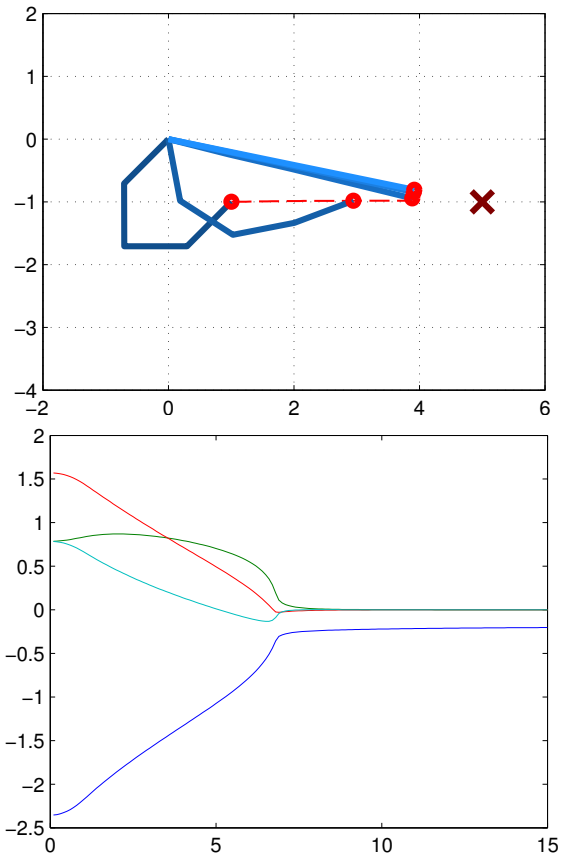

Figure 2.4: Example of a four joint manipulator approaching a singularity under singularity robust control (top). The joint velocities are damped, resulting in a smooth continuous joint space trajectory (bottom).

with I being the identity matrix. The parameter k weights between exactness (small) and feasibility (large) of the solution and can be chosen adaptively, based on the manipulability measure $w = \sqrt{\det(JJ^T)}$ as an estimate of the closeness to singularities:

$$k = \begin{cases} k_0(1 - \frac{w}{w_0})^2 & \text{for} \quad w < w_0 \\ 0 & \text{else} \end{cases} \tag{2.6}$$

If the manipulability w gets smaller than a threshold w_0, the weight k is increased until it reaches its maximum value of k_0 at $w = 0$.

Figure 2.3 shows an example how Jacobian pseudoinverse motion rate control behaves when approaching a singularity. In figure 2.4 the same movement is done with the singularity robust pseudoinverse control.

2.4 Redundancy Resolution

The gradient projection method (Liegeois, 1977) is one of the most widely adopted approaches to resolve redundancy. It is based on a more general solution of equation 2.2:

$$\dot{q} = J^{\#}(q)\dot{x} + (I - J^{\#}J)z \tag{2.7}$$

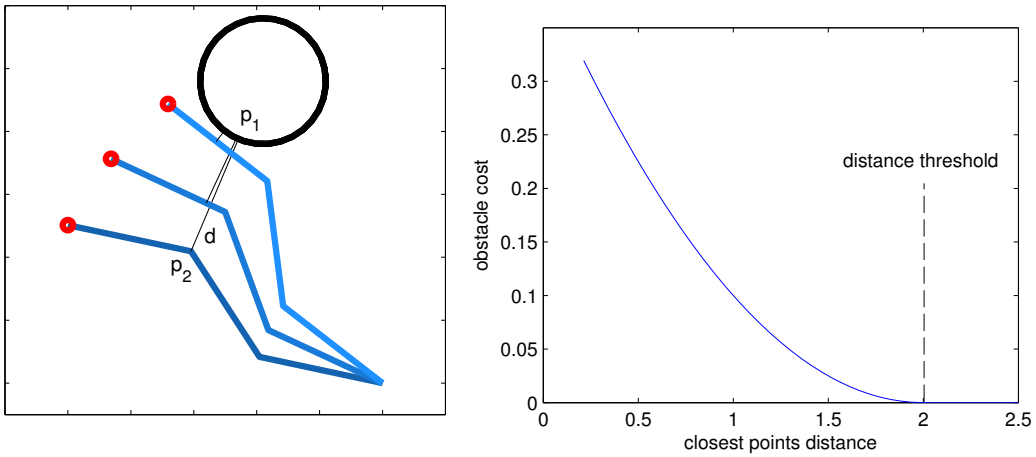

Figure 2.5: Computation of the obstacle avoidance cost function illustrated. The obstacle cost depends quadratically on the distance d of closest points p_1 and p_2 on the robot and obstacle. Only distances below the threshold d_B are considered.

The first term uniquely maps task-velocities $\dot{x} \in \mathbb{R}^m$ to joint-velocities $\dot{q} \in \mathbb{R}^n$, just as in equation 2.2, and the second term projects an arbitrary vector $z \in \mathbb{R}^n$ to the Jacobian nullspace. Of both projections, the nullspace $\mathcal{N}(J)$ and the range space $\mathcal{R}(J^{\#}) = \mathcal{N}^{\perp}(J)$, are orthogonal to each other. The vector z, projected on the nullspace, can only induce a manipulator self-motion without disturbing the task execution and by systematically varying z, all possible inversions of equation 2.1 are created.

This way, the choice of z controls how the manipulators redundancy is resolved. With the gradient projection method, the redundancy is used to optimize the motion with respect to a secondary task objective, given by a scalar cost function $H(q)$ in joint-space. To minimize this cost function, z is set proportional to the negative gradient of the cost function in every control step:

$$z = -\gamma \left(\frac{\partial H}{\partial q} \right)^T \tag{2.8}$$

The overall joint motion thus consists of a task component minimizing $||\dot{x} - J(q)\dot{q}||$ and a secondary motion objective, minimizing $||H(q)||$. Note that the minimization is done locally in every control step and does in general not result in a global cost minimization over the whole trajectory. In the following sections two possible secondary motion cost functions are described.

The singularity robust pseudoinverse $J^*(q)$ (equation 2.5) could be used in this scheme as well. However, the orthogonality of the task- and redundant-space projections can be violated in certain situations, meaning that the task and redundant space motions are no longer strictly separated. As a result, the joined minimization of the task- and secondary-motion objectives might fail once the redundant space motion influences the task motion or vice versa. In practice however, situations where the singularity robust pseudoinverse mapping is substantially different from the simple pseudoinverse solution are limited to a narrow region around singularities and the impact on the complete trajectories remains rather small.

2.4.1 Joint Limit Avoidance

A cost-function H_{jc} that avoids the mechanical limits of joints is developed in Liegeois (1977):

$$H_{jc}(q) = \frac{1}{2}\sum_{i=1}^{n}\left(\frac{q_i - q_{0,i}}{q_{max,i} - q_{min,i}}\right)^2 \tag{2.9}$$

The function penalizes the deviation of joint i from a given reference position $q_{0,i}$ with quadratic costs and the contribution of each joint to the overall cost is weighted by its motion range, given by $q_{min,i}$ and $q_{max,i}$.

2.4.2 Obstacle Avoidance

To prevent the manipulator from colliding with itself or with external obstacles, a cost function that is based on the distance of closest point pairs is defined in Gienger et al. (2010). The costs of an individual pair p_i, consisting of points $p_{1,i}$ and $p_{2,i}$, are given by a quadratic cost function g_{p_i}, based on the distance d_{p_i}

$$g_{p_i} = \begin{cases} s(d_{p_i} - d_B)^2 & 0 \leq d_{p_i} \leq d_B \\ 0 & d_{p_i} > d_B \end{cases} \tag{2.10}$$

Costs are only created below a threshold distance d_B and the parameter s determines the slope of the parabola at zero distance. The overall cost function H_{oc} incorporates all individual costs over the set of all closest point pairs $\{p_i = (p_{1,i}, p_{2,i}) \mid i = 1...P\}$:

$$H_{oc} = \sum_{i=1}^{P} g_{p_i} \tag{2.11}$$

This cost function is not yet defined in the joint-space. To do so, the distance gradient is expressed w.r.t. the closest points p_1 and p_2 and mapped into the joint space with the corresponding body point Jacobians J_{p_1} and J_{p_2}. In combination with the gradient of the distance based cost function, the joint-space gradient w.r.t. a single point pair p_i becomes

$$\frac{\partial g_{p_i}}{\partial q} = \frac{\partial g_{p_i}}{\partial d_p}\frac{\partial d_p}{\partial x}\frac{\partial x}{\partial q} = 2s(1 - d_B/d_p)(J_{p_2} - J_{p_1})^T(p_2 - p_1) \tag{2.12}$$

and summing over all individual gradients yields the overall cost-function gradient

$$\frac{\partial H_{oc}}{\partial q} = \sum_{i=1}^{P}\frac{\partial g_{p_i}}{\partial q} \tag{2.13}$$

2.4.3 Examples

How the pseudoinverse control approach behaves is shown in some simple example movements. The controlled robot is the previously used simple planer manipulator with four joints. The task space is the two-dimensional space of end-effector position, leaving the end-effector orientation unspecified. The task is represented as an attractor point and the second order dynamical system from (Gienger et al., 2010) (see section 2.5.1) is used to compute task velocities $\dot{x}$ for given current task space positions x and targets y. Joint-space velocities $\dot{q}$ are then computed according to the motion rate control law.

Different redundant space resolution strategies are compared. In figure 2.6 the plain pseudoinverse approach is used, according to equation 2.2.

Figure 2.7 and figure 2.8 illustrate redundancy resolution with the gradient projection method, equation 2.7. In figure 2.7, the joint limit avoidance cost function is used, given by equation 2.9. As visible in the cost function plot, the redundant space movement quickly converges from an initial high cost posture to a lower cost posture.

The obstacle avoidance cost function, equation 2.10, is shown in figure 2.8. Note that the cost function is not guaranteed to monotonically decrease, indeed the costs initially raise for this example.

2.5 Humanoid Whole Body Motion

All control methods described up to now are quite general and could be applied to any kind of redundant or non-redundant manipulator. There are many robot classes that are successfully controlled with these methods and this section introduces a control framework for one class of particular interest – humanoid robots. Humanoid robots are directly inspired by human capabilities, for an introduction and overview of existing systems refer to (Kemp et al., 2008). Central topics in this field are the realization of human like two legged locomotion and balancing, as well as two handed manipulation incorporating complex upper body structures and hands of different levels of dexterity. This two areas can also be combined for activities including walking and upper body manipulation. Other important aspects are human inspired perception and also social behavior and communication.

Humanoid robots have body structure morphological similar to humans, that is they have two arms, two legs, a torso and a head. The structure is more complex than a single sequential manipulator, posing some challenges from the control perspective. Not only is the state space high-dimensional and redundant, the morphology also diverges from the sequential manipulator and resembles a tree. Rooted at the torso, five branches originate: One for each arm and leg and one for the head. To tackle this tree structure, most control methods separate the body into functional groups. While the legs and hip are used for walking, manipulation only involves the torso and the arms. It is then possible to address the walking problem with specific control methods, e.g. (Harada et al., 2004). Movements that only involve the upper body can be generated with the previously presented motion control approach for redundant manipulators.

One of these upper body control methods is the *whole body motion control* framework developed by Gienger et al. It aims to facilitate the high level of versatility of humanoid robots given by their complex body structure. As one example, manipulations can be done with either the left or the right hand, or even with both. The framework consists of multiple components and a summary is given in (Gienger et al., 2010). It is build around the known resolved motion rate control approach with gradient projection redundancy resolution and adds some enhanced abilities: Flexible task representations, displacement intervals for weakly constrained motions, self-collision and obstacle avoidance. Further it is possible to use the control method as the basis for stance point and trajectory optimization, including the selection of optimal grasps. The framework is able to compute motions of the upper body of a humanoid robot in real-time, while walking and the lower-body of the robot are not explicitly controlled. Walking abilities are handled in the framework by specifying high-level stance positions, control of the actual walking pattern is not considered.

2.5.1 Task Attractor Dynamics

Tasks can be defined in a very flexible way by utilizing the tree structure of the robot. Instead of controlling the end-effector with respect to a fixed robot coordinate frame only, whole body motion control allows to define tasks as the movement of one body with respect

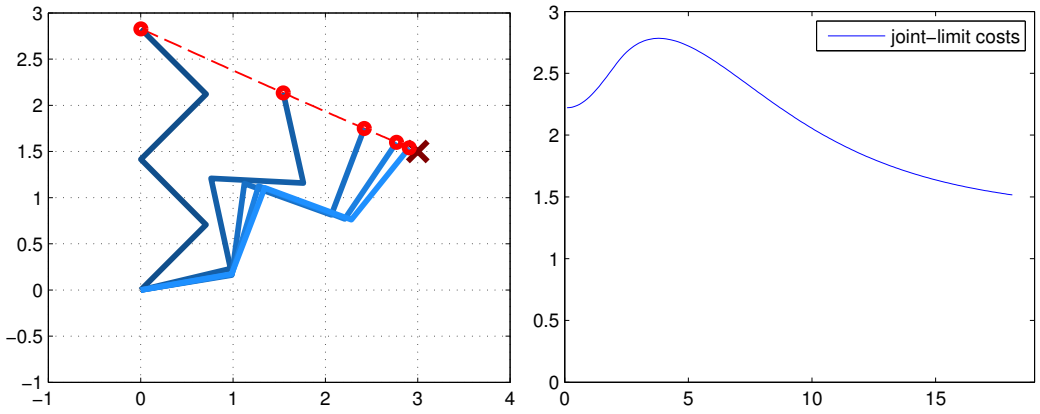

Figure 2.6: Motion rate control of a redundant manipulator, doing a movement towards the target position marked by the cross (left). The pseudoinverse approach is used here without additional redundancy resolution. For comparison, the joint-limit cost function is plotted against time (right).

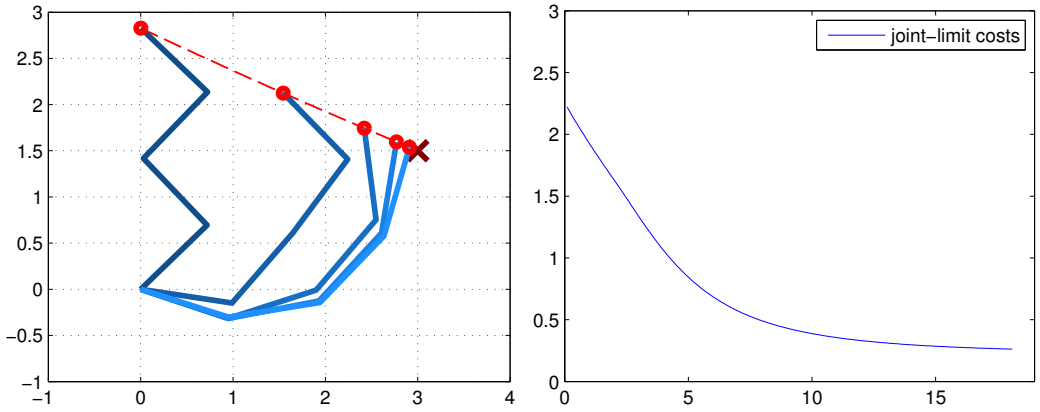

Figure 2.7: Resulting trajectory if the redundant space is utilized to avoid joint limits (left). The value of the joint limit cost function is plotted against time (right).

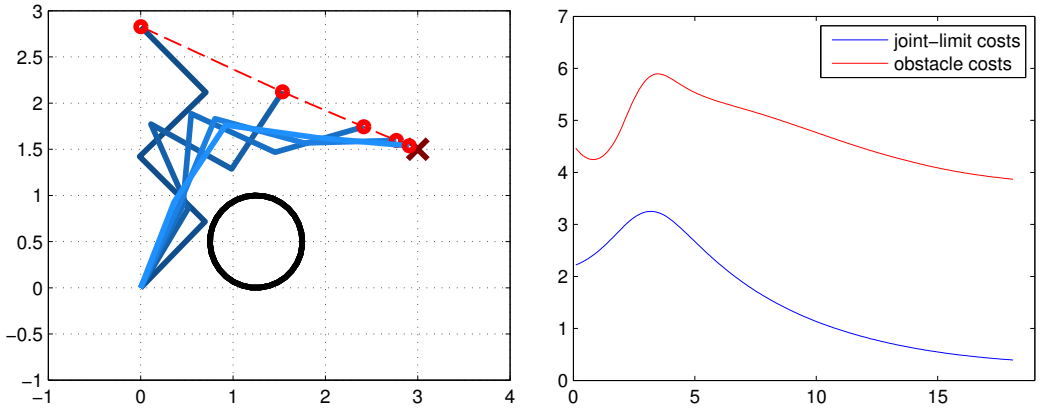

Figure 2.8: Using the redundant space for obstacle avoidance (left). The value of the obstacle avoidance cost function is plotted against time (right). Also, the joint-limit cost function is shown, although not explicitly minimized here.

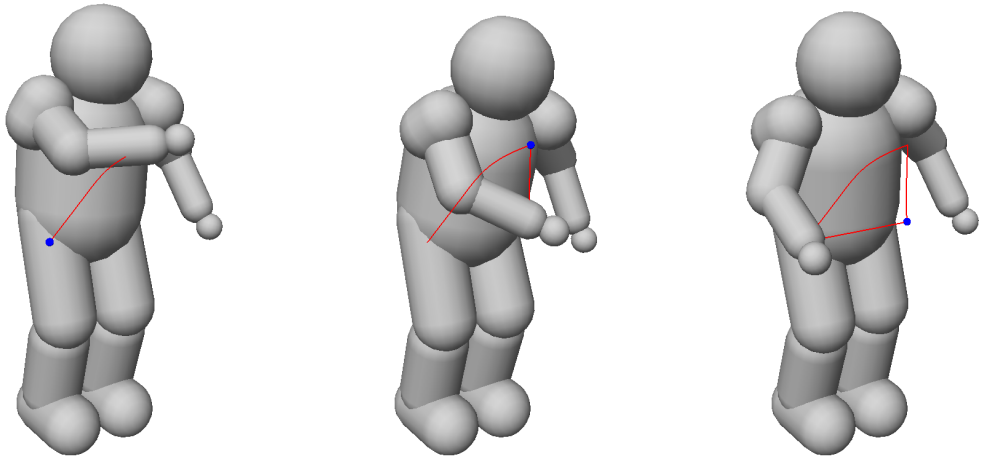

Figure 2.9: Simulation of a humanoid robot under whole body motion control. Starting at the marked positions (blue ball), three position targets are approached. The resulting trajectory is shown in red.

to any other body in the tree (Gienger et al., 2010). For example, it is possible to control the left hand position with respect to the head or relative to the right hand position. Since external objects can also be part of the tree, it is also possible to control motions with respect to other objects too. In addition to that, the dimensionality of task vectors is not fixed. Gienger et al. (2005) illustrates the cases of three-dimensional position and five-dimensional position and orientation tasks, but in principle any kind of task modality could be used. This way, it is easy to only constrain relevant dimensions and to leave all others unconstrained. For example, consider the task of grasping a cylindrical object, symmetrical around its vertical axis. In order to pick up the object, it is sufficient to control the hand orientation relative to this symmetry axis, while the actual orientation around this axis is not relevant for the task.

Independent of the actual task definition, the goal is always represented by a target attractor point, following the dynamical system approach described in section 2.1. The dynamical system used here is a second order attractor dynamic similar to DMP. Given a single target attractor point, a vector field is defined over the whole task domain, in which every trajectory converges to the target. The dynamical system is given in a discrete formulation in Toussaint et al. (2007). Alternatively, it can be expressed with a continuous state transition function $\ddot{y} = h(\dot{y}, y, u, t)$, defining an acceleration $\ddot{y}$ for every position y and current velocity at this position $\dot{y}$:

$$\ddot{y} = \alpha(u(t) - y) - \beta\dot{y} \tag{2.14}$$

Coefficients α and β determine the convergence behavior. The task enters the dynamic with the time dependent parameter $u(t)$, which is just a time dependent linear interpolation between an initial starting point y_k and a target point y_{k+1}:

$$u(t) = \begin{cases} y_k + (y_{k+1} - y_k)t & \text{for} \quad 0 \leq t \leq 1 \\ y_{k+1} & \text{else} \end{cases} \tag{2.15}$$

The second order transition function can be reduced to a first order transition function

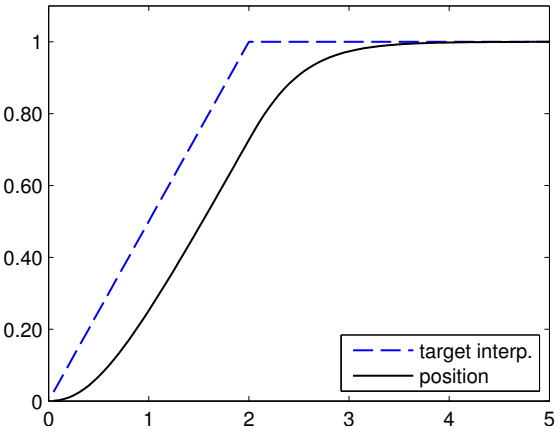

Figure 2.10: Example of a trajectory created with the second order attractor dynamic, showing the linear task function $u(t)$ (dashed blue) and the position y (black).

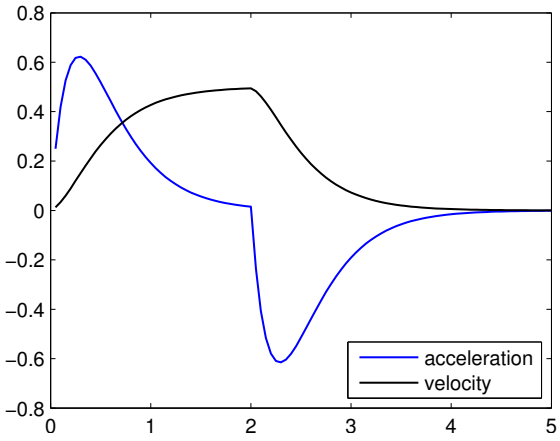

Figure 2.11: Acceleration $\ddot{y}$ (blue) and velocity $\dot{y}$ (black) profiles for the attractor dynamic example.

$\dot{x} = f(x, u, t)$ by representing it in the phase space $(y, \dot{y}) = (x_1, x_2) \in \mathcal{X}$:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \alpha(u(t) - x_1) - \beta x_2 \end{aligned} \tag{2.16}$$

Figure 2.10 shows a resulting trajectory, starting at $(y, \dot{y}) = (0, 0)$ with the target $y = 1$. Figure 2.11 illustrates the corresponding velocity and acceleration profiles. Parameters are $\alpha = 10, \beta = 6$.

2.5.2 Instantaneous Motion Optimization

Since most humanoid robots possess a high level of redundancy that can be exploited to fulfill secondary movement characteristics, the whole body motion control framework employs the gradient projection method introduced in section 2.4. Given desired task velocities $\dot{x}$, computed by the previously described attractor dynamic approach, matching joint velocities $\dot{q}$ are computed according to equation 2.7. During assembly of the Jacobian, the two arm morphology has to be respected.

The redundant space movement is given by an arbitrary joint-space cost function $H(q)$ that is locally optimized. Two different secondary movement objectives are defined. The cost function in (Gienger et al., 2005) is used to avoid joint-limits by penalizing deviations from a given reference- or home-posture, refer to section 2.4.1. Figure 2.12 demonstrates redundancy resolution with this cost function, showing the effect of favoring natural looking human-like postures.

Another use of redundancy is to avoid collisions of the robot, either with external obstacles or with itself, as shown in (Sugiura et al., 2007, 2010). For this a cost function based on the distance to obstacles is defined, as described in section 2.4.2. Figure 2.13 shows an example where the motion is optimized to penalize joint-limits and to stay away from the obstacle. Comparing this motion to the previous case of mere joint-limit avoidance, it is visible how the redundancy is effectively used to successfully circumvent the obstacle, without altering the task execution.

Using the collision avoidance cost function adds computational costs to each control iteration. Closest points between robot segments and obstacles have to be determined

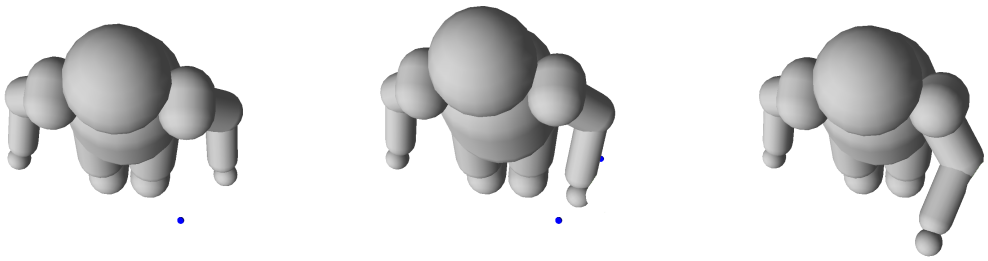

Figure 2.12: Example motion with joint limit avoidance. Local redundancy resolution drives the robot towards postures minimizing the deviation from a pre-defined home position, resulting in a natural looking movement.

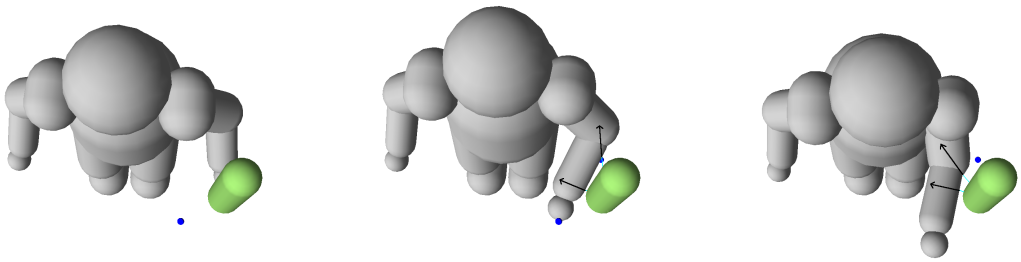

Figure 2.13: Example motion with joint limit avoidance and obstacle avoidance. Based on the distance of robot segments to the obstacle, locally optimal avoidance motions (black arrows) are projected into the redundant space.

and in order to project the corresponding costs into the joint-space, local Jacobians at the closest points have to be computed. However, if the number of segments is limited and simple geometric primitives are used, the overhead remains small and control can still be done in real-time. Throughout this work, all geometries consists of spheres, boxes and sphere-swept-lines, for which efficient distance computation and collision checking is possible, e.g. (Larsen et al., 1999; Van den Bergen, 1999)

Despite the computational burden, the present collision avoidance method exhibits principal limitations. One is the restriction to the redundant space. If an obstacle can not be avoided without modifying the task trajectory, a collision can not be prevented. The second limitation is due to the local nature of the gradient descent cost minimization. Although an optimal motion is done in every step, optimality is only locally evaluated. With respect to the whole trajectory, gradient descent is likely to confine to sub-optimal solutions or even to run into local minima.

2.6 Constraints

The previous sections outlined ways to represent and generate movements for robotic manipulators. Starting with some task description, given in a suitable representation, motions for each joint are computed with the Jacobian pseudoinverse approach. If the mechanism possess redundancies regarding the task, the resulting freedom of multiple solutions can be effectively exploited with the gradient projection method, for example to avoid joint-limits or obstacles. At the end of this process, joint-velocities are calculated and by integrating them in time, a joint-space trajectory is created.

In this section, another look at the motion generation process is taken, examining the behavior of the resulting movement paths. More specifically, the question is addressed if it

is indeed possible to create every movement possible, or if there are paths that can not be taken and positions that can not be reached. The question is if the set of possible paths and the set of reachable states is limited in any way by *constraints*.

Two types of constraints were already encountered before: Joint-limits can hinder a manipulator to reach postures outside an allowed joint-range and obstacle constraints forbid postures that result in a collision of the robot with itself or with other obstacles. These constraints block certain parts of the state space and can be characterized as a binary function, indicating for every state of the robot if it is allowed or not. They prevent the creation of every possible movement and limit the reachable positions, but since they originate from mechanical properties of the modeled system, their adherence is highly desirable and does not result in practical limitations.

A different kind of constraint is not as easily characterized. *Kinematic constraints* limit the freedom of movement in a less obvious way. Following Latombe (1991), kinematic constraints can be distinguished into two classes:

Holonomic constraint:

Supposing there is a robot at time t and state q, a holonomic constraint imposes a scalar constraint of the form $F(q,t) = 0$.[2] If there are k independent constraints defined, a holonomic constraint determines a $(m-k)$ dimensional manifold of the m dimensional state space. The robot can freely move in this manifold but is not able to leave it. An example for a holonomic constraint is a block that lies on a table and that can only be manipulated by pushing it. The block can not move freely in space, it can not be lower than the table and it can not go above the table, its movements are limited to sliding on the table surface. However on the table surface, the block can be pushed to any possible position.

Nonholonomic constraint:

If a scalar constraint on the robot state q can not be expressed without derivatives, i.e. is of the form $F(q,\dot{q},t) = 0$, it is called a nonholonomic constraint. The set of possible velocities $\dot{q}$ is restricted to a subspace of the tangent space T_q of state q. Thus a robot is no longer able to move from any state to any state, even if both states are in principle allowed. The directions in which motion is allowed are determined by the nonholonomic constraint. A common example is the car-like motion behavior. A car can move on a horizontal plane, but it can only move forward and backward, with the direction determined by the state of the steering wheel. It is not possible to slide the car perpendicular to its driving direction. However, we know by experience that a car is nevertheless able to reach every possible state – only the path might become more complex. For example it is possible to reach states perpendicular to the driving direction by performing a parallel parking maneuver.

Note that the presence of derivatives in a constraint equation does not always imply a nonholonomic constraint. Sometimes it is possible to eliminate the derivatives in a different but equal form of constraint equation. Only if this integration is not possible, the constraint is said to be nonholonomic.

The car example is a good way to show the relevance of this principal distinction into holonomic or nonholonomic constraints. Only the non-holonomity of the system allows the car to drive to every possible state. In contrast, the movement range of a car with a holonomic constraint would be severely limited, as for example happening when the steering

[2] Inequality constraints $F(q,t) \leq 0$ are also possible

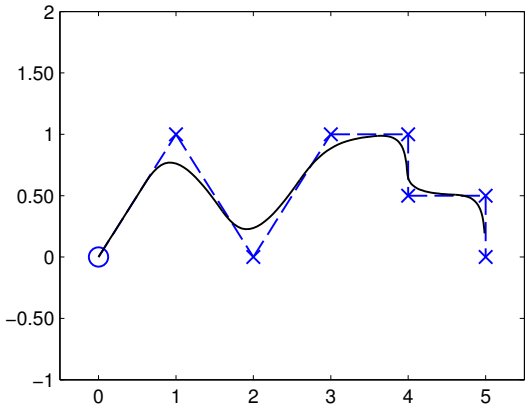

Figure 2.14: Starting at the position marked by the circle, several target positions marked by the crosses are fed into the system in sequential order. The switch from one target to the next occurs before the target is reached.

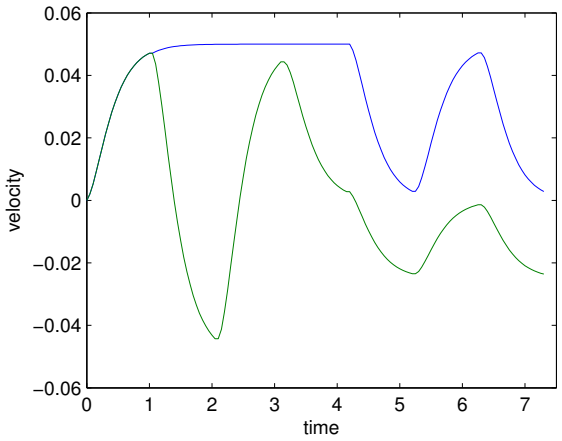

Figure 2.15: The corresponding velocity profiles show that there is always one velocity component grater than zero, resulting in a steady movement without stops. (blue: velocity in x-direction, green: velocity in y-direction)

wheel remains fixed at one position. Possible movements would then be limited to a manifold described by a closed circle.

In the following, the nature of the constraints arising from the described control methods is analyzed. This is especially interesting once motion control is used for planning, where movements no longer consist of single point to point motions but of multiple consecutive motions on a complicated path. Characterizing the constraints answers the question if there are any additional limitations on the paths, besides the compulsory prevention of joint-limit or obstacle constraints. This can be systematically addressed by examining the *reachability* of the system by looking at the *reachable set* of all states of the system (LaValle, 2006, chapter 14). It is defined as the set of all states visited by any possible trajectory starting at a fixed state. A general property is that if the constraints are holonomic, the set of reachable states is always limited to a manifold of the state space. If the constraints are nonholonomic, the complete state space can be reachable, depending on the specific properties of the constraints.

To characterize the motion control approach used in this work, two components are examined separately. The process of computing task velocities $\dot{x}$ from a task target, covered in the next section 2.6.1, and the control law computing joint-velocities $\dot{q}$, covered in section 2.6.2.

2.6.1 Attractor Dynamics

In section 2.5.1 a second-order attractor dynamical system was presented, with the goal of creating a smooth task space trajectory towards a given task target. Equation 2.14 constraints the acceleration $\ddot{y}$ in dependence of the current state position y and state velocity $\dot{y}$. The desired task target is given by $u(t)$, an attractor point linearly shifted towards the target. The system can be written as a constraint equation

$$\alpha(u(t) - y) - \beta\dot{y} - \ddot{y} = 0 \tag{2.17}$$

including first- and second-order derivatives. Due to the acceleration constraint, it is not possible to instantly change the direction of motion, as shown in figures 2.14 and 2.15. If the target position is changed before a target was reached, the motion generation process

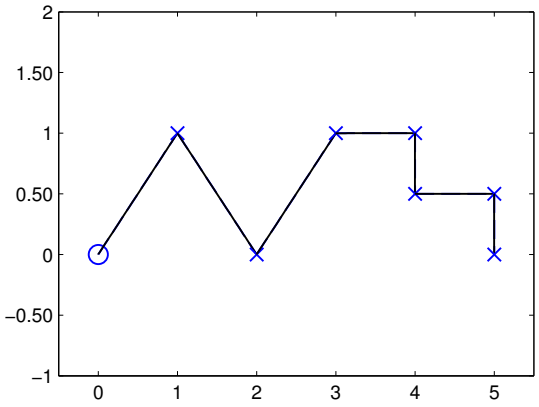

Figure 2.16: The same target positions are approached in sequential order, but this time all intermediate positions are reached with zero velocity.

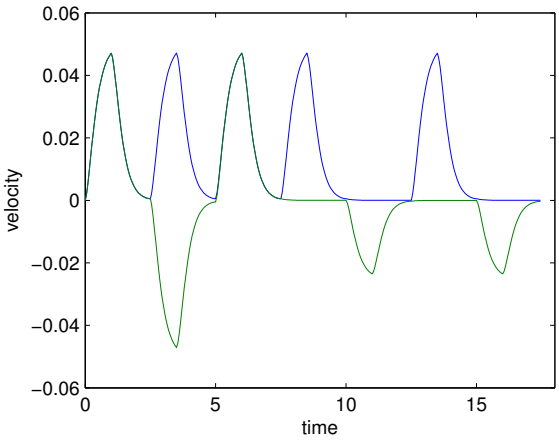

Figure 2.17: The velocities are (close to) zero when a target is reached and the system switches to the next target. (blue: velocity in x-direction, green: velocity in y-direction)

smoothly changes the motion direction towards the new target. There is momentum left from the previous motion and since acceleration is constrained, counteracting the momentum needs some time. This limitation of possible accelerations in every state of the motion implies the presence of nonholonomic constraints. Indeed, Appendix A shows that it is not possible to express this constraint in an integrated form not depending on velocities, in particular on the initial velocity at the beginning of the motion. Looking at a series of movements in the previous example, this means that each sub-trajectory is influenced by its predecessor because the initial velocity of every new movement equals the end velocity of the previous one.

A special case arises if movements are restricted to direct target-to-target movements, starting at a rest position with zero initial velocity and ending at another rest position with an almost zero velocity. If subsequent target-to-target movements are done, the influence of the previous movement is neglectable because the close to zero initial velocity has almost no impact. All possible movements in all directions can be done from any sub-trajectory end-point, regardless of the actual sub-trajectory that led to that end-point. Figures 2.16 and 2.17 illustrate how the direction changes between target points can now be precisely followed.

Regarding the state reachability of the dynamic system, it can be assumed that every position can be reached. Although the set of possible sub-trajectories starting at a given state is constrained, i.e. certain accelerations might be impossible, the set of reachable trajectory end-points is not limited. If the additional condition of zero velocity at switching points between sub-trajectories is imposed, the dynamic system behaves like a holonomic constraint: Originating from such a switching point all possible motions are allowed, there are no impossible accelerations. All target states can be reached on a straight-line path and every subsequent straight-line path is completely independent of its predecessor.

2.6.2 Motion Rate Control

Section 2.2 introduced the Jacobian psuedoinverse motion rate control method as a way to generate joint-space motions $\dot{q}$ for desired task motions $\dot{x}$. If the mechanic structure is redundant with respect to the task, the control system is underactuated and movements in the joint-space are implicitly subject to differential constraints. De Luca and Oriolo

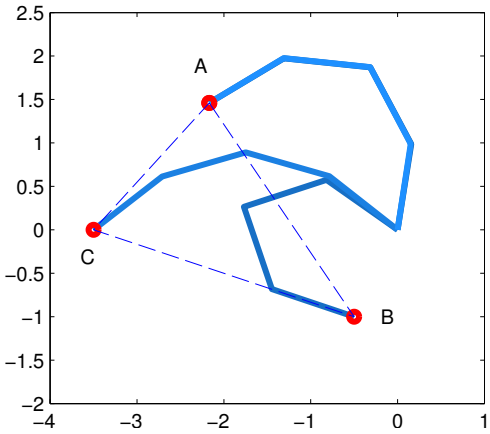

Figure 2.18: Simulation of a manipulator performing a closed loop movement from A to B to C and back to A. Redundancy is resolved with the gradient projection method and the joint-limit avoidance cost function. The manipulator converges to the same posture at position A after performing the loop.

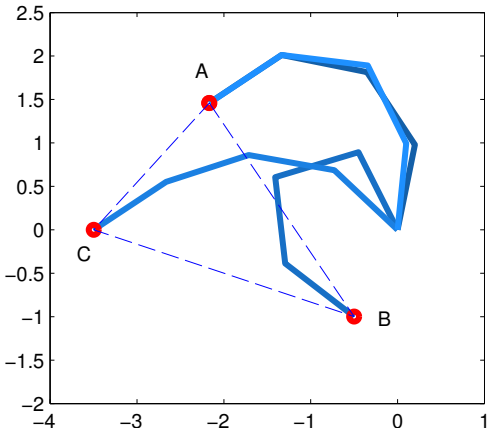

Figure 2.19: The same closed loop movement from A to B to C and back to A. In comparison to the figure to the left the gradient steps are smaller, resulting in different redundant space configurations. After performing the loop, the manipulator converges to a slightly different configuration at position A.

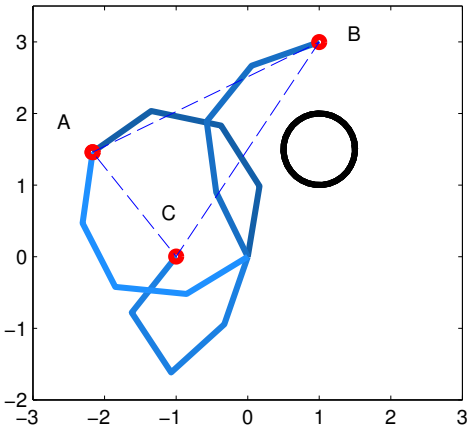

Figure 2.20: A closed loop movement from A to B to C and back to A in presence of an obstacle. The redundant space cost function consists of both joint-limit avoidance and obstacle avoidance. Due to the obstacle, the manipulator converges to a completely different configuration at position A after the loop. It starts bending to the right (dark blue) and ends bending to the left (light blue).

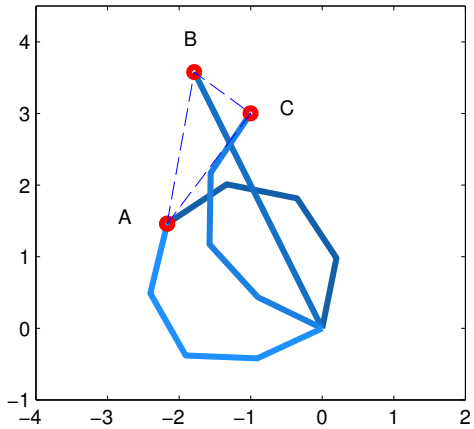

Figure 2.21: A closed loop movement from A to B to C and back to A passes through a singularity at position B. As a result, the manipulator reaches A with a different configuration, representing an alternative solution to the inverse kinematic problem. It starts bending to the right (dark blue) and ends bending to the left (light blue).

(1997) characterize these constraints as being either holonomic or nonholonomic and show the equivalence to the well studied problem of the repeatability or cyclicity of redundant manipulator movements. Holonomic constraints result in a cyclic control scheme where every closed task-space loop also results in a closed joint-space loop. If constraints are nonholonomic, complete cyclicity is lost. Integrability of the constraints or equivalently cyclicity of the control can be tested with the help of the Frobenius theorem (Shamir and Yomdin, 1988). Unfortunately, this depends heavily on the actual mechanism and Jacobian inversion and it is hard to state general properties.

In this work, pseudoinverse control with gradient projection redundancy resolution is used, refer to section 2.4. With equation 2.7, it is possible to explicitly write every possible control law within this scheme as

$$\dot{q} = J^{\#}(q)\dot{x} + (I - J^{\#}J)z \tag{2.18}$$

The second term maps an arbitrary vector z to the nullspace $\mathcal{N}(J)$, while the first term uniquely maps to the orthogonal space $\mathcal{R}(J^{\#}) = \mathcal{N}^{\perp}(J)$. By choosing z, every possible inversion of equation 2.1 can be created and the actual choice determines if the control is holonomic or not. Without loss of generality, the following statements about the holonomy of gradient projection control can be made:

1. If the system is holonomic, the control scheme is cyclic. A path starting at (x_0, q_0) and ending at (x_n, q_n) is cyclic if a closed task space loop $x_n = x_0$ results in a closed configuration space loop $q_n = q_0$ as well. This implies that the redundant inverse kinematic mapping has to map x_0 and x_n to the same q, thus z has to be unique for every q in equation 2.18 to let $\dot{q}$ converge to the same q.

   The gradient projection method sets z as the cost-function gradient $z = \gamma\nabla H(q)$. While the gradient is unique for every q, the convergence to the same q can not be guaranteed! Depending on the step-width γ, the control might converge to different joint configurations at the same task position, as illustrated in figures 2.18 and 2.19. If the cost function $H(q)$ possess local minima, like the obstacle avoidance cost function in section 2.4.2, the gradient might even converge to entirely different minima, see figure 2.20.

2. The control law in equation 2.18 becomes unfeasible at singularities, where J is not of full rank. For this reason, the treatment of cyclicity and holonomic control is often limited to non-singular regions of the space (Shamir and Yomdin, 1988; De Luca and Oriolo, 1997). However, it is possible to move through singularities with a singulartiy robust pseudoinverse, see section 2.3. The control law returns feasible joint-rates even inside singularities and trajectories can thus move from one region of the solution space to another region, as illustrated in figure 2.21. It is visible that cyclicity is lost in this example, because the same task positions are now reached by completely different joint configurations. The space of valid joint-postures for a given task position is partitioned into different solution regions connected through singularities. Notably this is also the case for non-redundant structures. A two joint manipulator (relative joint angles θ_1 and θ_2) with a two dimensional positioning task is not redundant, yet every task position can be reached with the "elbow" joint either pointing to the left ($\pi > \theta_2 > 0$) or to the right ($-\pi < \theta_2 < 0$). To pass from one solution to the other, one has to go through the "stretched out" ($\theta_1 = \theta_2 = 0$) or "folded" singularity ($\theta_2 = \pm\pi$).

Considering these two points together, the gradient projection control scheme can only be characterized as holonomic under two conditions: The first condition is that holonomic

behavior is limited to singularity free regions. Passing through a singularity, the system enters a different solution region with a different posture for the same task position and holonomy is lost. The second condition requires that the global minimum of the redundancy resolution cost function is always found inside a singularity free region, resulting in an unique mapping of states to a manifold. A diverging mapping can result in diverging postures for identical task positions, also violating holonomy.

In practice, these to conditions are generally not met. With the use of singularity robust control, the system can pass through singularities. Also, using gradient decent based optimization for redundancy resolution might lead to different convergence points and states diverging from the manifold. Thus, the motion control approach presented in this chapter has to be characterized as possessing nonholonomic behavior. The reachability of the control system with respect to the complete state space is not limited to a single manifold. It is possible that identical task positions are reached with different joint postures, either by diverging redundant space gradients or after the transition into a different solution space region through a singularity. This does not necessarily mean that the whole configuration space becomes reachable, but at least the region of reachable configurations is substantially enlarged in most cases.

Chapter 3

Relaxed Motion Control

The previous chapter 2 introduced motion control methods for redundant manipulators, where the redundancy is exploited to optimize secondary motion objectives without disturbing the primary task. In this chapter, a control method is developed that weakens this strict prioritization in order to gain more freedom in motion optimization. After an introduction in section 3.1, the relaxed motion control method is described in section 3.2. It can also be integrated into the whole body motion control framework, as elaborated in section 3.3.

3.1 Multiple Motion Objectives

The Jacobian pseudoinverse gradient projection technique demonstrates a way to utilize redundancies for the fulfillment of additional motion objectives, section 2.4. This kind of motion optimization is by design restricted to the redundant space and is not interfering with the task. Motion objectives are strictly prioritized: Most important is the fulfillment of the primary task and only if there is some freedom left, secondary tasks like obstacle avoidance are considered. While this property is often desired if precise task execution is the main objective, one can think of situations where this strict devision is not necessary or even cumbersome. For example, consider a simple pick and place task where the goal is to reach a certain target position. A direct and straight trajectory towards the target is of course a reasonable solution, but not necessarily the only solution. Every deformation of the trajectory is also a valid solution, as long as the end-points remain fixed. While the straight trajectory minimizes the total trajectory length, other trajectories might minimize other cost objectives, like the already introduced joint-limit or obstacle distance costs. For instance, if we allow the modification of the task trajectory according to obstacle distance costs, the resulting trajectory could bend around obstacles.

This example illustrates a case where it makes sense to weaken the strict prioritization into task objective and redundant space objective. By deliberately allowing the deviation from straight task execution, it is possible to put greater weight on the optimization of secondary motion properties. This is especially beneficial if the available redundant space freedom is limited. If the task at hand allows it, one can trade precision and predictability of task execution for lower cost trajectories, for example for trajectories staying further away from obstacles.

In this chapter, the control scheme of chapter 2 is enhanced to allow deviations from direct task execution in order to foster the minimization of an additional cost function. The strict division into primary and secondary motion objective is broken by allowing the secondary motion to interfere with the primary one. The influence of the secondary motion can be fixed to have the same priority than the primary one, or can be adaptively weighted to have a lower priority in conflict situations. This leads to a motion control law with a "relaxed" attitude regarding the compliance with different task objectives.

Several control methods can be found in the literature that handle multiple motion objectives simultaneously. Escande et al. (2010) solve a stack of tasks with quadratic

programming and conflicting objectives are resolved by defining a strict hierarchy. For cases in which such strict priorities are not desired, Salini et al. (2011) resort to a soft hierarchy of tasks with an importance weighting strategy that also allows to do smooth transitions. Tasks and constraints are formulated as an optimization problem and solved with linear quadratic programming. With our proposed control method, it is also possible to define a task hierarchy to resolve conflicts and to switch between tasks in a smooth way. In contrast to the optimization based methods, a pseudoinverse control approach is used, with tasks and constraints formulated as potential functions. Also, task weighting is done on task- instead of joint-level.

A comparable control based approach is presented in (Brock and Khatib, 2002), based on an adjustable task representation called elastic strips. Consisting of a sequence of sub-targets, the strip can be dynamically adapted according to external forces exerted by obstacles, pushing the path away, and internal forces, ensuring its closure. Tasks can also be defined with priorities, using the operational space formulation, and it is possible to suspend individual tasks and to smoothly switch between them. Unlike elastic strips, the relaxed motion control method developed in this chapter does not operate on a whole sequence of sub-targets. Only a single target is given and the resulting path is always consistent, without the need to maintain the path closure.

A way to relax task constraints in the whole body motion control framework from section 2.5 is developed in (Gienger et al., 2006). The secondary motion component, originally used for redundancy resolution, is projected onto the task space. There it is overlaid with the target directed motion, leading to a displacement away from the original trajectory in a direction that minimizes the secondary motion cost function. If the secondary motion optimization criteria is given by a joint-limit avoidance cost function, it is shown that resulting trajectories indeed stay further away from joint-limits and the movements are more natural looking. Since the trajectory displacement is limited to an allowed interval, it is still possible to ensure certain task necessities. For example the hand can be restricted to stay in a tunnel around the task trajectory or the angular deflection of a grasped bottle can be limited, avoiding an accidental pouring of the content.

The proposed relaxed motion control method also uses the approach of projecting a secondary motion joint-space gradient onto the task space, superposing the primary target reaching motion. However, the resulting displacement is not restricted to a certain interval that limits the freedom in fulfilling the optimization. Task constraints are weakened to a greater extend, the only requirement remaining is to exactly reach the commanded target position. That is a difference to (Gienger et al., 2006), where the adherence to the displacement interval is guaranteed, not the convergence to the target.

The control scheme in (Sugiura et al., 2010) also utilizes the whole body motion control framework, employing the gradient projection method for redundant space obstacle avoidance. Since this approach puts onesided priority to task reaching, an additional controller is developed that interchanges priorities. The first priority is to avoid obstacles, while target reaching plays a subordinate role in the redundant space of the obstacle avoidance motion. The control output of both controllers is then smoothly blended between the two priorities task reaching and self-collision avoidance, based on the closest distance between robot segments. If the distance falls below a defined threshold, the priority of self-collision avoidance gets increasingly larger while the priority of task reaching gets proportionally lower. This way, uncritical task targets are reached and critical targets resulting in self-collisions are executed as far as possible.

Relaxed motion rate control allows to smoothly vary the influence of the secondary motion to the task execution with an a-priori determined weighting parameter, comparable to the priority weighting of Sugiura et al. (2010). But unlike their approach, only a single control

equation is used. Also the weighting is not automatically adapted as a function of segment distances but in dependence to the target distance. The controller is enforced to always converge to the target position, even if it is necessary to move close to obstacles. The basic principle of the proposed method was presented in (Behnisch et al., 2011), but without the weight adaption technique enforcing the target convergence developed in this work.

3.2 Task Space Motion Optimization

The proposed relaxed motion control technique builds upon the known resolved motion rate approach, see the previous chapter 2. In the control equation, relating task velocities $\dot{x}$ to joint velocities $\dot{q}$, the task velocity $\dot{x}$ is replaced by the composite term $\dot{x} = \alpha\dot{x}_{tg} - \beta\dot{x}_{cf}$:

$$\dot{q} = J^{\#}(\alpha\dot{x}_{tg} - \beta\dot{x}_{cf}) - \gamma(I - J^{\#}J)\nabla H \tag{3.1}$$

The task velocity now consists of a weighted superposition of the target directed task space velocity $\dot{x}_{tg}$ and the task space velocity $\dot{x}_{cf}$, minimizing a arbitrary joint-space cost function $H(q)$. This velocity is determined by projecting the cost function gradient to the task space as

$$\dot{x}_{cf} = \frac{\partial H}{\partial x} = \frac{\partial H}{\partial q}\frac{\partial q}{\partial x} = \nabla H^T J^{\#} \tag{3.2}$$

The motion consists of three components now, a target directed motion, a local cost optimization motion in task space and a local cost optimization motion in the redundant space. The influence of each is weighted by the parameters α, β and γ, respectively.

Since the redundant space motion is by design not interfering with the task space motion, the weighting of this component with γ is independent of the weighting of the task space motion. Thus the value of this parameter is not very critical. For the weighting of the task space components however, this is not the case. Here the motions are both in the task space and are thus directly influencing each other.

The challenge is to weight both parts appropriately. For example, consider the cost function is designed to avoid obstacles, resulting in a task velocity away from obstacles. The two objectives of getting closer to the target and staying away from obstacles might be contradicting in certain target and obstacle configurations. In such cases, we propose to control the task space behavior by modifying the parameter β. If it is set to 1, there is a large emphasis on cost function minimization, e.g. obstacle avoidance. If it is set to 0, pure target following without any deviation is done. An example with obstacle avoidance is shown in figure 3.1 and 3.2, employing the cost function presented in section 2.4.2.

It is also possible to dynamically adapt α to assign an even higher priority to cost minimization, as shown in (Behnisch et al., 2011). In situations where the cost function gradient is large, α is smoothly tuned down, decreasing the influence of the target directed velocity and increasing the relative weight of β and the cost minimization velocity.

The example in figures 3.1 and 3.2 shows that, depending on the spatial arrangement of target and obstacles, the control might converge to different points for different β. Why this is happening is illustrated in figures 3.3 and 3.4 for the case of $\beta = 0.5$. The two velocity vectors $\dot{x}_{tg}$ and $\dot{x}_{cf}$ are both acting in the same space, and depending on α and β, it might happen that their difference is zero, $\alpha\dot{x}_{tg} - \beta\dot{x}_{cf} = 0$. In order to prevent the controller to converge to points that are not the target, it has to be ensured that the target motion is always dominating the overall motion.

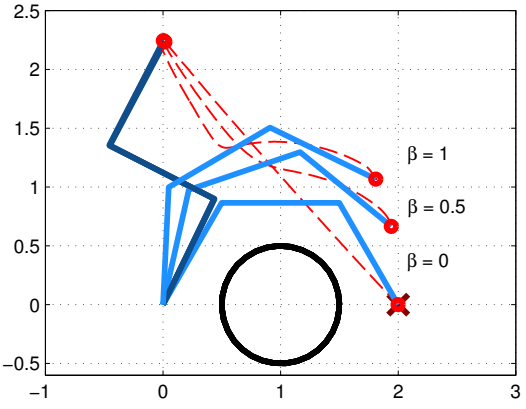

Figure 3.1: Example of mixing the projected obstacle avoidance cost function with the target directed motion. Different values of β between the extremes full target following ($\beta = 0$) and full obstacle avoidance ($\beta = 1$) are shown.

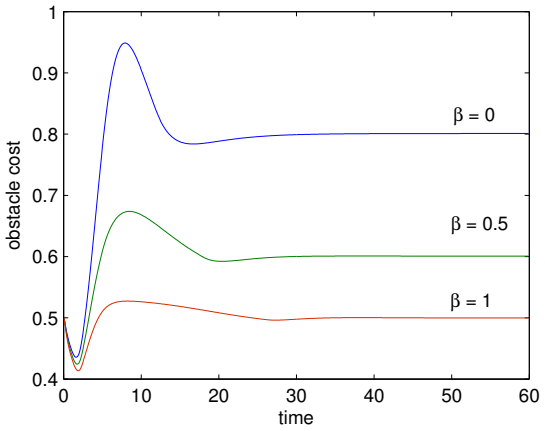

Figure 3.2: Obstacle avoidance cost function H_{coll} over time, for different β. If the obstacle cost gradient is mixed in to a greater extend for larger β, the costs remain smaller.

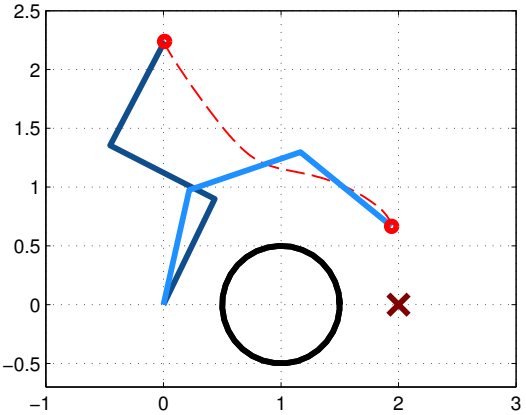

Figure 3.3: Mixing target motion and obstacle avoidance motion with $\beta = 0.5$, the controller fails to reach the target and converges to a different task position.

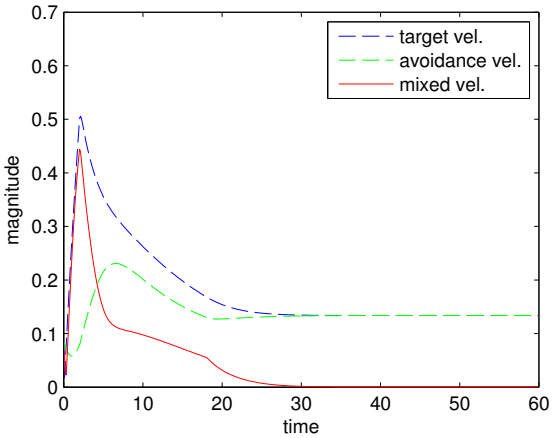

Figure 3.4: Magnitudes of the target velocity vector, avoidance velocity vector and resulting mixed velocity vector over time. At one point, target and avoidance velocity become equal and cancel each other out, resulting in zero mixed velocity.

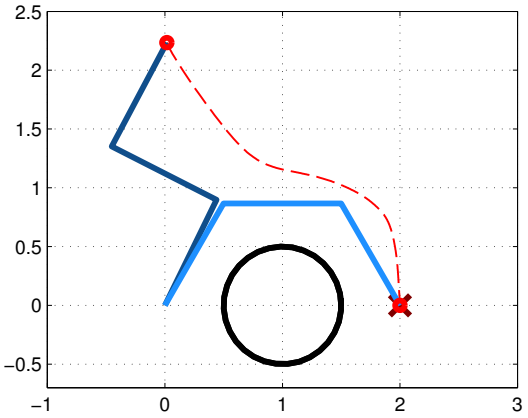

Figure 3.5: The same example with adaptive control of β'. The controller is able to reach the target position and still avoids the obstacle.

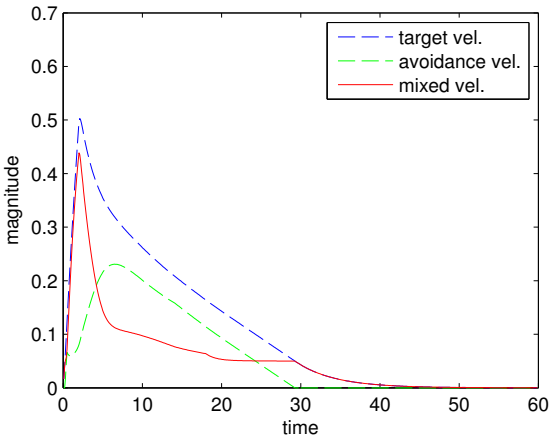

Figure 3.6: Magnitudes of the target velocity vector, avoidance velocity vector and resulting mixed velocity vector over time. By adapting β', it is ensured that the target velocity always dominates the avoidance velocity.

A way to do this is to ensure that the magnitude of the target directed velocity is always greater than the magnitude of the cost minimization velocity, i.e. that

$$\alpha||\dot{x}_{tg}|| > \beta||\dot{x}_{cf}|| \tag{3.3}$$

is valid during the whole movement. By modifying the influence of the cost minimization velocity by adapting β in equation 3.1, this condition can be guaranteed. β is replaced by β', in dependency of the following three cases:

$$\beta' = \begin{cases} 0 & \text{if} \quad \alpha||\dot{x}_{tg}|| \leq \epsilon \\ (\alpha||\dot{x}_{tg}|| - \epsilon) \;/\; ||\dot{x}_{cf}|| & \text{if} \quad \alpha||\dot{x}_{tg}|| < \beta||\dot{x}_{cf}|| + \epsilon \\ \beta & \text{if} \quad \alpha||\dot{x}_{tg}|| \geq \beta||\dot{x}_{cf}|| + \epsilon \end{cases} \tag{3.4}$$

If the target motion is below a small threshold ϵ, the cost minimization motion is completely disabled by setting β' to zero. The cost minimization term vanishes, $\beta'||\dot{x}_{cf}|| = 0$, and condition 3.3 holds. If the target motion is at least greater than the cost minimization motion by ϵ, the condition also holds and β' is simply set equal to β. If the target motion is in between these two boundaries, $\epsilon < \alpha||\dot{x}_{tg}|| < \beta||\dot{x}_{cf}|| + \epsilon$, the adaptive weighting parameter β' is computed to achieve the dominance of the target motion term by reducing the influence of the non-target motion. The required condition is met by enforcing the relationship $\alpha||\dot{x}_{tg}|| - \beta'||\dot{x}_{cf}|| = \epsilon$, scaling the non-target motion down in order to keep the magnitude of the target motion larger by ϵ.

Figures 3.5 and 3.6 illustrate the control behavior when using adaptively weighted obstacle avoidance. Both objectives of reaching the goal and staying away from obstacles can be met. How the resulting trajectories look like for different β is shown in figures 3.7 and 3.8.

The solution of determining β' in dependency of the velocity magnitudes is a conservative approach. It does not take the direction of the vectors into account and the premise that equal vector magnitude results in zero motion only holds if the vectors are exactly opponent to each other. In all other cases, there is some resulting movement, possibly towards the target as well. By enforcing a larger target motion in these cases, our approach is restricting

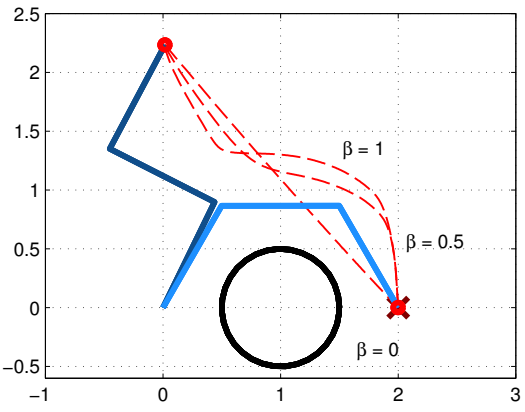

Figure 3.7: The example of mixed target directed and obstacle avoidance motion revisited. Different values of β are shown, but this time β' is adapted to achieve the target convergence of the controller.

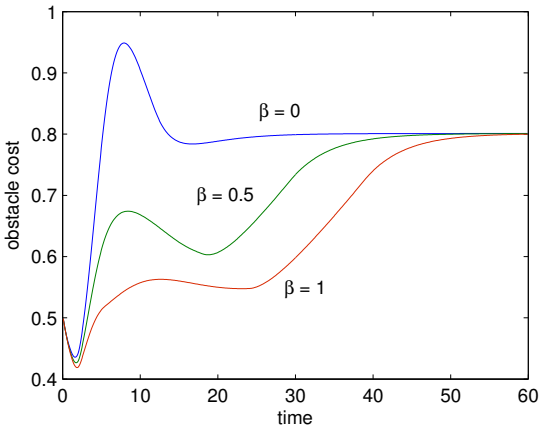

Figure 3.8: Obstacle avoidance cost function H_{coll} over time. Since the controller is forced to converge to the target, the final costs are the same for different β. Seen over the whole motion however, the costs can be effectively reduced in earlier stages of the movement with a larger β.

the cost function minimization more than necessary. An advantage of our conservative solution is that target convergence can be guaranteed right away in all circumstances.

Figure 3.9 shows another example. Here, two secondary motion objectives are used. One is the previously used obstacle avoidance and the other is joint-limit avoidance, as described in section 2.4.1. The corresponding cost functions are jointly optimized in the redundant space. Task space avoidance motions however are optimized with respect to obstacle costs only. The example demonstrates how these two costs behave for a motion sequence towards two task targets, comparing the case of strict hierarchical motion generation ($\beta = 0$) and the relaxed approach ($\beta = 1$) with enforced task convergence, i.e. adaptive β' control.

The resulting task trajectories confirm that the relaxed approach succeeds in simultaneously achieving the two tasks of obstacle avoidance and target reaching. The two targets are exactly reached and the paths in-between are significantly changed towards a greater distance to obstacles. At the end of the movement, the manipulator converges to a different posture with less obstacle costs. This can also be seen in the cost evolution plot. The obstacle costs are constantly lower over the whole trajectory if the relaxed method is used (red solid line), compared to the obstacle costs of redundant space optimization (red dashed line). Consequently, the overall costs integrated over the trajectory are lower. The joint-limit costs are also shown for reference (blue lines). Their optimization is restricted to the redundant space and is not possible to the same extend. Note that the costs are different for both approaches because of the different task trajectory and different manipulator postures.

3.3 Relaxed Whole Body Motion Control

The technique of relaxed motion control, as developed in the previous section, can be integrated into the whole body motion control framework of section 2.5. Since the control principle is the same, the whole body motion control law can be replaced with the relaxed motion control of equation 3.1. The task space projection (equation 3.2) and the adaptive β weighting (equation 3.4) then apply accordingly.

As secondary motion objectives besides the task motion, the known joint limit avoidance

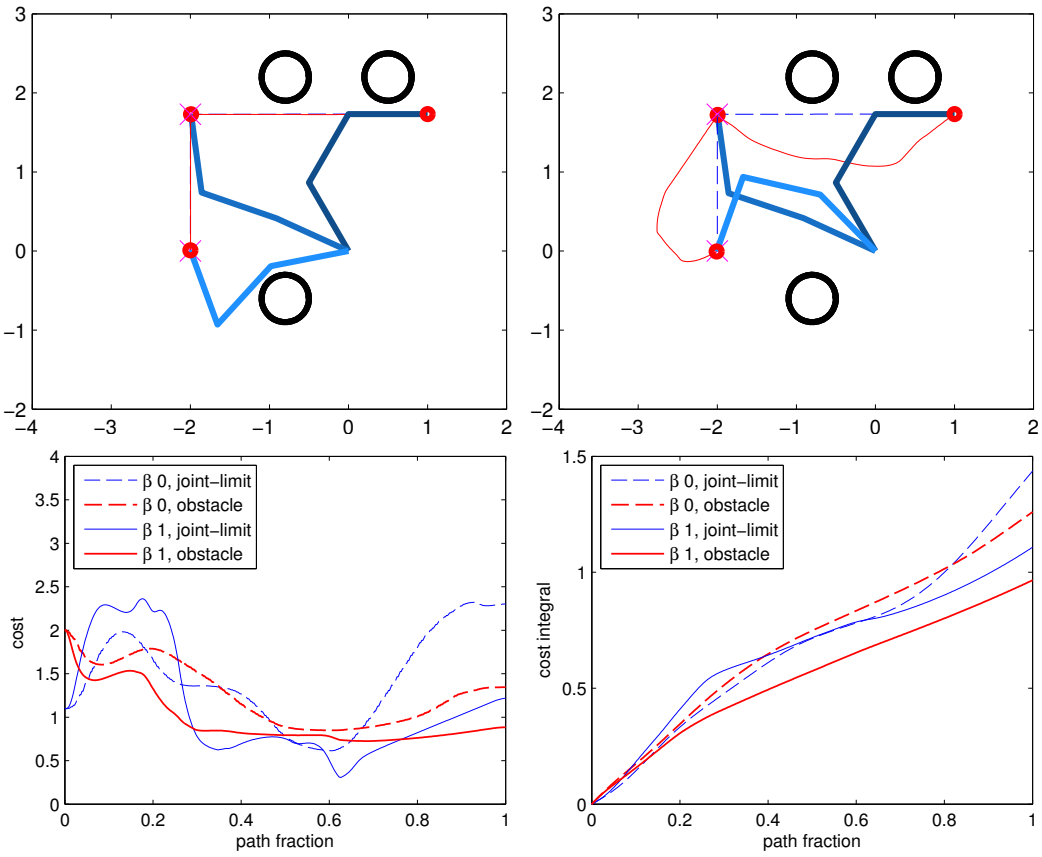

Figure 3.9: Comparing hierarchical motion generation with redundant space optimization $\beta = 0$ (top left) against the relaxed approach $\beta = 1$ (top right), also optimizing the task trajectory w.r.t. obstacle avoidance. Two consecutive movements are done, starting at the right side, the manipulator moves to the left and then down. Resulting costs are plotted over the path duration (bottom left), showing the individual evolution of joint-limit and obstacle costs. Cumulative cost over the whole movement are also shown (bottom right).

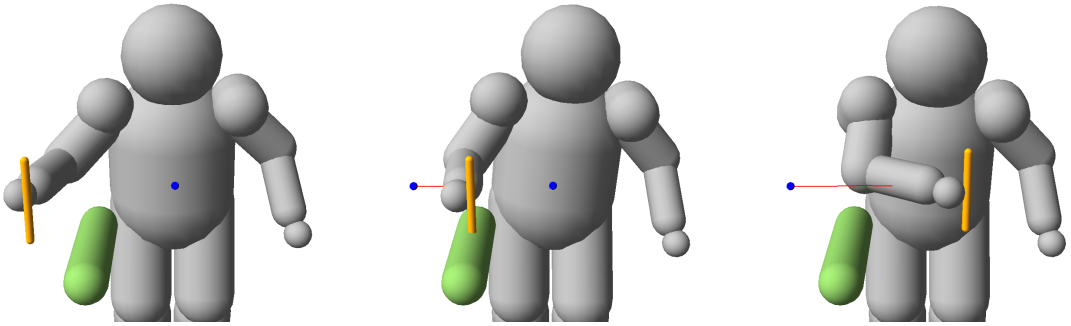

Figure 3.10: Example motion using Whole Body Motion Control with strict task priority. The trajectory results in a collision of the gripped object (orange) with the obstacle (green).

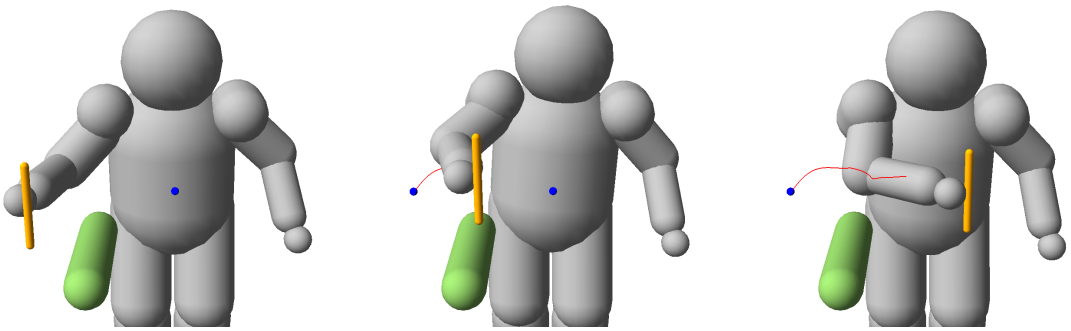

Figure 3.11: Example motion using Whole Body Motion Control and the Relaxed Task Space Control method. The secondary motion is given by a obstacle avoidance cost function and the task avoidance is weighted with $\beta = 1$. The obstacle is successfully circumvented.

cost function H_{jc} (section 2.4.1) and obstacle avoidance cost function H_{oc} (section 2.4.2) are used. The redundant space motion is computed using the sum of both joint-limit avoidance and obstacle avoidance, while the task space deviation motion is only based upon obstacle avoidance. This way, only the priority of the obstacle avoidance task is increased and allowed to interfere with the target task, while joint-limit avoidance stays on low priority, restricted to the redundant space. The reason for that is that we want to focus on circumventing obstacles and joint-limit avoidance could be counteracting this efforts. Joint-limit avoidance still happens, but only in a way that does not interfere with the task.

Some care has to be taken to assure the continuity of the trajectory, since the closest-points distance and therefore the obstacle cost function can be discontinuous. It is also possible that the a-priori weighting parameter β is changed during a trajectory or a trajectory sequence, in which case it needs to be smoothed to maintain continuity. In both cases, a simple low-pass filtering approach is taken.

The behavior of the developed control method is shown for a simulated humanoid in figure 3.11. To illustrate the difference to whole body motion control with strict task priority, refer to figure 3.10. The impact of the adaptive weighting technique with a dynamically changed β' is shown in figure 3.12. It is visible that indeed all trajectories converge to the desired task target. Another example with two obstacles is shown in figure 3.13. Again different values of β lead to different trajectories, but in this example a large influence of obstacle avoidance leads to collisions, while a lower weighted obstacle avoidance results in a trajectory successfully reaching the target.

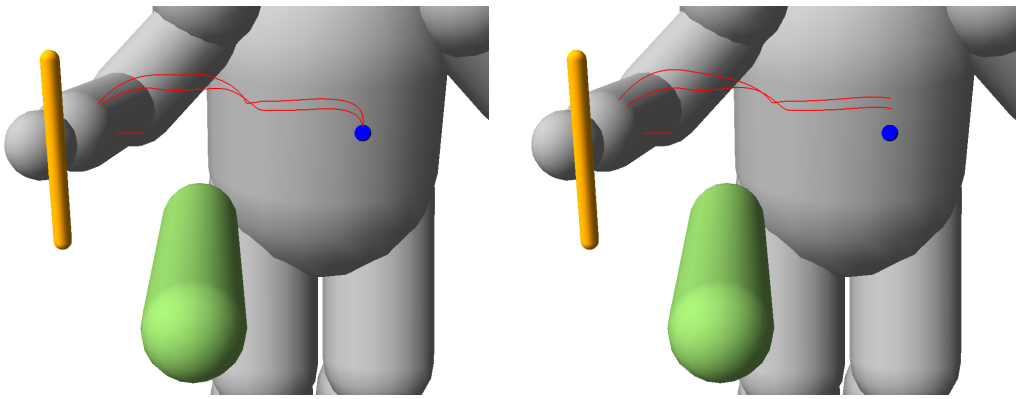

Figure 3.12: With adaptive weighting, it can be assured that all trajectories converge to the target (left). Trajectories for β values of 0, 0.5 and 1 are shown, from bottom to top. Without adaptive weighting, trajectory endpoints diverge significantly from the target.

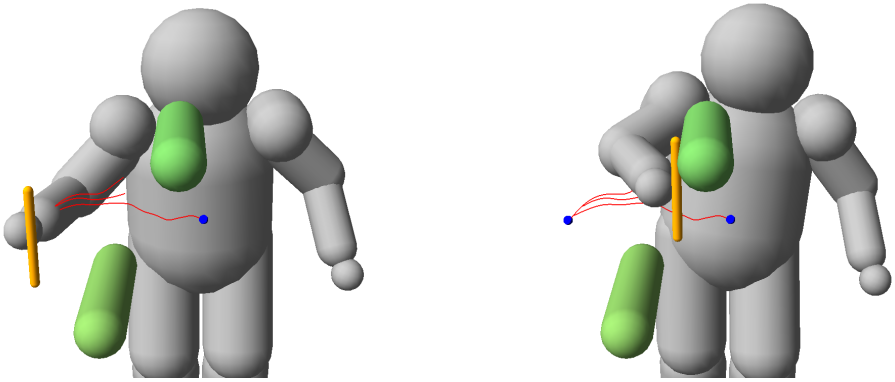

Figure 3.13: Depending on the spatial distribution of obstacles, different values of β are needed for a collision free trajectory. With $\beta = 0.25$ the target is reached, while for $\beta = 0.5$ and $\beta = 1$ a collision occurs.

Chapter 4

Motion Planning

This chapter introduces the field of *motion planning*. Although the purpose of motion planning is comparable to the previously described motion control methods, the creation of movements to fulfill a certain task, the perspective is different. While motion control can be seen as an optimization based approach, as elaborated in section 4.1, motion planning treats the problem more generally as the problem of finding an arbitrary path that is able to reach the goal. Section 4.2 formally defines the general motion planning problem. The problem can be solved by an exhaustive search over all possibilities with combinatorial motion planning, section 4.3. Since these methods are usually too costly for more complex robots, many practical applications rely on sampling-based motion planning, introduced in section 4.4. Difficulties and challenges of sampling-based planning are subject of the last section 4.5.

4.1 Planning as Optimization

As seen in chapter 2 and particularly in chapter 3, it is possible to adapt the motion generation process according to external influences. For example, with the relaxed motion control method it is possible to dynamically modify a trajectory to circumvent an obstacle. This capability might already be seen as a form of planning, since it requires the ability to change the course of actions to achieve a certain goal under changing conditions. The mentioned control methods achieve this change of actions by locally optimizing a combination of different cost functions, encoding conditions like obstacles or other motion constraints. This approach has much in common with the *potential field* motion planning technique, an early and widely used solution for planning robotic movements.

Potential field motion planning, as stated by Latombe (1991), is based on the general idea of treating the robot as a particle under the influence of an artificial potential field. The field induces an artificial force acting on the robot and by following the direction of this force, motion is generated. A typical choice to model the potential field is to use an attractive potential, pulling the robot towards the goal, and an repulsive potential, pushing the robot away from obstacles. The sum of both functions establishes the overall potential function and taking the gradient yields a force acting on the robot. An early implementation for articulated robots can be found in (Khatib, 1986). Since the fields are evaluated for every iteration step, the motion can instantaneously react to sudden changes in the potential, like movement of obstacles or even the appearance of new obstacles.

This way of motion planning acts as performing gradient descent on some potential function, which is the same as doing a local optimization of some cost function. Potential field motion planning is motion planning by optimization. If optimization is only performed locally, as for example the case for gradient descent optimization, it is subject to local minima that lead to undesired behaviors. Consider a point robot moving in the plane. Attracted by the goal, the robot enters a U-shaped obstacle, where it is repelled from the walls. At a certain point, the attracting and repelling potentials will form a local minima,

simultaneously pulling the robot towards the wall and pushing it away from the wall. No more motion is generated and the robot is trapped.

To overcome these problems, optimization has to be done on a global scale. Various planning approaches can be found in the literature that take a step into this direction by considering the entire trajectory for optimization, instead of doing just instantaneous optimizations. This task is sometimes called *trajectory planning* and is also treated in the field of *optimal control*.

Some planning methods optimize a trajectory directly in the state space. CHOMP (Ratliff et al., 2009) uses covariant gradient descent to optimize trajectories according to obstacle and smoothness costs. Although a naive initial guess can often be transformed into a valid solution, the convergence is sensitive to local minima. The method in (Kalakrishnan et al., 2011) relies on a stochastic optimization process instead that is more robust against local minima and allows non-differentiable and non-smooth cost functions. However, the process might still fail for an unfortunate initialization.

The work in (Zhang and Knoll, 1995) represents a trajectory by a sequence of sub-goals and associated smooth B-splines on joint level. The parameters of the splines are optimized regarding motion duration, trajectory length and bending energy, while smoothness is maintained. Although heuristics are introduced that dynamically insert additional sub-goals in case local minima or collisions are encountered, the initial sub-goals have to be pre-determined in a reasonable way. This is also the case for the approach in (Pan et al., 2011), where an initial trajectory is refined using a similar B-spline representation. Using a mixture of recursive spline sub-division and spline parameter optimization, the trajectory is smoothed while constraints are maintained even in cluttered environments.

There exist several methods that extend or incorporate the motion rate control approach of chapter 2. Building upon gradient projection redundancy resolution, see section 2.4, Nakamura (1991, chapter 5) consistently optimizes the redundant space motion over the whole trajectory. The task space trajectory however remains unchanged. An optimization approach that operates on the task trajectory directly is the elastic strips framework (Brock and Khatib, 2002). An initial trajectory is adapted according to external obstacle forces, pushing the trajectory away, and internal forces, ensuring the closure of the trajectory. The method presented in (Toussaint et al., 2007) operates on a task attractor dynamic system representation. Different criteria like trajectory length, smoothness, joint-limit avoidance and obstacle distance can be efficiently optimized, using analytic gradients in a back-propagation scheme. This way, sub-optimal and invalid trajectories can be adapted to varying conditions in changing environments.

The approach of treating planning as an optimization problem has some convincing advantages. First of all, the presented methods offer a fast solution for generating movements in, or close to real-time. In addition, the found solutions can be optimal with respect to various cost formulations, for instance the length of the trajectory or maintaining the largest possible distance to obstacles. With this two properties, planning speed and optimality regarding external conditions, it is possible to achieve some robustness towards disturbances like changing obstacle positions.

The described methods are able to solve many planning challenges, but limitations remain. A considerable drawback is their sensitivity to local minima of the cost functions. Although various attempts to overcome this problem are developed, the vulnerability to inappropriate initializations always remain. Most solutions are tailored to a specific planning challenge. A true global view that incorporates all motion possibilities is not achieved and completeness can not be guaranteed.

4.2 The General Motion Planning Problem

In the previous section, motion planning was treated as an optimization process. Planning in this context means to formulate a problem as a set of suitable cost functions and to provide a way to systematically adapt an initial guess inside a range of possible solutions. How to determine the right solution structure in a general way is not clear. The ability to solve a planning problem without a pre-structuring for the task at hand seems to require a higher level of planning.

In this section, *motion planning* is formulated in a more general way. By classifying different approaches according to their ability to solve this formal problem, it is possible to draw a separating line between planning as a mere adaption of a pre-determined problem structure and planning as a global, knowledge free search in the space of all possible solutions.

Following Latombe (1991), lets define the state space as the space containing all possible states of any moving entity. Although a motion planning problem can appear different in terms of geometry and kinematics, the state space provides a common ground on which the same motion planning algorithms can be applied. Assuming a robot $\mathcal{A}$ moving in a workspace $\mathcal{W}$, the state space is the configuration space $\mathcal{C}$ of all possible configurations q of $\mathcal{A}$. The subset of $\mathcal{W}$ that is occupied by $\mathcal{A}$ is denoted by $\mathcal{A}(q)$.

The goal of motion planning is to find a path for the robot that obeys all constraints given by its intrinsic properties and its environment. For example, constraints might arise from mechanical limitations like physical joint range limits. If these limits can be expressed as a certain set of configurations, they form a forbidden subset of the configuration space $\mathcal{L} \subset \mathcal{C}$. The most common case of constraints for motion planning are obstacles present in the workspace. They can be integrated into the configuration space representation by mapping workspace obstacle regions $\mathcal{B}_i$ to configuration space obstacles

$$\mathcal{CB}_i = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{B}_i \neq \emptyset\} \tag{4.1}$$

Given some joint space constraint definitions, one can define the free part of the configuration space $\mathcal{C}_{free}$. In the case of both joint limit and obstacle constraints, the free part contains all configurations not forbidden by limits or inside an obstacle

$$\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{L} \ \cup \ \mathcal{C} \backslash \bigcup_i \mathcal{CB}_i \tag{4.2}$$

Now, the general motion planning problem can be formally defined, sometimes referred to as the *piano mover's problem* (LaValle, 2006, chapter 4): The goal is to compute a free path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ between the initial configuration $q_{init} \in \mathcal{C}_{free}$ and the goal configuration $q_{goal} \in \mathcal{C}_{free}$ with $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$ or correctly report that such a path does not exist. Various approaches to solve this formal motion planning problem are presented in the remainder of this chapter.

4.3 Combinatorial Motion Planning

A whole family of planning algorithms, solving the formal motion planning problem given in the previous section, are summarized as combinatorial or exact motion planning methods. A characterizing property of these methods is their completeness with respect to the problem definition. Every solvable problem instance will actually be solved. If no solution is found, it proves that no solution exists.

A prerequisite of algorithmic completeness is a an exact and complete representation of the state space and its constraints, i.e. an explicit representation of C_{free}. Given such a

characterization of the free space, the approach is then to construct a roadmap, a graph that covers the connectedness of the free space. From every position, it should be trivial to reach the next roadmap node and once on the roadmap, every other node and every other position can be reached. Given a roadmap, solving subsequent planning queries is easy and consists mainly of a graph-search, for which efficient methods are available, like the A* algorithm (Hart et al., 1968). The most demanding part is to create the roadmap, especially for high-dimensional state spaces with complex constraints, like obstacle regions consisting of arbitrarly complicated geometries.

Construction of the roadmap often proceeds by computationally partitioning the space into simpler geometric sub-regions, like polygonal cells (e.g. triangulation into triangles, decomposition into cylinders) or more general semi-algebraic models. By checking the accessibility of neighboring regions, a complete roadmap capturing the adjacency of free regions can then be created. More details and example algorithms can be found in (LaValle, 2006, chapter 6).

The principle of explicitly constructing a free-space representation allows to completely solve the motion planning problem, a major advantage of these methods, but also accounts for their major weakness. To be able to consider all solution possibilities, the costly process of explicitly partitioning the space and enumerating all movement possibilities has to be shouldered. Depending on the complexity of the problem, this can quickly become intractable. For example, Canny's roadmap algorithm has a worst-case runtime exponential in the number of dimensions of the state space (LaValle, 2006, chapter 6).

More general, the computational complexity of the formal motion planning problem, as given in the previous section, solved by a complete algorithm, has been shown to be PSPACE-complete. The class PSPACE denotes problems solvable with a polynomial amount of storage space and PSPACE-hardness implies NP-hardness. However, this high complexity does not necessarily mean that complete motion planning is impossible, most applications are in fact simpler than the theoretical upper bounds and for many applications powerful heuristics can be found (LaValle, 2006, chapter 6). The next section introduces the field of sampling-based motion planning as techniques that are very successful by weakening the strict completeness guarantee.

4.4 Sampling-Based Motion Planning

The understanding that exhaustive combinatorial search inevitably leads to high complexity bounds triggered the development of a new methodology in motion planning. The key idea of *sampling-based motion planning* is to abandon an explicit and complete model of the free space and to rely on a systematic probing process instead. Using a so called *local planner*, sub-paths in the state space are created and checked against constraint violations. This constraint violation check is done with a suitable *simulation module*, for example testing for collisions between a geometric robot model and obstacles. The placement of the sub-paths is determined by a *sampling scheme*, with the objective of organizing the probing process and the resulting sub-paths in an adequate way. By repeating the sampling, local planning and constraint checking processes, an implicit representation of the free space can be build up in an incremental fashion. Since the sub-path construction is encapsulated into the local planner, the sampling scheme can be implemented without the knowledge of a concrete problem instance. Local planning can be treated as as a "black box", returning a valid path segment if possible. This allows the sampling-based planning approach to be relatively easily used in a broad range of applications.

It is important to note that sampling-based planning comes with the cost of loosing strong

completeness guarantees (LaValle, 2006, chapter 5). As described in the previous section, a complete algorithm has to either return a solution, or to correctly report the nonexistence of a solution. In sampling-based planning the placement of samples is often randomized and a complete coverage of the space can only be reached asymptoticly after infinite runtime. These methods are said to be *probabilistically complete*, the probability of finding a solution converges to one at infinity. They find a solution with increasing probability, but they can not proof the nonexistence of a path, a required property for completeness. If the samples are placed in a deterministic way, like on a regular grid, the planning algorithm can guarantee to find a path with respect to the given sampling density, it is *resolution complete*. But still, it gives no information about solutions that might exist below the resolution limit.

Despite this limitation, sampling-based motion planning is successfully used for many challenging problems. In particular, it is very practical for high-dimensional problems where complete methods are intractable. Here the drawback of only being probabilistically complete turns into a major advantage, because the search space can be greatly reduced and still most of the time solutions are found. This is enough for most real world applications, although there are some pathological cases that are hard to solve.

Due to the success of these methods there have been a lot of work on this topic. A comprehensive overview is given in (Choset et al., 2005; LaValle, 2006). The different methods basically vary the main components sampling scheme and local planner to implement different planning strategies and specialized heuristics for problem specific affordances.

4.4.1 Random Trees

The family of motion planning algorithms presented in this section are designed for the problem of finding one solution path for one given goal state in one specific situation. A situation is determined by the current state of the system, i.e. the start state, and the state of all relevant constraints, like the current placement of obstacles. This setup is often described as *single-query* motion planning. Essentially, most single-query methods try to incrementally grow a tree in the free space, from the initial state until the goal state is reached. In a nutshell, five main components can be identified (LaValle, 2006): (1) Initialization of the tree data structure, (2) selection of already present tree nodes for extension, (3) sampling of new target states, (4) local planning to extend the selected node and finally (5) integration of new nodes into the tree. How these components are implemented varies along the different algorithms.

A popular sampling-based random tree algorithm is the Rapidly Exploring Random Tree (RRT) (LaValle, 1998; LaValle and Kuffner, 2000). The implementation of the five main components are shown as pseudo code in algorithm 4.1. After tree initialization, the first step in each iteration is to pick a random sample x from a uniform state space distribution. Based on the sampled state, a node of the present tree is selected for extension. The RRT determines this node by searching the closest node p of the tree. Given the closest node, an attempt is made to extend the tree with the local planner, generating a sub-path originating at the selected closest tree node p towards the random sample x. The local planner executes the desired motion as long as obstacle- and all other constraints are satisfied. Finally, the end-state of the local tree extension is added to the tree. These steps are iterated until the goal state is reached or a maximum number of iterations respectively maximum execution time has passed.

The RRT algorithm has the characterizing property of biasing the search towards a fast exploration of the space. Due to the nearest neighbor node selection, the probability of selecting a certain tree node is proportional to the likelihood of a random sample to be placed into the the empty space surrounding it. The volume of a nodes empty space is

Algorithm 4.1 RRT

```
T = initialize tree
while not done and not time over do
    x = sample uniform random state
    p = nearest neighbor tree node of x
    n = extend tree node p towards x
    T = add new tree node n
end while
```

Algorithm 4.2 EST

```
T = init. tree, W = init. weights
while not done and not time over do
    p = select random tree node
          w.r.t. weight distribution W
    x = sample in neighborhood of p
    n = extend tree node p towards x
    T = add new tree node n
    W = update weight distribution
end while
```

given by its Voronoi region and therefore the probability of selecting a node is directly proportional to the Voronoi region. In early iterations, the tree reaches out fast into the space, because the Voronoi regions of nodes close to the outer boundary of the tree are larger. When they are no more over-proportional large Voronoi regions in later iterations, the tree continues to explore the entire space more densely, until it converges towards the sampling distribution. Figure 4.1 illustrates the RRT growing process.

A similar approach of building single-query random trees is the Expansive Space Tree (EST) algorithm (Hsu et al., 2002). It also incrementally builds a search tree rooted at the initial state towards a goal state, but the implementation and execution order of the individual tree growing components differs. As shown in algorithm 4.2, the first step is to select a tree node p for extension. The selection is random, but based on a distribution of weights w that are maintained for each node, determining the probability of how frequent a certain node is chosen. Once a node is selected, a target state x is sampled in its neighborhood, followed by a tree extension attempt of the local planner. In addition to adding the local planner end-state as a new tree node, the distribution of tree weights is updated to reflect the growth of the tree. The EST growing process is sketched in figure 4.2.

Different biasing behaviors can be implemented within this approach. Hsu et al. (2002) aims to achieve a uniform coverage of the space by setting the node weights inversely proportional to the number of nodes already present within a fixed neighborhood. Since the weight is higher for nodes with fewer neighbors, the growth is steered towards less explored regions of the state space. This concept is enhanced further with the Guided EST (Phillips et al., 2004). Here the node weight takes additional information into account, namely the number of outgoing edges of a node, how recently it was created and an estimated total cost to the goal. This results in a tree that prevents too many expansions of a single node, keeps the expansion on the frontier of the tree and focuses the search towards the goal.

The Utility-Guided Random Tree (Burns and Brock, 2007) proceeds similar to the EST algorithm: First a node is selected and then a target state is generated. The target sampling is differentiated into selection of a direction and selection of a distance. During all of these execution steps, a notion of the utility of a particular state space exploration is used in order to maximize the usefulness of every tree growing iteration. For instance, nodes also have associated weights, corresponding to their utility. To be able to estimate the expectation value for a certain exploration utility, an approximate probabilistic model of the state space is build in an incremental fashion by gathering all information about former explorations.

The described random tree planning methods all employ different heuristics to bias the search. Comparing the RRT algorithm on one side with the EST, Guided EST and Utility-Guided Tree on the other side, the main difference lies in the order in which the steps state sampling and node selection are done. While the RRT first creates a sample and

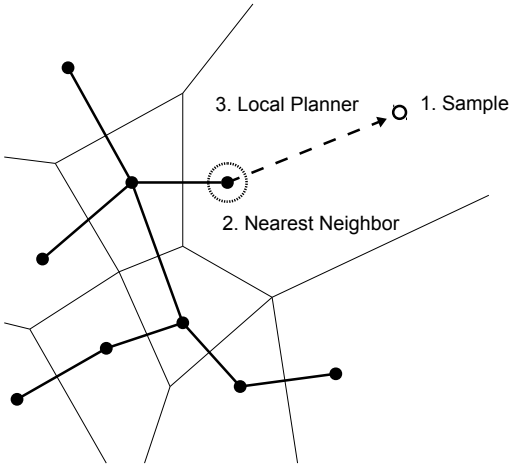

Figure 4.1: Illustration of the incremental RRT growing process. After sampling a random state (1), the nearest neighbor node is searched (2) and the local planner tries to extend the tree towards the sample (3). Note that the probability of selecting a tree node for extension is proportional to the Voronoi region surrounding it.

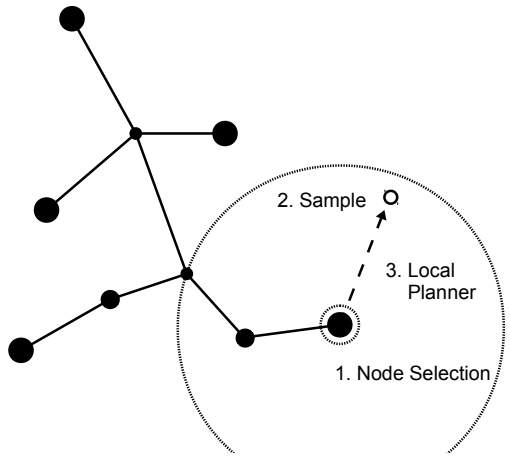

Figure 4.2: Incremental EST growing. First step is to select a tree node (1), then a state is sampled (2) and the tree is extended towards the sample (3). To bias the tree growth, nodes can have different selection probabilities (weights), shown by the size of the nodes here.

then uses the sample to select a node, the other algorithms first select a tree node and then create a sample in the neighborhood of the node. RRT node selection is determined by the distance based Voronoi bias and in order to incorporate further heuristics one has to change the distance metric or the sampling domain where the metric is defined. With the other EST-like family of algorithms there are no general limitations for heuristics, because the tree node selection is completely independent. The described methods already demonstrated some different biasing techniques and it is easy to incorporate others.

A characteristic property of most random tree methods is their capability to solve problems subject to nonholonomic constraints (refer to section 2.6). Such constraints are encapsulated in the local planner component and every path created automatically conforms to them. Caused by the additional constraints however, it can become more difficult to reach certain states and as a result, a uniform coverage of the state space is not as easily achieved.

4.4.2 Bi-Directional Trees

It is possible to extend the random tree approach to perform a bi-directional search by using two search trees (Kuffner and LaValle, 2000). One RRT is rooted at the start state and another RRT is rooted at the goal state. Both trees are simultaneously grown, the start tree forward and the goal tree backwards, and with a certain frequency direct connection attempts between them are done. Once the tree connection succeeds, a valid solution path is found. With this approach, the search can be greedily focused towards the goal and the number of necessary tree extensions can be reduced.

A bi-directional tree can be used if there is a unique goal state, where the backward tree can be rooted. For some applications, multiple goal states or even a continuous goal region are possible, and although multiple backward trees might be used in these cases, the performance gain decreases or might even vanish. Another problem are nonholonomic motion constraints. During the connection step, it is necessary to connect to already present tree

nodes i.e. to plan sub-paths that exactly reach a certain state. While trivial with holonomic systems, nonholonomic systems require to solve a boundary value problem, substantially complicating this step.

4.4.3 Roadmaps

Another popular approach in sampling-based motion planning is the use of roadmaps in a *multi-query* planning setup, as introduced by Kavraki et al. (1996) with the Probabilistic Roadmap algorithm (PRM). The idea is to incrementally construct a graph that covers the connectivity of the free space, very much like the combinatorial methods from section 4.3. Instead of an explicit state space representation however, random states are sampled and connected by the local planner. Unlike the random tree approach, each new sample should be connected to multiple nodes in order to form graph structures. If the sampling and connection process is repeated long enough, all nodes are eventually connected in one big graph that covers all areas of the free space. By modifying the way how states are connected to the graph and by adapting the sampling scheme, the probabilistic roadmap approach can be tuned for many applications. Some variants are described in (LaValle, 2006, chapter 5.6).

Roadmap methods handle the planning problem in a comprehensive pre-processing phase and assume that the setup does not change in later query solving phases. If constraints are modified, parts of the roadmap are no longer valid. It is possible to incorporate changing environments into the approach, see for example (van den Berg, 2007), but more effort has to be invested to maintain the roadmap by re-evaluating and repairing node connections.

Roadmaps do not naturally extend to problems with nonholonomic motion constraints. Many newly created states have to be connected to already present nodes, similar to the connection step of bi-directional trees and equally making it necessary to solve a boundary value problem.

4.5 Challenges for Sampling-Based Planning

Theoretically, sampling-based motion planners are able to solve every planning problem that can be formulated in terms of a state space that can be sampled and a local planner that is able to connect samples and to check for constraint violations. Due to the limitations of probabilistically completeness however, the running time of these planners is not bounded and it is not possible to prove the nonexistence of a solution.

Many complicated problems can indeed be solved, for example geometric puzzle games or industrial part assembly problems, where complex objects have to be moved in highly constrained environments. Another interesting application area is in computational biology, where sampling-based methods are used to find possible dock positions of proteins or to simulate complex protein folding processes. More details and further examples can be found in (LaValle, 2006, chapter 1). For these applications however, speed is of minor importance and runtimes easily reach minutes or hours.

To get an insight into the reasons for such long runtimes, the high dimensionality of certain state spaces and the complex structure of constraints in this spaces have to be considered. It is known that such conditions often lead to computational problems, often described as as the *curse of dimensionality*, e.g. see (Bishop, 2006). One problem for instance is that the number of samples, necessary to cover a space with a certain resolution, grows exponentially with the number of dimensions. As a consequence, an exhausting sampling-based search would also need an exponentially increasing number of samples.

But even in lower dimensional spaces, the structure of constraints can hinder the sampling progress. Two particularly difficult constraint conditions can be repeatedly found in the

literature. One are *narrow gaps* of free space that have to be discovered during sampling. The probability to find a path passing through it decreases with the size of the gap. *Narrow passages* or *corridors* worsen this problem, since they require to repeatedly place samples into small fractions of free space while inside a corridor. Another source of difficulties are *motion constraints* arising from nonholonomic systems, as for example the case for planning with dynamics. Although the random tree methods are capable of solving these problems, they are usually not very efficient. State spaces are high dimensional, since they have to incorporate velocities or derivatives of higher order. Also, it is difficult to achieve a uniform coverage of the state. In the following, some approaches to overcome these problems by adapting to specific system characteristics and constraints are presented.

There are several variants and enhancements of the basic RRT scheme. Cheng and LaValle (2001) address differentially constrained and high dimensional systems, where a naive euclidean metric results in a poor performance characterized by an over-exploration of certain nodes. To avoid this problem, the underlying metric itself is not changed, but the original Voronoi bias is weakened by allowing to skip tree extensions, as determined by a constraint violation frequency heuristic. Paths with a higher collision probability can thus be avoided. Also the algorithm takes care that identical extensions are not repeated.

Yershova et al. (2005) introduce a refinement called Dynamic Domain RRT. The sampling scheme is enhanced to incorporate the distribution of configuration space obstacles into the Voronoi bias, motivated by the observation that the exploration speed can dramatically decrease if many frontier nodes of the tree are in close proximity to obstacles, as it may happen if large portions of the space are blocked. The idea is to reduce the probability of extension for nodes that are near to boundaries of the free space by individually adapting their sampling domain. If this is done in a dynamic way, it is possible to automatically balance between exploration and refinement behavior and to better adapt to the local shape of the free space.

The algorithm in (Ladd and Kavraki, 2005) also aims for a balanced search by estimating the current coverage of the state space with a non-uniform cell partition. Based on a density estimate on this space subdivision, the exploration process is guided towards less covered regions. Interior-Exterior Cell Exploration (KPIECE) (Sucan and Kavraki, 2009a) estimates the state space coverage with a discrete grid-based cell representation. In contrast to the previous method, this representation is defined on a different lower dimensional space than the state space, in order to avoid the curse of dimensionality. An exploration strategy based on the distribution of projected samples towards the grid cells is developed, aiming to equalize the coverage of cells. The exploration heuristic also differentiates between interior and exterior cells to bias the search towards cells on the frontier of the tree. The definition of a suitable grid presentation and state projection is problem specific, but the authors claim that finding a random projection leading to a good performance is easy.

Other approaches rely on machine learning techniques to utilize information gathered during the exploration. The work in (Li and Bekris, 2010) is motivated by the observation that the exploration of random tree planners can be heavily biased towards a specific direction due to the presence of dynamics and underactuation. To counteract this effect, the algorithm tries to learn the observed bias with principal component analysis in an offline computation step and utilizes this knowledge in subsequent planning runs. By modifying the RRT sampling to prefer directions that were previously underrepresented due to the undesired bias, a more balanced exploration of the state space can be achieved.

The method presented in (Dalibard and Laumond, 2011) tries to learn the structure of the free space and to focus the exploration along free directions. If the free space can be described by a local submanifold of the full state space, biasing the search according to this submanifold can significantly accelerate the exploration process. During sampling, principal

component analysis is used to characterize the local distribution of previously successful movements and random direction samples are changed in order to favor directions where the local variance is high. This way, the exploration proceeds with higher probability in locally free directions and sampling is guided to follow narrow passages.

Chapter 5

Hybrid Motion Planning

The previous chapters introduced the areas of motion control and motion planning. Various control and planning approaches have been presented and their specific strengths and weaknesses were highlighted. Resting on this basis, the main contribution of this thesis can be developed, the concept of *hybrid motion planning*. Central idea is the integration of local control and global planning into a hybrid approach, as presented in section 5.1. A detailed description of the concept is given in section 5.2. Section 5.3 then discusses how global planning can be implemented with sampling-based random-tree methods and compares two different strategies in a simple case study. The question if planning can be guaranteed to be complete is subject of section 5.4.

5.1 Task Level Planning

Motion control is capable to efficiently produce movements, as seen in chapters 2 and 3. Tasks can be encoded with flexible motion primitives and optimization techniques can be incorporated that minimize different cost functions instantaneously, e.g. with the gradient projection approach, or over a whole trajectory, e.g. with trajectory planning. These methods are capable of solving many problems, but by treating motion generation mostly as an optimization problem, they can be hindered by local minima. Although some problems can be avoided with problem specific modeling to cope with specific cost structures and constraints, this is not possible in a generally applicable way.

In contrast, motion planning does not have these problems. As discussed in chapter 4, the generation of motions is handled as a search problem that can be solved by evaluating all possibilities. Motion planning methods are global approaches, not subject to the limitations of motion control. However, other difficulties arise. There is the demanding computational complexity of combinatorial planning and likewise, sampling-based approaches suffer from high dimensional state spaces and complex constraints.

Our proposed hybrid motion planning framework utilizes concepts from both motion control and motion planning in order to benefit from their individual strengths and to address their weaknesses. Key idea is to shift planning to a task specific representation. This way, the problem can be decoupled into two processes. The task representation is searched on global level with motion planning methods, systematically specifying task targets, while motion control techniques are used to generate local movement trajectories corresponding to these task targets.

Figure 5.1 schematically illustrates the idea. The state space fully describes the state of the system of interest, including its constraints. Assuming we are dealing with a robotic manipulator, the state space encodes the state of its joints and constraints could be joint-limits and obstacles. Most motion planning approaches directly operate on this level. With hybrid motion planning however, a second task space is defined. This representation is chosen according to the actual intention behind planning, the task that has to be achieved. For example, assuming the manipulator has to reach for an object, the spatial position of the

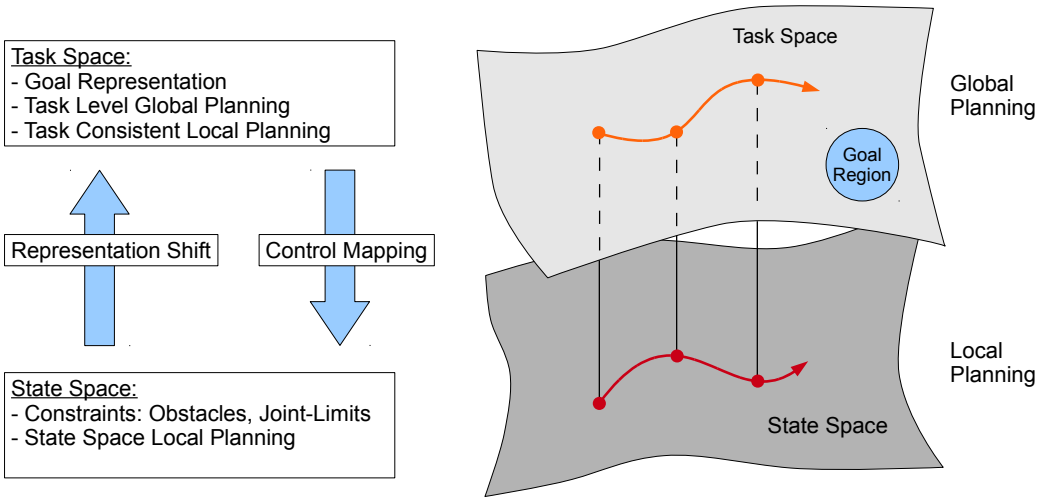

Figure 5.1: The motion planning problem is decomposed into global search on a task representation and local control in the state space.

hand relative to the object could be the task space. The goal of planning can then be given in the task space as well, a desired relative position between hand and object. Utilizing the representation shift from the state to the task space, global planning operates now directly on task level by specifying task targets. For the manipulator reaching for an object, these task target would be positions of the hand relative to the object. Local planning then generates a movement towards the target, by employing a motion control component that creates a full state space trajectory. In the case of our manipulator, a joint-level trajectory is generated, driving the hand towards the desired target. Motion control establishes a mapping between task space and state space motions. This process is now repeated until the goal is reached, until the hand reaches the desired position.

Adding a search layer on top of motion control enables our approach to find solutions with a global scope, counteracting the limitations of local control. In general, global search with motion planning methods is costly, but by limiting the search to the task space, the dimensionality can be reduced. The effort of searching a space can decrease dramatically with every dimension less. Also, focusing on the task representation fosters progress towards the actual goal at hand stronger than an uninformed search in the state space. The search is biased towards states that have a better chance to achieve the task. If multiple states are satisfying the goal condition, the planning process automatically discovers which states are feasible in the current situation. It is not necessary to determine the set of feasible goal states beforehand, another advantage of the proposed method.

Local planning of state space trajectories can exploit the optimization abilities of motion control methods. Some constraints can be handled locally by employing joint-limit or obstacle avoidance optimization for example. This way, using motion control does not only support motion planning by establishing the representation shift from state to task space, but also by helping to avoid constraints. The better this constraint avoidance works, the less global search effort remains for global planning. If local planning succeeds in circumventing constraints more often, the resulting trajectories are longer and the task targets can be further apart, the space can be searched with a coarser resolution.

5.2 Formulation

The hybrid motion planning concept is now transfered into a mathematical formulation. Section 4.2 elaborates the notion of the free configuration space $\mathcal{C}_{free}$ as the region of the configuration space not forbidden by constraints. Only states that lie in this region are allowed and a valid solution to any motion planning problem has to restrain to it.

The key idea of hybrid motion planning is to define a second space in addition to the configuration space, in which the task to be solved is represented. This space is referred to as the task space $\mathcal{X}$. It can be of arbitrary topology and dimension, the only condition is that for every configuration $q \in \mathcal{C}$ there exist a corresponding task space representation $x \in \mathcal{X}$, there is a mapping $k : \mathcal{C} \to \mathcal{X}$. The laws of motion, determining the evolution of states, are given by two state transition functions:

$$\dot{x} = f(x, u, t) \tag{5.1}$$

$$\dot{q} = g(q, \dot{x}) \tag{5.2}$$

Given a current task space position $x \in \mathcal{X}$ and a task control input $u \in \mathcal{X}$, equation 5.1 maps to task transition velocities $\dot{x} \in T_x(\mathcal{X})$, in dependency of time t. $T_x(\mathcal{X})$ denotes the tangent space of $\mathcal{X}$ at position x. The second equation 5.2 describes the transition of configuration states. Given a current configuration $q \in \mathcal{C}$ and desired task transition $\dot{x} \in T_x(\mathcal{X})$ as input, the configuration space velocity $\dot{q} \in T_q(\mathcal{C})$ is computed.

With this definitions, a motion trajectory can be generated as follows. Function 5.1 returns a task velocity $\dot{x}$, which is fed into function 5.2 to get a configuration velocity $\dot{q}$. A configuration space trajectory is then generated by integration of the configuration velocities over time. The corresponding task space trajectory is uniquely given with the task space mapping $x = k(q)$. What has to be known is the initial start condition, that is a start configuration q_{init} and a start task position x_{init}, and a sequence of control inputs u for every time t. This sequence of control inputs is exactly what is searched during planning.

In analogy to the piano movers problem formulation, section 4.2, the hybrid motion planning problem can now be formulated in a similar way. The path is represented in two spaces, task space $\mathcal{X}$ and configuration space $\mathcal{C}$, every point of the path is a tuple $p \in (\mathcal{X}, \mathcal{C})$. Find a sequence of task space control inputs $u(t)$ over time t such that the resulting path with initial condition $p_{init} = (x_{init}, q_{init})$ and final condition $p_{goal} = (x_{goal}, q_{goal})$ is collision free, that is the configurations of the entire path lie in $\mathcal{C}_{free}$. With the path being $\tau : [0, t] \to (\mathcal{X}, \mathcal{C}_{free})$, $\tau(0) = p_{init}$ and $\tau(t) = p_{goal}$. While the final task position x_{goal} should be in a given goal region, satisfying $x_{goal} \in \mathcal{X}_{goal}$, a goal configuration q_{goal} is not set in the goal description and is left undefined. If such a valid solution path does not exist, it should be correctly reported.

Global planning drives the overall planning process by specifying task control inputs. Appropriate task space transitions are created, based on this input, and these task space transitions are in turn resulting in configuration space transitions. Local planning can happen in two places now.

If the system is redundant, a desired task space transition do not uniquely determine a configuration space transition. There is some freedom left in choosing a particular configuration trajectory. Local planning can exploit redundancy according to various criteria, for example by employing local optimization techniques. Chapter 6 presents a hybrid planning approach with exploitation of redundancy in greater detail.

Independent of the presence of redundancy, the configuration space transition does not have to exactly follow the desired task space transition. Local planning can also modify the

task trajectory. In chapter 8, a control method is utilized that optimizes the task trajectory in order to circumvent obstacles.

5.3 Task Space Sampling

Hybrid Motion Planning limits the globally searched space to a lower dimensional, problem specific task space. Although many ways to perform this global search are possible, an interesting approach is to use sampling-based motion planning, reviewed in section 4.4. These methods have the advantage of being able to solve complex problems, while being relatively easy to implement. The representation shift towards task space planning is accomplished by using motion control methods as local planning components. Since these methods can posses nonholonomic motion constraints, as shown in section 2.6, the single-query random-tree approach fits particularly well into the approach. The bi-directional tree method or multi-query roadmaps would require a specific treatment of these constraints, unnecessarily complicating the implementation and preventing a general solution independent of the actual motion model.

This section investigates the question, which random-tree sampling strategy fits best into the hybrid planning context. Without further knowledge, a reasonable approach is to aim towards a rapid exploration of the global task space as a fast way to discover movement alternatives on a course scale. Actual small scale motion generation is left to local planning, exploiting local properties of the space. This approach is taken by a greedy task space exploration strategy, described in section 5.3.3, transferring a known random-tree algorithm to the new task representation.

Focusing on the task space only and completely ignoring the state space local planning process however might not be always justified. Since motion control techniques can exhibit nonholonomic behavior, identical task positions can be reached with different configurations. If local planning always picks the right configuration, there is no problem in ignoring alternative states during sampling the state space. If this can not be guaranteed, as the limited scope of local methods suggests, it makes sense to actively explore these alternatives by incorporating state space information into global sampling. A novel sampling strategy simultaneously invloving the task and the configuration space is developed in section 5.3.4.

The behavior of these two sampling approaches is compared in a case study involving a simple redundant robot model. For this a measure of the sampling progress is needed, estimating how well the search tree covers the space, or more precisely the two spaces used in the hybrid planning framework. Such a measure is given by the *dispersion*, described in section 5.3.1. The concept of dispersion is well known in the field of motion planning and is for example used to achieve a greedy dispersion reducing sampling, as explained in section 5.3.2.

5.3.1 Dispersion Estimation

Dispersion is a measure of how well a space is covered with samples. It measures the sampling density of points by finding the largest completely empty area remaining in the space. A formal definition can be found in (Lindemann and LaValle, 2004): The dispersion δ of a point set $P \subset X$ with respect to the space X and metric ρ on X is defined as

$$\delta(P, \rho) = \sup_{q \in X} \min_{p \in P} \rho(q, p) \tag{5.3}$$

With other words, for each sample q from X the nearest neighbor p from the point set P is determined and the largest of these nearest neighbor distances corresponds to the

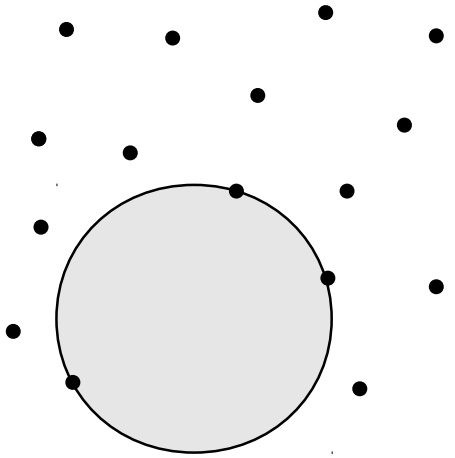

Figure 5.2: Dispersion measures the largest empty region of a space, not covered by points from a finite set. With the euclidean metric, this corresponds to the largest empty ball that can be placed into the space.

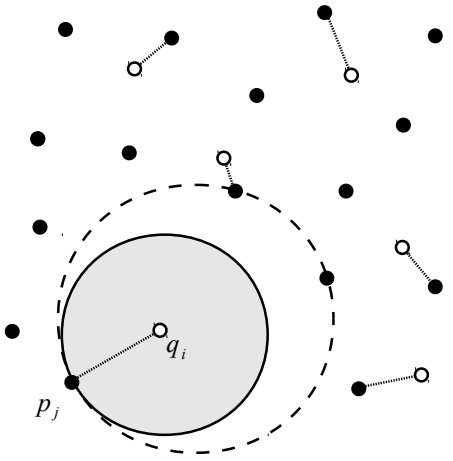

Figure 5.3: To estimate the disperision in practice, the sample-space is approximated by a finite set of points q. After finding the nearest neighbor p of each q, the largest nearest neighbor distance approximates the true dispersion.

dispersion. To get the exact dispersion measure, it would be necessary to cover X with infinitely many points. For a practical approximation of dispersion, a finite set of state space sample points is chosen. Figure 5.2 illustrates the concept of dispersion. The approximate dispersion estimation process is illustrated in figure 5.3.

5.3.2 Dispersion Reduction Sampling

The notion of dispersion is a useful concept for sampling-based planning, since it can provide information about how well the state space is covered and where samples are missing. In order to estimate the dispersion for sampling-based planning, the point set P corresponds to the set of all tree nodes and the sample space X to the state space. During computation of the dispersion according to equation 5.3, a set of nearest neighbor tree nodes $p_j \in P$ is identified, each with an associated distance towards the nearest sample point $d_j = \rho(p_j, q_i)$. The largest distance d_{max} denotes the global dispersion and the corresponding tree node p_{max} lies in the vicinity of the largest uncovered area.

The RRT algorithm, as introduced in section 4.4.1, can be interpreted as a method that systematically lowers the state space dispersion. In each iteration, tree nodes are extended with a probability proportional to their Voronoi region and as a result, the growth is strongly biased towards less covered areas of the space. The largest Voronoi region corresponds to the region with the largest nearest neighbor distance and exploring this region results in decreasing the overall dispersion. The RRT does not always explore the largest Voronoi region but puts a greater bias on it, thus driving the search towards decreasing the dispersion in a probabilistic way.

A variant of the RRT algorithm explicitly lowering the dispersion of the tree as much as possible is presented in (Lindemann and LaValle, 2004). In each iteration, an attempt is made to identify and explore the largest empty region by finding the tree node p_{max}. Since finding the true global dispersion is intractable, the approximative approach using a finite set of sample points is used, as shown in figure 5.3. Once p_{max} is found, an

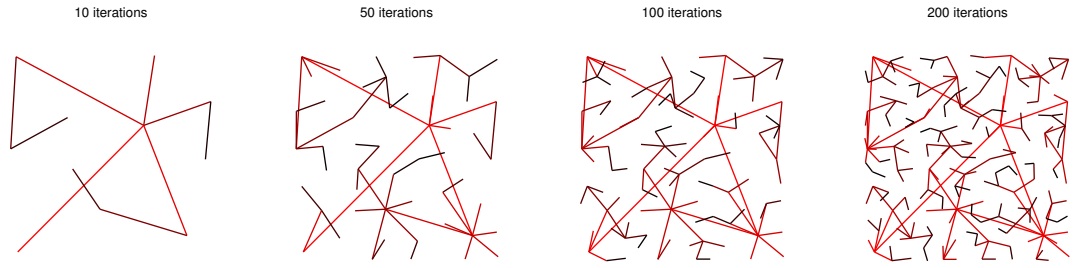

Figure 5.4: Incremental growth of a Dispersion Reducing RRT, after 10, 50, 100 and 200 iterations.

exploration step towards the associated sample q is done, which can be guaranteed to lower the global dispersion. This behavior results in a much more regular tree structure and a faster exploration, as can be seen in figure 5.4.

5.3.3 Task Space Dispersion Reduction

The idea of the dispersion reducing RRT (Lindemann and LaValle, 2004) can be transfered to the hybrid planning framework. The sampled space is now the task space and local planning employs a motion control method to generate state space trajectories. How the Task Space Dispersion Reducing RRT (DRTask) proceeds is shown in algorithm 5.1.

Instead of generating a single random sample, like the original RRT, a number K of uniform random samples from the task space x_k are drawn. For every sample, the nearest neighbor p_k in the already present tree is computed, using a suitable task space metric $\rho(x, y)$ with $x, y \in \mathcal{X}$. Every sample x_k and corresponding tree node p_k are stored in a list of candidate explorations, together with the actual computed closest distance d_k.

In the next step, the candidate list is sorted according to decreasing distance. This is a way of ordering the possible tree extensions by their expected impact in reducing the dispersion. Afterwards, the tree extension resulting in the largest dispersion reduction is first in the list, followed by the second best tree extension etc. The sorted list is then iterated and the local planner generates configuration space trajectories for each element, starting at the closest tree node p_k and extending towards the target sample x_k. If the local planner succeeds in creating a local trajectory that reaches the target sample, the new trajectory is added as a branch to the tree and the iteration of the candidate list is terminated.

The algorithm as presented is operating in an exploration-only mode, no planning goal is given and thus no test for goal reaching is included. Instead, tree growing is just iterated N times. Extending the algorithm to perform proper planning is easy, but not necessary for the present evaluation of exploration behavior.

5.3.4 Simultaneous Task and Configuration Space Dispersion Reduction

In this section, an alternative exploration strategy is developed. Instead of an exclusive focus on task space dispersion reduction, like the previously described DRTask algorithm, the new strategy incorporates the dispersion of the underlying state space as well. Sampling should be done in a way that simultaneously reduces the dispersion in both spaces.

Since both spaces can be different in terms of dimensionality and topology, the dispersion estimate differs in both spaces. Moreover, it can be assumed that in most cases, the maximum dispersion tree nodes are not identical. To reduce the task space dispersion, a different tree node has to be extended than to reduce the configuration space dispersion. In

Algorithm 5.1 Task Space Dispersion Reducing RRT (DRTask)

```
tree with nodes p_i = (x_i, q_i) ∈ T = (X, C)
metric ρ(x, y), x, y ∈ X
for i = 1...N do
    for k = 1...K do
        x_k = sample from X
        p_k = nearest tree neighbor to sample x_k with distance d_k = min_{p∈T} ρ(x_k, p)
        L_N = add tuple (p_k, x_k, d_k) to candidate node list L_N
    end for
    SL_N = sort list L_N by decreasing distance d_k

    for k = 1...K do
        (p_k, x_k) = tree node p_k and target sample x_k form sorted list SL_N
        extend p_k towards target sample x_k
        if extension successful then
            add new branch from p_k to x_k
            break for
        end if
    end for
end for
```

order to resolve this conflict, an exploration strategy is proposed that prioritizes dispersion reduction. First step is to select the tree node with the largest configuration space dispersion, followed by sampling the task space target with the largest task space dispersion. Thus, reduction of configuration space dispersion has a higher priority, determining where to search, while task space dispersion guides the sampling and chooses in which direction the search proceeds to achieve the largest reduction impact. Tree node selection operates with global scope, while determination of the extension target is limited to a local neighborhood around the selected node.

Algorithm 5.2 schematically shows the new sampling strategy, named Simultaneous Dispersion Reducing Tree (DRSim). The dispersion estimation technique from (Lindemann and LaValle, 2004) is used to identify the tree node with the potentially largest dispersion reduction effect in the configuration space. K sample points x_k are drawn from the configuration space $\mathcal{C}$, their nearest neighbor nodes p_k in the already present tree are determined, based on a suitable configuration space metric $\varphi(q, p)$ with $q, p \in \mathcal{C}$ and each tree node is put into a list L_N of candidate nodes, together with the associated distance d_k. The list is sorted according to decreasing distances, now containing the tree nodes ordered by their estimated impact on the configuration space dispersion.

Next, the list is iterated, and for each node p_k a task space target x_t is chosen. To find the target with the greatest impact in reducing the task space dispersion, L samples x_l are created in the task space $\mathcal{X}$ within a certain neighborhood of p_k and the nearest tree neighbor of each sample is searched, using a task space metric $\rho(x, y)$ with $x, y \in \mathcal{X}$. Because the sampling range is limited, the set of possible nearest tree neighbors is also limited to the subset of direct relatives of the node. After L candidate targets are created and stored in the list L_S, the target x_t with the largest nearest neighbor distance is found by sorting the list. Identifying the exploration direction with the potentially largest impact of reducing the task space dispersion, the selected tree node p_k is extended towards the target x_t. If the extension is successful and not hindered by constraints, the new branch is added to the tree

Algorithm 5.2 Simultaneous Dispersion Reducing Tree (DRSim)

```
tree with nodes p_i = (x_i, q_i) ∈ T = (X, C)
metric ρ(x, y), x, y ∈ X, metric φ(q, p), q, p ∈ C
for i = 1...N do
    for k = 1...K do
        q_k = sample from C
        p_k = nearest tree neighbor to sample q_k with distance d_k = min_{p∈T} φ(q_k, p)
        L_N = insert pair (p_k, d_k) into candidate nodes list L_N
    end for
    SL_N = sort list L_N by decreasing distances d_k

    for k = 1...K do
        p_k = node k from sorted list SL_N
        for l = 1...L do
            x_l = sample from X in neighborhood of node p_k
            d_l = nearest neighbor of childs c of node p_k, distance d_l = min_c ρ(x_l, c)
            L_S = insert pair (x_l, d_l) into candidate samples list L_S
        end for
        SL_S = sort list L_S by decreasing distances d_l

        x_t = first node from sorted list SL_S
        extend p_k towards target x_t
        if extension successful then
            add new branch from p_k to x_t
            break for
        end if
    end for
end for
```

and the iteration of candidate nodes is terminated. If not successful, the target sampling and selection process is repeated for the next candidate tree node in the sorted list SL_N.

In contrast to the previous RRT based approaches, the new exploration strategy decouples the node selection and target sampling steps, proceeding in a similar way as the EST (Hsu et al., 2002) inspired methods, described in section 4.4.1. Random node selection according to node-wise coverage estimation is replaced by the selection of the node with the highest configuration space dispersion. While the dispersion measure can be interpreted as another way to estimate the coverage, node selection is much stronger biased to greedily reduce the dispersion in every iteration. While the EST randomly samples target points, the new sampling approach greedily reduces the task space dispersion by selecting the target with the largest anticipated dispersion reduction effect. In combination, both node selection and target sampling lead to greedy and simultaneous reduction of task- and configuration space dispersion.

5.3.5 Case Study: Simple Redundant Robot Model

To evaluate the behavior of the two proposed sampling strategies in the context of the hybrid motion planning framework, a simple setup is used. A robot is modeled as a redundant, multi-link planar manipulator, with a two dimensional task space given by the end-effector position. The singularity robust motion rate control and redundancy resolution methods of

Algorithm 5.3 Motion Control Local Planner

extend tree node $p = (x, q) \in (\mathcal{X}, \mathcal{C})$ towards target $u \in \mathcal{X}$
while not max. control time t_{max} exceeded **and not** constraints violated **do**
    get desired task velocities $\dot{x}$ from current position x and target u
    compute joint velocities $\dot{q}$ given configuration q and task velocities $\dot{x}$
    update current task position x with $\dot{x}$
    update current joint position q with $\dot{q}$
    check updated q for constraint violations
end while

chapter 2 are used as local planning components. Algorithm 5.3 shows the local planning process and how a tree node p is extended towards a task target u. In dependency on the current task position x, a desired task velocity $\dot{x}$ is computed (task transition equation 5.1), using the second order attractor dynamic (section 2.5.1). Based on the task velocity $\dot{x}$ and current configuration q, the motion control methods are employed to determine the configuration velocity $\dot{q}$ (configuration transition equation 5.2).

According to the hybrid planning formulation, planning is separated into task space and state space planning. Here, the state space is given by the robots configuration space, spanned by the number of joints, and the task space state is defined to be the two dimensional end-effector position. Local planning in the task space performs a straight target directed movement, while local planning in the state space utilizes the redundancy to accomplish secondary motion objectives with local optimization. Three conditions for redundant space movements are used here. One is the pure singularity robust control approach without explicit redundant space handling, section 2.3. The other two conditions are joint-limit avoidance and combined joint-limit and external obstacle avoidance. Corresponding cost functions can be found in section 2.4.

The two proposed task space sampling strategies, DRTask and DRSim, are compared with respect to their ability to explore. Since both algorithms aim to greedily reduce the dispersion in every iteration, their exploration success is quantified by measuring the remaining dispersion after each iteration. Of primary concern is the exploration of the task space, the space where the global search is done and the planning goal is defined. The current task space dispersion is estimated with a fixed grid of points, covering the reachable task space, as shown in figure 5.5. Although not actively sampled, the exploration also proceeds in the underlying configuration space and to get an understanding of the coverage of this space, the remaining dispersion is also measured in the configuration space. Similar to the fixed task space estimation points, a regular grid is placed into the configuration space.

5.3.6 Results

2D Task Space, 3D Configuration Space, No Constraints

A first evaluation is done for a simple unconstrained setup. A three-joint manipulator can be freely moved by sampling targets in the two dimensional position task space. Local planning is done with singularity robust pseudoinverse motion control without any further secondary motion objectives.

An example task space exploration tree produced with the DRTask sampling approach can be seen in figure 5.6 and the corresponding configuration space tree is shown in figure 5.7. The disk-shaped reachable workspace is indeed rapidly explored and the resulting tree

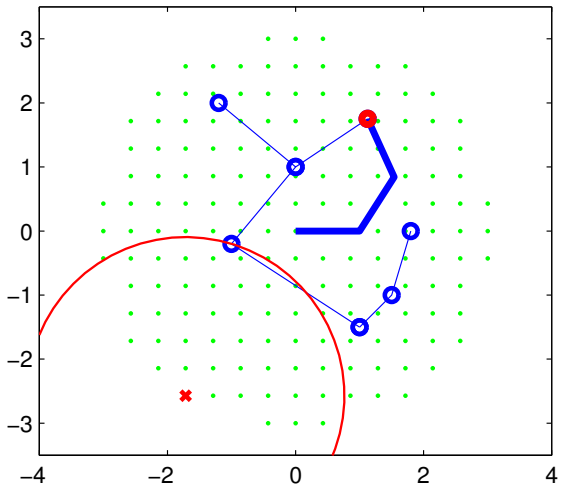

Figure 5.5: A 3-joint manipulator with a 2D end-effector position task space. The dispersion of the the tree nodes (blue) is estimated with respect to the sample points arranged in a regular grid (green), covering the reachable task space. The current dispersion corresponds to the largest nearest neighbor distance (red).

nodes cover the space quite regular. The distribution of tree nodes in the configuration space is not as easily interpretable, but one apparent impression is that the space is not regularly explored at all. There seems to be a manifold of densely packed nodes and a large empty space surrounding it.

Example results for a run of the DRSim algorithm are shown in figure 5.8 and 5.9, showing the task space and the configuration space trees, respectively. The task space region reached by the tree is quite similar to the reached area of the DRTask tree, although the exploration is much more chaotic and exhibits no regular structure. Looking at the covered configuration space however, it seems like the DRSim tree reaches a larger portion of the space.

Lets examine the space coverage quantitatively by looking at the dispersion measure in the task and the configuration space. The evolution of these quantities over 500 iterations and averaged over 30 runs is shown in figure 5.10 for the DRTask algorithm. The aim of reducing the task space dispersion in every step is achieved with a very fast initial reduction phase, characterizing a rapid exploration behavior. The corresponding configuration space dispersion is also reduced, but the reduction trend nearly vanishes after some first initial iterations and thus the dispersion in this space stays high over the whole interval.

The dispersion reduction trend for the DRSim algorithm is shown in figure 5.11. Looking at the task space dispersion curve, it is visible that the dispersion is also reduced in every iteration, but slower as with the DRTask algorithm. Also the mean dispersion after 500 iterations is slightly greater. In the configuration space, the dispersion behaves differently. It is decreasing over the whole interval, not showing the same saturation effect as with DRTask. At the end of the interval, the configuration space dispersion is on average lower with the DRSim algorithm than with the DRTask algorithm. This can also be seen in table 5.1, where the final values of the dispersion measure after 500 iterations are summarized.

Lets see how the dispersion behaves when another local planning method is used. In figures 5.12 and 5.13 the dispersion history for both algorithms is plotted, but this time the joint-limit avoiding redundancy resolution control method is used. The DRTask exploration strategy shows a very similar behavior. While the task dispersion is rapidly reduced, the configuration dispersion lowers only initially and then stays at a high level. Using the DRSim method does reproduce the outcome of the previous trial. Again is the dispersion in both spaces reduced, but in contrast to the previous trial, the configuration space dispersion is not constantly lowered but converges to a comparable high value than with DRTask. Overall, both exploration techniques perform similar in reducing dispersion, as can again be verified in table 5.1, only the convergence speed of DRTask is better.

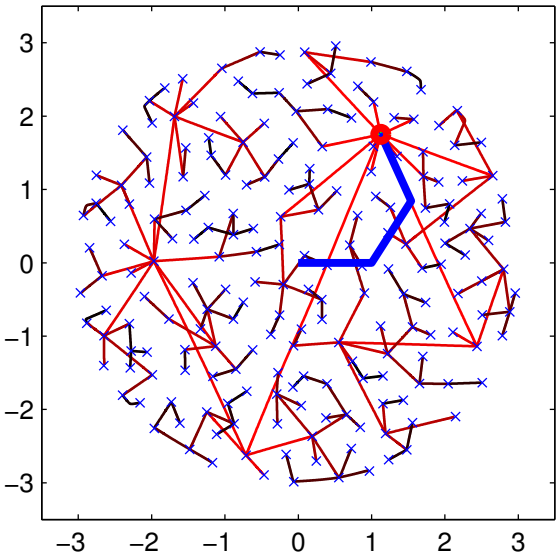

Figure 5.6: Task Space Dispersion Reducing Tree (DRTask) after 200 iterations, using singularity robust control and the 2D end-effector position task space.

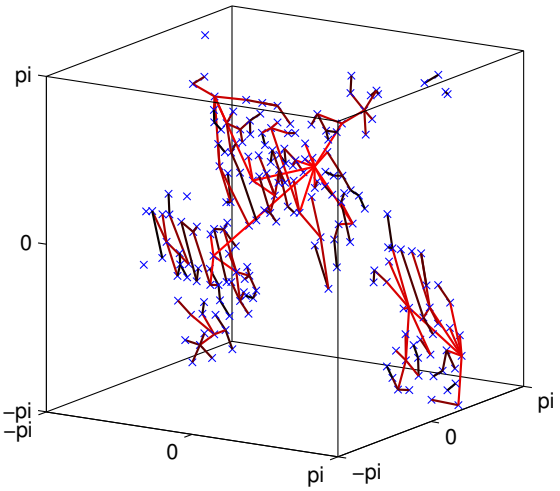

Figure 5.7: The corresponding tree in the 3D configuration space, spanned by the joints of the manipulator.

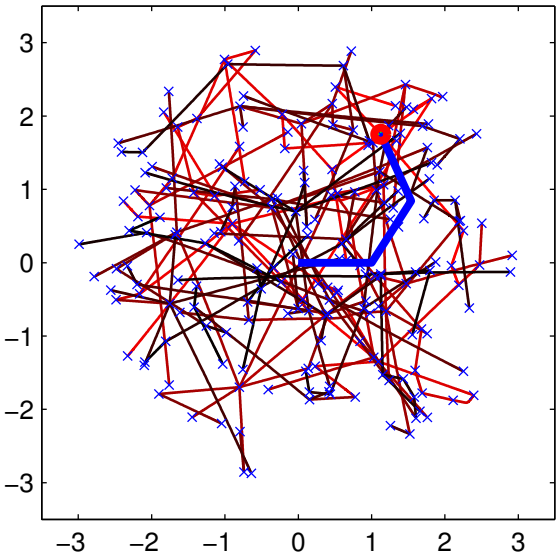

Figure 5.8: Simultaneous Task and Configuration Space Dispersion Reducing Tree (DRSim) after 200 iterations, using singularity robust control local planning and the 2D end-effector position task space.

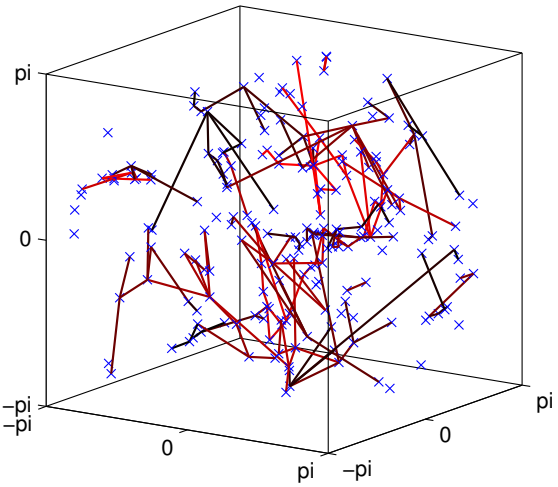

Figure 5.9: The corresponding tree in the 3D configuration space, spanned by the joints of the manipulator.

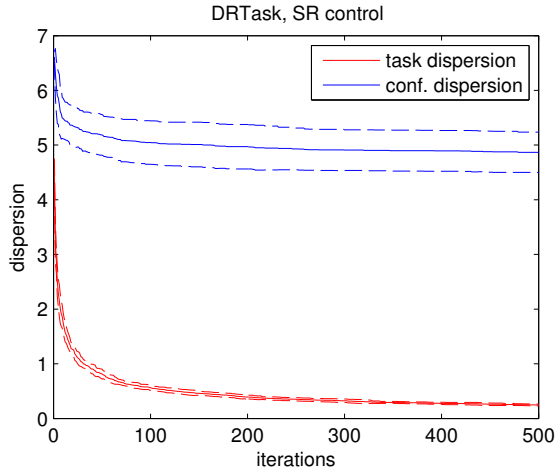

Figure 5.10: Evolution of dispersion measures, task (red) and configuration space (blue), during 500 iterations and averaged over 30 runs. Task Space Dispersion Reducing Tree (DRTask) sampling with singularity robust pseudoinverse control local planning.

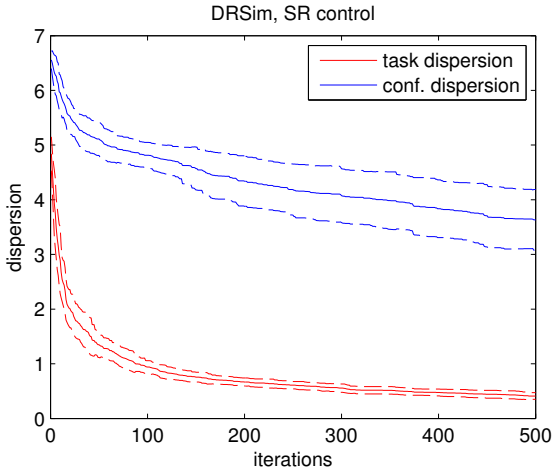

Figure 5.11: Evolution of dispersion measures, task (red) and configuration space (blue), during 500 iterations and averaged over 30 runs. Simultaneous Dispersion Reducing Tree (DRSim) sampling with singularity robust pseudoinverse control local planning.

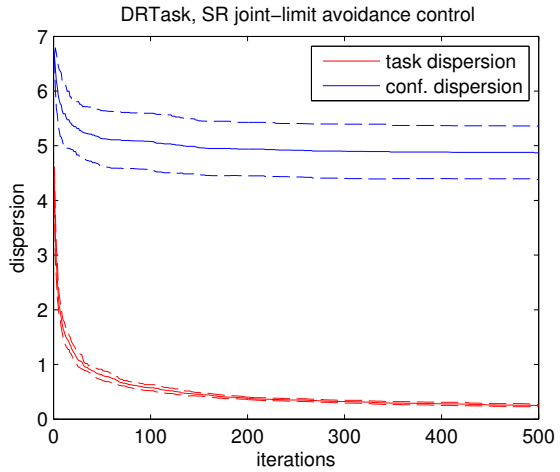

Figure 5.12: Dispersion evolution during 500 iterations, averaged over 30 runs. DRTask with singularity robust joint-limit avoidance local planning.

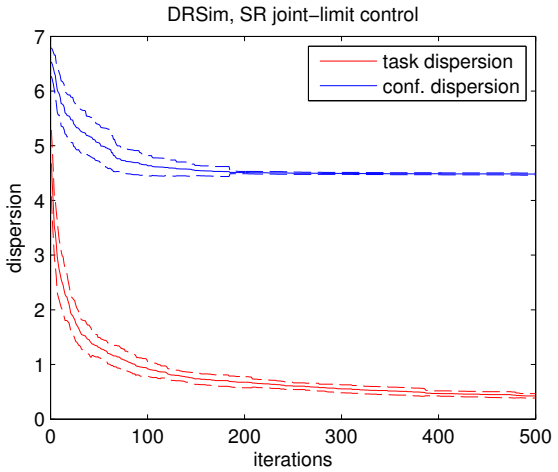

Figure 5.13: Dispersion evolution during 500 iterations, averaged over 30 runs. DRSim with singularity robust joint-limit avoidance local planning.

Setup	Algorithm			
3 joints, no obstacle	DRTask		DRSim	
Control	Task Disp.	Conf. Disp.	Task Disp.	Conf. Disp.
Singularity Robust	0.25 (0.02)	4.87 (0.38)	0.41 (0.06)	3.63 (0.55)
Joint Limit Avoidance	0.25 (0.02)	4.87 (0.49)	0.43 (0.04)	4.48 (0.02)

Table 5.1: Average remaining task space and configuration space dispersion and standard deviation, after 500 iterations, over 30 runs.

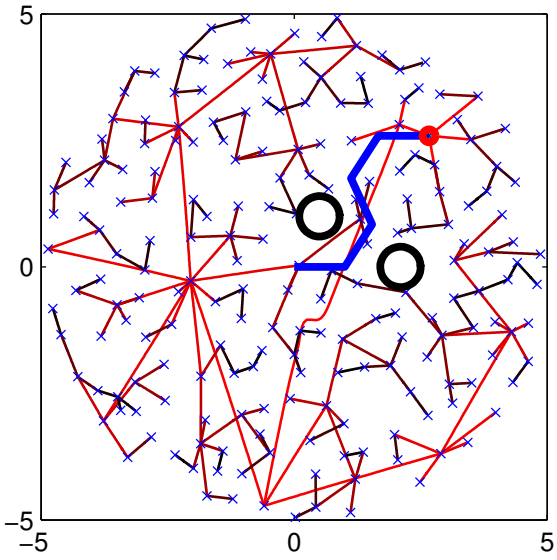

Figure 5.14: DRTask exploration tree after 200 iterations, with combined joint-limit and obstacle avoidance control. The manipulator is initially trapped between two obstacles, but during exploration a path escaping the trap is found.

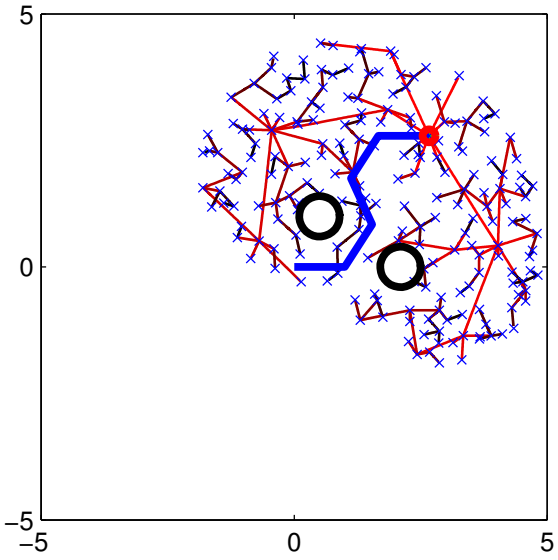

Figure 5.15: The same setup, but this time the exploration was not able to escape the trap.

2D Task Space, 5D Configuration Space, Obstacles and Joint-Limits

So far, the setup was quite simple, the whole space was easily accessible. To create a more interesting environment, additional constraints are now added to the robot and to the workspace. Configurations are constrained by obstacles and by joint-limits of the manipulator. Also the number of joint and therefore the dimensionality of the configuration space is increased to five joints respectively dimensions. Two obstacles are placed in a way that substantially limits the freedom of movement, making it necessary to traverse a narrow passage.

The layout of he obstacles and the initial configuration of the manipulator can be seen in figure 5.14. For this example, sampling is done using the DRTask approach with combined joint-limit and obstacle avoidance local planning. The whole task space is explored and regularly covered. Another run of the algorithm is shown in figure 5.15. In contrast to the previous example, the space is not completely covered, the exploration was not successful. Apparently, the manipulator was not able to get out of the obstacle trap. Such cases happened several times and increasing the number of iterations up to 2000 did not help, there are always cases where the exploration gets stuck.

For a better understanding of this effect, the evolution of the task and configuration space dispersion is evaluated, again over the course of 500 iterations and averaged over 30 runs. The results reveal two clearly distinguishable trends. Some runs reached a low task dispersion value below 4, while others quickly converged to a substantially higher value. This separation can be interpreted as a grouping into runs that were able to escape the obstacle trap, achieving a low dispersion, and runs that got stuck, not able to explore the whole task space. Figure 5.16 shows the dispersion evolution separated into these two groups. The majority of trials (26/30) belong to the group of unsuccessful explorations, only for a few cases (4/30) the whole space was explored.

The same evaluation is done for the alternative DRSim exploration strategy. In contrast to the previous results, the trials show a coherent behavior of continuously decreasing

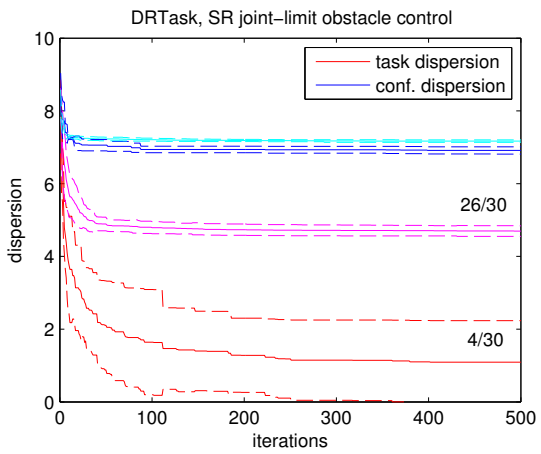

Figure 5.16: Dispersion evolution during 500 iterations, averaged over 30 runs. DRTask with combined joint-limit and obstacle avoidance control.

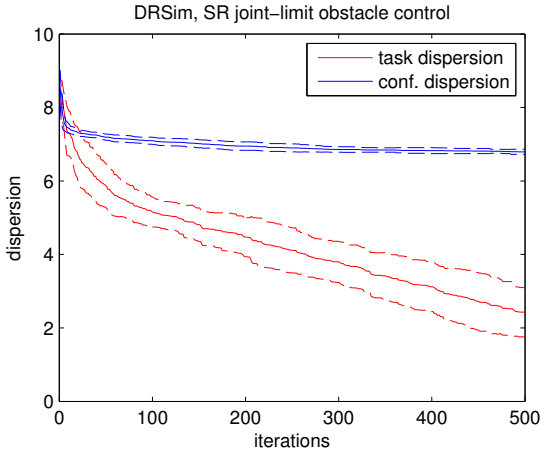

Figure 5.17: Dispersion evolution during 500 iterations, averaged over 30 runs. DRSim with combined joint-limit and obstacle avoidance control.

dispersion. No group of runs converging to a higher dispersion can be found. The average evolution of dispersion is displayed in figure 5.17. Although the rate of dispersion reduction is slower now, all runs succeed in systematically exploring the task space and in escaping the obstacle trap. For an example exploration tree refer to figure 5.18. Figure 5.19 shows a particular path traversing the trap.

The exploration behavior of the two sampling approaches is also evaluated for the other local planning methods, used in the previous setup. Table 5.2 summarizes all results, including plain singularity robust motion control, joint-limit avoidance redundancy resolution and combined joint-limit and obstacle avoidance. As already discussed for the latter case of joint-limit and obstacle avoidance control, exploration does not always succeed in covering the whole space and gets sometimes trapped. To a various degree, this is the case for DRTask sampling under all three local planning conditions. For singularity robust control without explicit redundancy resolution, most trials successfully explored the whole space (28/30) and only few are trapped (2/30). If joint-limit avoidance is added however, not a single exploration is able to escape, all 30 runs are trapped. Finally, with both joint-limit avoidance and obstacle avoidance, some trails are able to escape (4/30) while a larger fraction is not able to cover the whole task space.

The exploration of the DRSim sampling strategy shows a more reliable behavior. All trials are able to traverse the obstacle gap and to successfully explore the task space, for the cases of singularity robust control and combined joint-limit and obstacle avoidance control. Although only some runs (5/30) escaped if pure joint-limit avoidance is used, this is already an enhancement compared to the DRTask approach, where not a single run was able to escape.

The results of table 5.2 further show differences in the dispersion reduction performance of both approaches. If DRTask is able to escape the trap, the task space dispersion drops to a low value around or below 1. With DRSim, the reached task dispersion remains larger, close to 2.5. Comparing the configuration space dispersion values, no such large differences can be seen. If the exploration proceeds through the gap, the dispersion is a little lower, but the achieved minimum value remains high for both approaches.

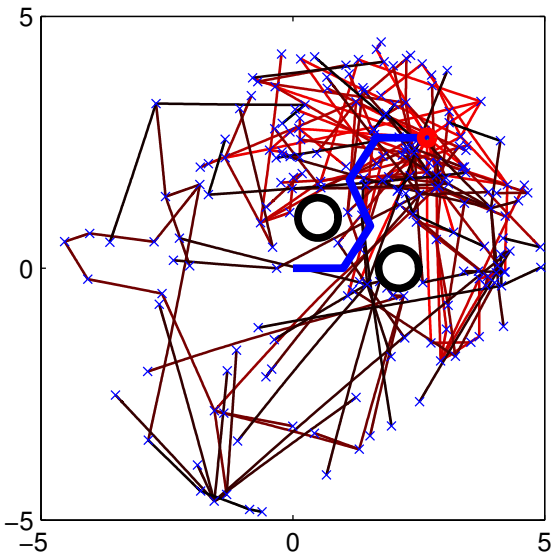

Figure 5.18: DRSim exploration tree after 200 iterations, with combined joint-limit and obstacle avoidance control. A path escaping the obstacle trap is found and the whole space can be explored.

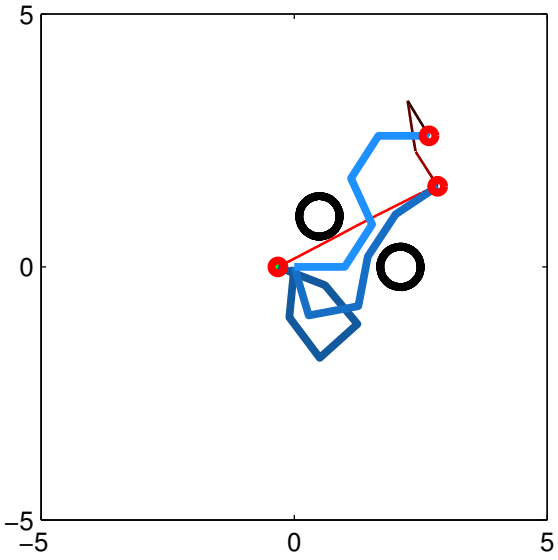

Figure 5.19: An individual sub-path of the DRSim tree, traversing the narrow passage between the two obstacles.

Setup	Algorithm			
5 joints, obstacle trap	DRTask		DRSim	
Control	Task Disp.	Conf. Disp.	Task Disp.	Conf. Disp.
Singularity Robust	28/30 escaped		all escaped	
	0.65 (0.18)	6.85 (0.10)	2.69 (1.21)	6.89 (0.21)
	2/30 trapped			
	5.72 (0.04)	7.18 (0.02)		
Joint Limit Avoidance			5/30 escaped	
			2.57 (1.57)	6.73 (0.01)
	all trapped		25/30 trapped	
	5.05 (0.12)	7.11 (0.05)	4.99 (0.04)	7.09 (0.01)
Joint Limit and Obstacle Avoidance	4/30 escaped		all escaped	
	1.09 (1.15)	6.91 (0.10)	2.43 (0.67)	6.79 (0.07)
	26/30 trapped			
	4.69 (0.15)	7.17 (0.04)		

Table 5.2: Average remaining task space and configuration space dispersion and standard deviation, after 500 iterations, over 30 runs.

5.3.7 Discussion

Two different sampling strategies were compared with respect to their exploration abilities. DRTask only regards the task space and tries to explore this space as fast as possible by greedily reducing the task space dispersion. In contrast, DRSim aims to balance the exploration in the task and the configuration space. Task space sampling also focuses on reducing the dispersion in every step, by selecting the task target sample with the highest local dispersion reduction effect. In addition, a configuration space dispersion measure is used to steer the exploration process on a global level towards less covered regions. This way, the dispersion is simultaneously reduced in both spaces.

The results of the simulation study show that the methodology of an incremental dispersion reduction sampling indeed lead to a continuously fast exploration of the task space. The DRTask approach can reduce the dispersion faster by determining the optimal extension step exclusively in the task space, while DRSim has to balance between task space and configuration space dispersion reduction, which is not always possible at the same time. Nonetheless, with the sampling process progressing, the actual amount of remaining task dispersion converges to a similar value with both approaches. The speed of exploration is lower, not the principal ability to explore.

The second more realistic setting with obstacle and joint-limit motion constraints reveals a more profound difference between both sampling approaches. Depending on the actual setup of constraints and local planning method, the exploration of the DRTask tree exhibits a divergent behavior. While in some cases the task space is rapidly explored to low dispersion values, other cases fail to explore and are converging to a significantly higher value. The exploration gets trapped and refining the exploration with more iterations does not help. The DRSim exploration on the contrary shows a more stable exploration ability. Task dispersion is continuously decreasing, even though the lowering rate is slower.

Since the DRSim sampling approach incorporates the configuration space dispersion, exploration is continued in regions with already low task space dispersion if the corresponding configuration dispersion remains high. A high configuration dispersion implies the presence of alternative configurations for similar tasks that might be discovered by re-visiting already explored regions. It was shown during evaluation that the configuration space dispersion can be indeed lowered to a greater extend this way. Considering this alternative configurations is the reason why DRSim prevents situations of trapped explorations. Even if configurations were chosen that turn out to result in a trap, alternative and more successful choices can still be discovered in later stages. DRTask, conversely, does not re-visit task regions and the further exploration is completely dependent on the previous choices. If these turn out to be unfortunate later, there is no way to recover.

In order to benefit from DRSim's ability to recover from local planning failures, the assumption is of course that configuration alternatives can be found with local planning. Technically, this property is given if local planning possesses nonholonomic behavior. In this study, all of the used local planning methods involve some sort of nonholonomic constraints. While the task representation behaves holonomic if trajectory start and end points are reached with zero velocity, resolved motion rate control does not, refer to section 2.6.

The impact of this nonholonomic behavior is quite different, depending on the redundancy resolution method and the concrete layout of constraints, particularly the placement of obstacles. Pure singularity robust control is not influenced much in the tested case, only a few trials did fail with DRTask and DRSim is successfully able to resolve all these cases. On the other hand, combined joint-limit and obstacle avoidance is strongly affected. Only for some cases DRTask can succeed here while DRSim benefits unambiguously and is again able to always escape the trap. If only joint-limit avoidance is used for redundancy resolution,

DRTask is not able to explore the whole task space. The DRSim exploration however is only able for some cases to traverse the obstacle gap. This can be explained by the relatively weak non-holonomy arising from joint-limit avoidance, which is mainly due to sub-optimal cost function optimization. It is not directly possible to systematically exploit this behavior to reliable generate alternative configurations by task sampling.

From this discussion of the properties of the two opposing sampling strategies, some conclusions can be inferred regarding global task space planning as proposed in the hybrid planning formalism. One observation is that the iterative dispersion reduction approach is a good heuristic to achieve a fast and throughout exploration. Secondly, it was shown that under the presence of nonholonomic constraints, the exploration ability is more robust if the state space is incorporated into the sampling heuristic. A too greedy task space sampling that neglects the state space coverage might lead to stagnating exploration.

5.4 Completeness

In this section, the issue of completeness of planning is addressed. With hybrid planning, the globally searched space is no longer the configuration space and the question arises, if completeness guarantees can still be made.

In motion planning, completeness is defined with the piano movers problem, refer to section 4.2. Basically, a complete motion planning algorithm has to be able to find every possible free path and to be able to recognize if no solution is possible. Completeness of hybrid planning was defined very similar in section 5.2. A difference is that there is no goal configuration given, only a goal task position, potentially satisfied by a multitude of configurations. Also a solution path is now represented by a sequence of task space control inputs. Hybrid completeness is given if a path between a start configuration and any goal configuration, whose corresponding task position matches a given goal task region, exists and a sequence of task control inputs can be found that creates such a configuration path. Paths are generated by these task space control inputs by integrating the described transition functions for task and configuration space velocities. To be able to examine the completeness of an actual hybrid planning approach, properties of these transitions have to be understood for both the global task space planning and the local configuration space planning process:

Global planning:

The global planning process is determined by the task space transition function f. Inputs to the system are given by means of the current task position x and a task control input u, while the output $\dot{x}$ is feed to the local planning process. The task space path results from integrating the output velocities $\dot{x}$. The global process is complete now if every possible task space path can be created by integrating a sequence of control inputs.

As explained in section 2.6, this property can be examined by characterizing the reachable set of a system. For instance in section 2.6.1, the second-order attractor dynamic task representation was shown to be able to reach every state of the system and with the special case of target-to-target movements with zero velocity at the target points, all possible task space paths can be created.

Local planning:

The local planning process utilizes a second transition function g, specifying configuration space velocities $\dot{q}$ for given configuration space positions q and target task velocities $\dot{x}$. Completeness can once again be achieved if it is possible to create every configuration space path by integration of the output velocities. That is, if every

possible path resulting from integrating $\dot{q}$ can be generated with the right sequence of task velocities $\dot{x}$.

If this is possible depends on the nature of the system g. It was shown in section 2.6.2 that if a redundant control system posses holonomic constraints, the reachable space is limited to a manifold of the configuration space. In this case, the local planning process can not be complete because there is no way to create paths leaving this manifold by altering the input sequence of task velocities.

For a redundant control system with nonholonomic constraints, the whole space might become reachable, dependent on the properties of the actual system. The gradient projection control scheme used in this thesis can be characterized as a nonholonomic system under certain conditions and consequently, local planning can create different paths with different input task velocities sequences. Full state space reachability however can not be generally established but is dependent on the actual kinematics, the employed cost function for redundancy resolution and the structure of constraints.

The completeness of hybrid planning can thus be determined by examining the planning processes in both layers. It is relatively easy to achieve completeness of the global planning component, as for example described for the attractor dynamic case. The overall completeness is then determined on the local planning layer, where completeness is harder to achieve. If the same task input always results in the same configuration, it is impossible to achieve completeness in configuration space even if the task space is completely searched. If different configuration can be produced, all possibilities might be reached, but this is not necessarily always the case.

Consequently, hybrid planning is not automatically complete in terms of considering all possible configurations. In fact, the limitation of the globally complete search to the lower-dimensional task representation already implies that not all motion alternatives are explicitly included into the search. However, a softer condition of completeness can be pursued instead. If the local planning component is able to choose valid constraint free configurations with high probability, then the probability of finding a global solution path is also high, given that the global planning layer explores a, preferably complete, set of alternatives.

Of course a misbehaving local planning behavior can always happen, as demonstrated in the previous section 5.3. In such cases, it is valuable to adopt a robust global sampling strategy like the DRSim approach to be able to recover from unfortunate local planning steps.

Chapter 6

Hybrid Planning for Redundant Robots

The previous chapter introduced our concept of hybrid motion planning on a rather theoretical level. This chapter develops an application of the method for motion planning with a redundant manipulator. After a short discussion of the specific challenges, a sampling-based hybrid planning algorithm is presented in section 6.2, using a simple space invariant coverage heuristic. The approach is evaluated for a humanoid robot as an example for a redundant robot. It is compared against an alternative planning method, performing a full configuration space search incorporating a goal directed task bias, as described in section 6.3. Evaluation is done with the help of a specifically developed simulation framework 6.4. The evaluation setup and results are discussed in the last part of this chapter.

6.1 Redundant Robot Motion Planning

Redundant robots are characterized as robots having more degrees of freedom than necessary to fulfill a certain task. A common case are sequential manipulators with more joints than task variables, as previously described in chapter 2. This type of redundancy can be often found in robotic systems, for example in industrial manipulators. Redundancy occurs in multi-limb systems as well, for example in multi-arm setups or multi-fingered hands. A particularly interesting family of redundant systems are humanoid robots. In section 2.5, a control framework for humanoid robots was presented, already highlighting some of the specific challenges regarding the implementation of motion control techniques for these robots.

With respect to motion planning, redundant robots pose some challenges. They typically have a large number of joints, resulting in high-dimensional state spaces and complete motion planning by explicitly searching or sampling the state space can become prohibitively expensive. This is even more the case for applications involving more than one manipulator. Not only increases the state dimension with every additional joint, also the collision constraints become very complex. This difficulty arises in the context of bi-manual planning for instance and is described with more detail in chapter 7.

In contrast to the high-dimensional state spaces, the task to be planned is often defined in a lower dimensional space. For instance, the task for an industrial manipulator could be to move a tool to a specific spatial position while the posture is not specified. A humanoid with the task of grasping some object can typically do this in different ways, all equally valid. This poses difficulties for state space motion planning because the target is not exactly pre-determined. There exist a set or even a whole continuum of valid target postures and it is not clear how to select the right target in a certain situation.

These two properties, high-dimensional state spaces and lower-dimensional task specifications, motivate to treat redundant robot motion planning with the proposed hybrid motion planning approach. By focusing the search on the task representation, global search is limited to a lower dimensional space and the problem of assigning the right target posture is simultaneously solved without the need to determine the posture beforehand.

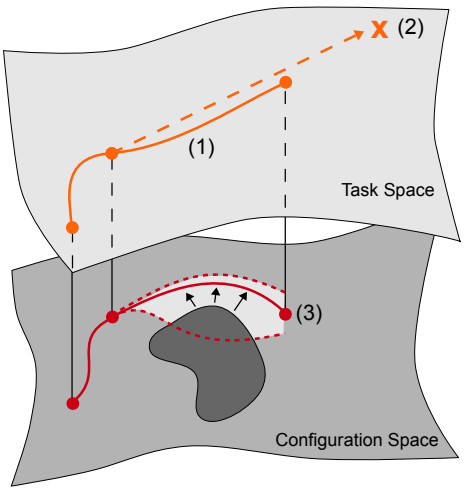

Figure 6.1: Hybrid planning with local redundancy exploitation. A task space trajectory (1) towards a task space control target (2) is created. The corresponding configuration space trajectory is locally optimized using the redundant space, shown in light gray (3). For example the optimization can be based on a obstacle distance potential function, driving the trajectory away from obstacles. Note that every possible trajectory in the redundant space is mapped to the same task space trajectory.

Key element of the hybrid approach is the availability of a control component establishing the representation shift from the state space to the task space. For redundant robots, various suitable motion control methods exist. Chapter 2 presented control techniques based on resolved motion rate control, able to efficiently create state space trajectories and to exploit redundancy for local motion optimization. Many different redundant space motions can be implemented, but in the context of motion planning the avoidance of constraints like joint-limits and obstacles are especially rewarding.

Figure 6.1 illustrates the hybrid planning process for redundant robots. The task space is globally searched by a systematic sampling of task targets, incrementally creating a planning tree of task space trajectories. Using the motion control framework, the task space trajectories are mapped to configuration space trajectories and by utilizing local redundancy optimization, obstacles can be avoided.

There are several motion planning approaches for robotic manipulators that include ideas of task specific search heuristics and representations. Decomposition techniques, for instance, proceed by dividing the problem into two subproblems. The first subproblem is to determine a workspace tunnel of free space, assumed to contain the swept volume of the robot following a solution path. The second subproblem is then to find a collision free path for the robot inside this tunnel. Creation of the tunnel can be done by iteratively sampling collision free spheres of a maximal radius in a goal directed best first fashion (Brock and Kavraki, 2001). For the movement inside the tunnel, a potential function is defined. The centers of the free space tunnel spheres attract the end-effector towards the goal, while another potential function repels the robot away from obstacles. Elastic Roadmaps (Yang and Brock, 2010) capture the connectivity of the workspace with a sampling-based roadmap. Paths of the map are followed employing a feedback control framework, while roadmap nodes are allowed to move in response to obstacles and the connections between them are constantly updated.

These decomposition techniques share the idea of utilizing control methods that deal with actual trajectory generation, while a global process determines a rough higher level plan. Since the global connectivity is determined in the workspace however, the approach is best suited for mobile manipulation applications, where the free space is relatively large and individual workspace positions can be often connected with local control. For stationary manipulation tasks with highly redundant robots in highly constrained environments, local connections are difficult and a reasonably precise approximation of the free workspace can become costly to obtain.

Relying on local control methods, the process of generating a movement along a workspace path can fail due to local minima. The decomposition technique in (Rickert et al., 2008) is able to recover from this situations by adaptively switching between two planning modes. In exploitation mode, a goal directed workspace tunnel is followed, again utilizing a feedback control framework. If this local workspace exploitation gets stuck, the algorithm switchs towards a RRT-like exploration mode, able to overcome local minima by sampling in the configuration space.

The planning method in (Gochev et al., 2012) is also designed to switch between planning modes. Motivated by the observation that large parts of a solution path can have a lower-dimensional structure, the planner constructs a state-space with adaptive dimensionality. The complete high-dimensional state space is only included in situations where no low-dimensional solution is possible, reducing the size of the state space and consequently the time and memory requirements. One planning mode searches a low-dimensional representation with fixed structure and dimension and a second planning mode searches the full high-dimensional space. The algorithm relies on a suitable discretization of both spaces, which can be problematic for higher dimensional spaces. If a too coarse discretization is used, a significant portion of motion freedom is lost, while a too fine discretization quickly raises memory and time requirements. Thus, the number of motion primitives that can be incorporated is limited in practice.

Other methods primarily operate in the full-dimensional state space, but employ different task driven heuristics. Bertram et al. (2006) accompany a RRT exploration process with a heuristic that is used to bias the search based on an estimate of the workspace distance of the robot to the goal and to obstacles. JT-RRT (Weghe et al., 2007) builds upon the same principle of searching the configuration space, but imposes a stronger goal bias by randomly performing direct movements towards the goal in the end-effector space. For this the Jacobian transpose is employed as a approximate mapping between end-effector poses and configuration space postures. The Jacobian transpose is also used in (Vahrenkamp et al., 2010) to guide a configuration space search, but here the goal is to simultaneously find a trajectory and a suitable grasp position for a one or two handed object manipulation task. End-effector motions are performed to a broader set of possible end-effector positions, determined by grasp generation and scoring heuristics. The BiSpace planner (Diankov et al., 2008) uses two search trees: A forward tree in configuration space starting at the current posture and a backward tree in the workspace, originating from the workspace target. The search of the RRT based algorithm is focused on connecting both trees utilizing a workspace metric. This way, the forward configuration search can be guided towards the goal.

General ways to guide the search by focusing on certain regions of the space were mentioned in section 4.5. Some of these methods also employ heuristics that can utilize lower dimensional information. Sucan and Kavraki (2009b) show that it is beneficial to aim towards a uniform coverage of a lower dimensional linear projection space instead of the full high-dimensional state space. It is demonstrated that for a redundant manipulator for instance, one can use the end-effector position as such a projection space to effectively guide the search towards the goal. The sampling approach of (Dalibard and Laumond, 2011) tries to learn such low-dimensional projections in order to detect lower dimensional structures of the free space. In contrast to the previous method however, only local constraint information is taken into account, no globally valid heuristic is identified.

The task representation employed by our hybrid method can be interpreted as a similar low-dimensional guidance space. Unlike to the previous learning method, the space is specifically designed for the task at hand, allowing to incorporate domain dependent knowledge which can result in a more powerful heuristic. The linear projection space coverage estimation technique uses a predefined representation as well, but like all previous state space sampling

methods is still searching the configuration space. It is possible to guarantee probabilistic completeness this way, but the problem of globally sampling a high-dimensional space is not eliminated. It is assumed that the low-dimensional projection or task level control heuristic guide the search, but the success in reducing the sampling effort is dependent on the actual situation.

The proposed hybrid approach, conversely, shifts global planning entirely to the task representation. A method performing the entire search in the task space as well is presented in (Shkolnik and Tedrake, 2009), where a Jacobian pseudo-inverse feedback controller generates configuration space trajectories for a highly redundant robot arm based on a low-dimensional task specification. It is shown that searching the space of task positions with a modified RRT algorithm can lead to an enormous performance improvement in contrast to high-dimensional configuration space search. This approach is similar to the hybrid motion planning framework presented in this chapter. Global sampling is also limited to a task representation and a pseudoinverse control framework is employed for local motion generation. Our method incorporates more powerful methods for both the global task space and the local configuration space motion generation. It is possible to solve a broader class of problems and to solve a larger subset of planning queries.

The next section describes a sampling-based planning algorithm, implementing the described framework. The algorithm is independent of the actual motion control method used. The actual task representation and local planning component used for evaluation are discussed in section 6.5.

6.2 A Hybrid Motion Planning Algorithm

The hybrid motion planning algorithm developed in this section tightly follows the EST random-tree approach (Hsu et al., 2002), as introduced in section 4.4.1. First step in each iteration of the incremental tree growing process is to select a tree node for extension. The selection is done by randomly picking nodes according to a probability distribution, assigning a individual weight to each node. This weight should reflect how well certain regions of the space are covered and by assigning higher weights to nodes in less covered regions the search is biased towards unexplored parts of the space.

Section 5.3 evaluated different task sampling strategies for hybrid motion planning. It was concluded that under the presence of nonholonomic constraints and in particular when resolved motion rate control is used, it becomes necessary to incorporate a configuration space coverage measure into the node selection heuristic. With the DRSim algorithm a suitable technique was developed, relying on the estimation of dispersion in the task and configuration space to guide the exploration. However, we would like to us a simpler heuristic here in order to ease a proof-of-concept implementation and evaluation. Instead of incorporating two coverage estimates, separately computed for both spaces, a single heuristic that is invariant to both spaces is used. The heuristic is taken from the EST implementation of (Plaku et al., 2007) and just counts the number of outgoing edges of a node and assigns the weight inversely proportional to this number. This is an estimate of local coverage, because every neighbor in the tree stands for a successful exploration in the direct vicinity of the node. It is invariant to the task respectively configuration space, because the node connectivity is the same, regardless of the spatial arrangement of nodes.

The complete *hybrid task space tree growing* (TaskTree) algorithm is shown in algorithm 6.1. It operates by executing the five EST steps node selection, target sampling, tree extension, tree node adding and weight update in each iteration, refer to section 5.3. Each node p_i in the tree data structure consists of a pair of task space position x_i and corresponding

Algorithm 6.1 Hybrid Task Space Tree Growing with Greedy Goal Bias (TaskTree)

tree with nodes $p_i = (x_i, q_i, u_i, t_i, w_i) \in T = (\mathcal{X}, \mathcal{C}, \mathcal{X}, \mathbb{R}, \mathbb{R}^d)$
metric $\rho(x, y), x, y \in \mathcal{X}$
while not goal reached **and not** time over **do**
  if goal bias **then**
    p_s = nearest tree neighbor to goal with $\min_{x \in T} \rho(x_{goal}, x)$ not yet goal biased
    $u_{tgt} = x_{goal}$
  else
    p_s = select tree node based on weights $w_i \in T$
    u_{tgt} = sample in neighborhood of p_s
  end if
  $(x_{tgt}, q_{tgt}, t_{tgt})$ = local planner from (x_s, q_s) towards u_{tgt} with max. duration t_{max}
  if $t_{min} \leq t_{tgt} \leq t_{max}$ **then**
    add new tree node $(x_{tgt}, q_{tgt}, u_{tgt}, t, w_{new})$ to T, branching from p_s
    update tree weights $w_i \in T$
  end if
end while

Algorithm 6.2 Node Selection

select tree node based on weights $w_i \in T$:
  p_s = choose randomly according to weight distribution $\{ w_i \mid i = 1...N \}$

configuration space position q_i as well as a task space control target u_i and control time t_i that was used to create the node. This way a node always represents the end of a local motion and can be continuously resumed with a new local motion towards another target. Also a weight w_i is associated with each node, determining the probability of the node to be selected for further exploration.

At the beginning of each iteration of the algorithm, a decision is made if the tree should explore or should try to reach the goal directly (goal biasing). The goal biasing case is done by selecting the closest tree node p_s to the goal position x_{goal}, according to the task space metric ρ, and setting the goal position as the local control target u_{tgt}. This method has a greedy behavior, since the closest node is always chosen. However the algorithm maintains a flag at each node to keep track if the node was already selected for goal biasing and to make sure that each node is only selected once. If the closest node is already goal biased the second closest is taken etc., thus even nodes that are far away from the goal will eventually be used, preventing the goal biasing to get stuck in unfortunate situations.

If no goal biasing is done, the algorithm performs an exploration step. The node selection module (algorithm 6.2) is called to select a tree node p_s at random with a probability distribution given by the exploration weights of each node. For example, consider a tree node j with associated wight w_j. The probability of selection of this node equals w_j / w_N with $w_N = \sum_{i=1...N} w_i$.

Next, the target sampling component (algorithm 6.3) samples a new task space target u_{tgt} in the neighborhood around the selected node p_s. Given the target, the local planner is called to create a new trajectory, starting at the selected node and extending the tree towards the sampled target u_{tgt} with a maximum control duration time t_{max}. The local planner utilizes the underlying control framework and is described later in section 6.5.3. The whole trajectory is represented as a pair of corresponding task space and configuration space points and after each iteration step, the local planning module returns the portion of

Algorithm 6.3 Target Sampling

sample in neighborhood of p_s :
u_{tgt} = sample from $\mathcal{X}$ in neighborhood of node p_s

Algorithm 6.4 Weight Update

update tree weights $w_i \in T$, selected node p_s :
weight of new node $w_{new} = 1$
weight of selected node $w_s = 1/\text{number of node edges}$

the trajectory not violating any constraints, represented by the trajectory end-points x_{tgt} and q_{tgt} as well as the elapsed control time t_{tgt}.

Finally, if the time duration of the trajectory is between min and max values, a new tree node is added to the tree, branching from the previously selected node p_s. Also, the weights of the tree nodes are updated by the weight update component (algorithm 6.4). The initial exploration weight of each new node is set and the exploration weight of the extended parent node is updated to reflect the exploration that just happened.

6.3 A Control-Enhanced Motion Planning Algorithm

The classical approach to motion planning is to search the robots configuration space directly. Like already mentioned, there are some related motion planning techniques that enhance the classical approach in order to deal with redundant robots. To compare the proposed hybrid motion planning concept, we choose to compare against these methods, in particular against the JT-RRT method (Weghe et al., 2007). In brief, JT-RRT performs a classical RRT search in the configuration space, but adds goal directed movements with an approximate task space control method, acting as a strong bias to focus the search. Also it is no longer needed to predefine a set of possible goal configurations, it is enough to define a goal in the lower dimensional task space and a valid configuration at the goal is computed during the search.

Algorithm 6.5 shows an adapted version from the original JT-RRT algorithm. It also grows a configuration space search tree with interleaved target directed motions in the task space. The difference of the presented algorithm is the use of a powerful local control method instead of the simple approximate Jacobian transpose. This way, the algorithm uses a even stronger bias towards the goal than JT-RRT.

Basically, there are two modes of operation. In the normal case, a classic RRT configuration space search is done, refer to section 4.4.1. First a random sample in the configuration space q_s is created and the nearest neighbor node p_n in the already explored tree is determined. For this nearest neighbor computation a suitable configuration space metric φ is defined. The second step then is to connect the nearest neighbor node with the sampled node, using a simple configuration space local planner, for example a simple straight line interpolation, and to simultaneously check this movement for constraint violations. If the movement is valid, the sampled point is added to the tree. Each tree node holds a configuration space position and the corresponding task space position, thus it is necessary to compute the forward kinematics of the sampled configuration space position x_s and add it to the tree as well.

The second mode of operation is the goal bias case, randomly executed with a certain probability. In this mode, a goal directed movement in the task space is done. For this, the

Algorithm 6.5 Configuration Space RRT with Greedy Task Space Goal Bias (ConfTree)

```
tree with nodes p_i = (x_i, q_i, t_i) ∈ T = (X, C, R)
metric ρ(x, y), x, y ∈ X
metric φ(q, p), q, p ∈ C
while not goal reached and not time over do
    if goal bias then
        p_n = nearest tree neighbor to goal with min_{x∈T} ρ(x_goal, x) not yet goal biased
        (x_g, q_g, t_g) = local planner from (x_n, q_n) towards x_goal with max. duration t_max
        if t_min ≤ t_g ≤ t_max then
            add new tree node (x_g, q_g, t_g) branching from p_n
        end if
    else
        q_s = sample from C
        p_n = nearest tree neighbor to sample with min_{q∈T} φ(q_s, q)
        if connect q_s and q_n then
            x_s = forward kinematics q_s
            add new tree node (x_s, q_s, t = 0) branching from p_n
        end if
    end if
end while
```

nearest tree node p_n to the task space goal x_{goal} is computed according to the task space metric ρ. Starting at the closest tree node, the control based local planner is invoked, creating a task space trajectory towards x_{goal} with a maximum duration t_{max}. If the trajectory is valid, that is longer than t_{min} and shorter than t_{max} and not resulting in collision, it is added to the tree.

It is possible to arbitrarily interleave these two operation modes, because every tree node holds the information needed for both operations, the configuration space posture and the corresponding task space position. If the goal biasing probability is large, fast progress towards the goal can be achieved but also the exploration breaks down. In practice it is reasonable to limit the amount of goal biasing and to let the configuration space exploration dominate the search.

6.4 Simulation Framework

A simulation framework was developed in order to evaluate the hybrid planning approach. Basically the framework consists of three modules, a geometric world model, a local control method and a sampling-based planner. Figure 6.2 illustrates how these modules relate to each other.

The robot and world model maintains the geometrical representation of the robot and other objects in the workspace. Its main purpose is to check for collisions and to compute distances, also this module can be used to visualize the planning setup and solution trajectories. It is implemented using the Vortex physics simulation library[1], offering fast collision detection and distance computation methods for geometric primitives like spheres, boxes and capsules as well as for polygon meshes. In the present simulations, only simple primitives are used for fast computations. The local control module encapsulates the motion control algorithm

[1] CM Labs Vortex,http://www.vxsim.com

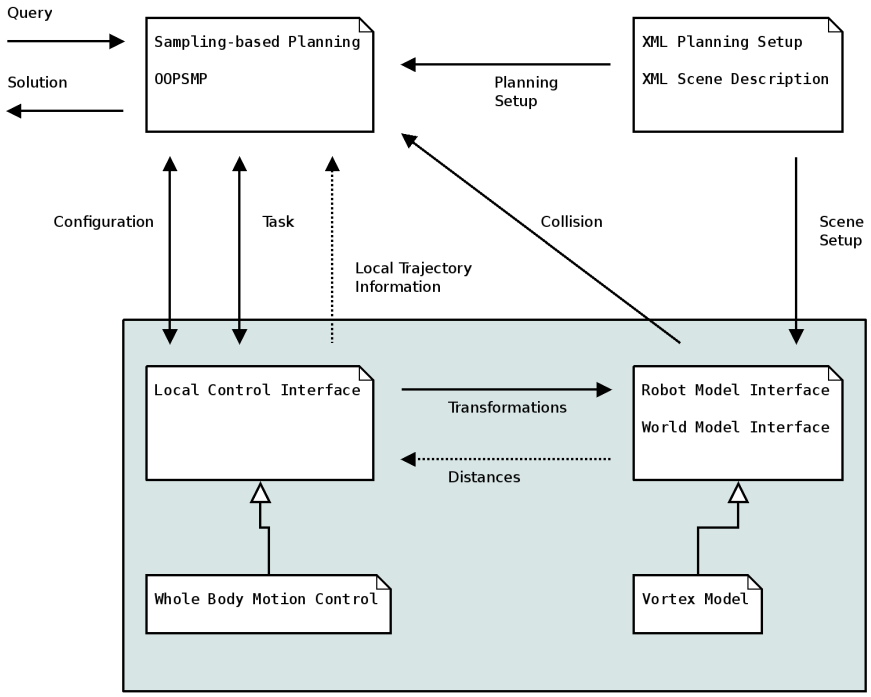

Figure 6.2: Schematic architecture of the hybrid motion planning framework. The three main components, sampling-based planning, local control and robot/world model are shown and the information flow is depicted by arrows.

used to create motions of the robot. Given a task specification, a trajectory is computed which fully determines the position of every segment of the robot for every timestep.

Finally, the sampling-based planning component takes care of the overall search process. It maintains a data structure of already searched parts of the space and decides where to search next. This part is implemented using the OOPSMP sampling based planning library[2] (Plaku et al., 2007). OOPSMP follows a plug-in driven design, making it easy to add or modify parts of the system. Following a factory design pattern, the whole setup can be specified with XML files that are parsed to dynamically create all needed component instances.

There are two loops in the architecture of the hybrid planning framework. One loop that is iterated in every step of the simulation is the coupling between the local control module and the robot model. This can be seen as the world simulation loop. Once the controller has determined a current robot posture, the corresponding transformation of each robot segment is passed to the robot model. The geometry of the current robot posture is then used to pass closest point distances between robot segments or between the robot and other objects back to the controller. Also the updated robot posture can be checked against collisions.

The other conceptual loop in the system architecture defines the interplay between the described world simulation loop and the sampling-based planning module. In order to set a new sub-target for the local control, the planning module sends the information of the current posture (configuration) of the robot and the task target, which the controller should reach. This is the input of the control system and after the system is iterated the same modalities are sent back as the output, i.e. the configuration and task position after a control step. In addition, another output is the information whether a collision occurred and possible some sort of information of the nature of the just created local trajectory.

[2]OOPSMP, http://www.kavrakilab.rice.edu

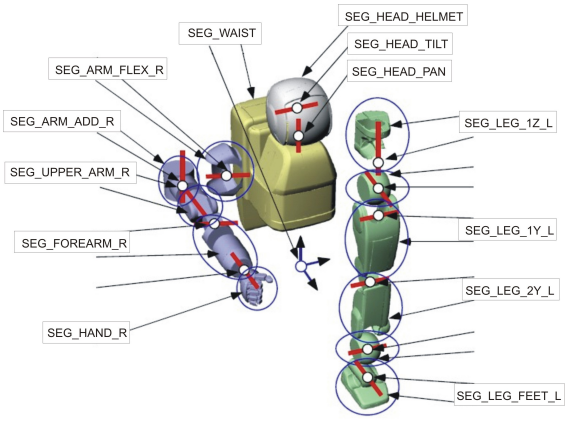

Figure 6.3: Kinematic model of the humanoid robot. In the simulation, only upper body movements are actively controlled, including the two arms, the torso and the hip. The legs are passively adjusted to counteract changes of the hip posture, but stepping or walking motions are not considered.

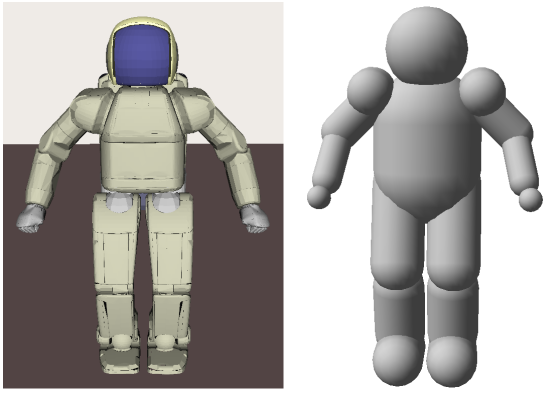

Figure 6.4: Geometric model of the humanoid robot. The real shape (left) is approximated with sphere-swept-line geometric primitives (right), greatly reducing the costs of collision and distance computations, while maintaining a reasonable level of accuracy.

6.5 Evaluation

In this section, the performance of the proposed hybrid motion planning algorithm is evaluated using a simulated planning task for a humanoid robot. Using the previously described simulation framework, the setup can be easily realized by implementing the local control module and the robot and world model. Local control wraps the humanoid whole body motion control framework from section 2.5. It is used during local planning to create motion trajectories and maintains the kinematic state of the robot. The kinematic structure of the humanoid robot used here is depicted in figure 6.3. The robot and the world model simulate the spatial geometry as shown in figure 6.4. Global planning implements the sampling-based hybrid motion planning approach (TaskTree) given in algorithm 6.1 and the classical JT-RRT based solution (ConfTree) in algorithm 6.5.

6.5.1 Task Representation

The whole body motion framework as the underlying local control method allows a flexible definition of tasks. In the present setup two task spaces are defined, one in the three-dimensional position of the hand relative to the robots coordinate frame and the other is a five-dimensional position and orientation representation, as introduced in (Gienger et al., 2005). Figure 6.5 shows this hand attitude representation. The axis centered in the hand coordinate frame and pointing in direction of the extended thumb is the so called grasp axis – the axis around which the fingers of the hand are closed. What is controlled now is the direction in which this axis is pointing but not the actual orientation angle around the axis. Thus a orientation representation is used that leaves one dimension unconstrained. The direction of the grasp axis can either be described in polar coordinates or by a point on the unit sphere centered at the hand coordinate frame.

Uniform sampling of the three-dimensional position space is trivial. Sampling in the neighborhood of a given central task space point is done by drawing points from a spatial Gaussian distribution with the peak at the center point. First a Gaussian distributed desired distance of a sample point is taken which defines a sphere on which then a uniform

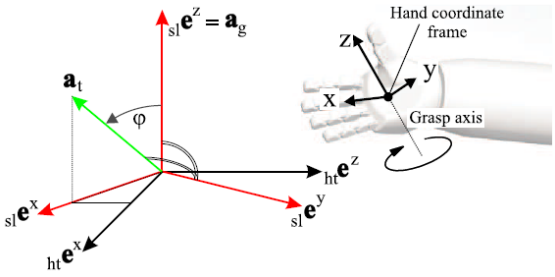

Figure 6.5: Two-dimensional hand attitude representation, from Gienger et al. (2005).

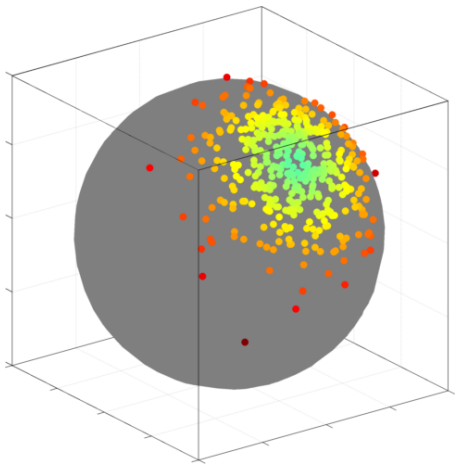

Figure 6.6: Samples from the Von Mises-Fisher distribution on the unit sphere, centered at $(0, -0.7071, 0.7071)$ with concentration $\kappa = 10$.

distributed random point is picked.

For sampling the five-dimensional position and orientation space, sampling of the orientation axis unit vector is added. This orientation vector is defined by the difference vector from a point on the three-dimensional unit sphere to the center of the sphere. Sampling thus reduces to finding points on the unit sphere. The transformation of the neighborhood sampling idea is done by applying the spatial Gaussian distribution concept to the sphere by using the von Mises-Fisher distribution, see figure 6.6. A receipt for transforming two uniform distributions on the unit interval into a three-dimensional von Mises-Fisher distribution is given in (Fisher et al., 1981).

6.5.2 Configuration Space

Independent of which of the two task representations are used, the control framework creates movements involving the whole body of the robot, represented in the configuration space. Even if only a part of the body is relevant for the task, other parts might also be moved due to the redundancy optimization. For example, if a reaching movement is done with the left hand, the right arm might also move in order to optimize some motion criteria. Although this behavior is wanted and useful, the configuration space is much larger than necessary for the task at hand. Because the dimensionality is crucial for the classical configuration space search approach (ConfTree), the configuration space is reduced to a minimum number of relevant dimensions for this method. For the setup in this study 9 dimensions are used, 5 for the arm and 4 for the upper body and hip.

Note that the configuration space dimensionality is not reduced for the hybrid motion planning method (TaskTree). Although including more dimensions increases the size of the Jacobian, the overhead is not as big as for the configuration space search and is thus neglected here.

6.5.3 Local Planning

As already mentioned, the local planner of TaskTree and the goal bias local planner of ConfTree both employ the whole body motion control framework, as introduced in section 2.5. Redundancy is locally optimized to fulfill secondary motion objectives and here two

Algorithm 6.6 Motion Rate Control Local Planner

local planner from $(x, q) \in (\mathcal{X}, \mathcal{C})$ towards $u \in \mathcal{X}$ with max. control duration time t_{max}
while not max. duration time t_{max} exceeded **and not** constraints violated **do**
    get desired task velocities $\dot{x}$ from current position x and target u
    compute joint velocities $\dot{q}$ given configuration q and task velocities $\dot{x}$
    update current task position x
    update current joint position q
    check constraint violation of updated q
end while

different conditions for optimization are used. In the first condition, only the joint limit avoidance cost function is used and in the second condition, the obstacle avoidance cost function is added.

In the ConfTree algorithm, the majority of local planning is done in the configuration space, where two configuration space points need to be connected. This is done by a simple linear interpolation with a constant step size.

6.5.4 Algorithm Implementation

The implementations of EST and RRT algorithms, available in the OOPSMP library, were extended in order to match the TaskTree and ConfTree approaches. Both algorithms need distances metrics, TaskTree only in the task space and ConfTree in the task and the configuration space. The task space distance metric for the position is a simple euclidean metric, for positions $p_1, p_2 \in \mathbb{R}^3$, $\rho_{pos}(p_1, p_2) = ||p_2 - p_1||$. When hand orientations are involved, the spherical distance between the grasp axes is added, $a_1, a_2 \in \mathbb{R}^3$, $\rho_{rot}(a_1, a_2) = \cos^{-1}(a_1^T a_2)$. For configuration space distances, the euclidean metric is used as well, $q_1, q_2 \in \mathcal{C}$, $\varphi(q_1, q_2) = ||q_2 - q_1||$.

Distances are used to compute nearest neighbors in both algorithms. In TaskTree, finding the nearest tree-node to the goal is only necessary in the goal biasing case. Also, this can be done in constant time if the fixed goal distance is stored with every node. In contrast, the ConfTree approach makes heavy use of nearest neighbor computations in the exploration step, having to find closest tree neighbors for randomly sampled configurations. Naively implemented, this operation has quadratic complexity, but by using more sophisticated methods like GNAT (Brin, 1995), the complexity can be dramatically reduced.

6.5.5 Setup

In the sampling-based planning methodology, the environments is only directly relevant to the local planner, which is treated as a black box. Thus in principle, the actual environment and operating workspace does not matter. This is also true for the proposed hybrid planning method, only the local control deals with the geometry of the problem at hand.

The performance, however, is very much dependent on the spatial layout of the environment. For humanoid robots a natural application area are human like environments, motivating the setup used in this study. Figure 6.7 shows the first setup, consisting of a table and a grid like structure, resembling common objects like windows or cupboards. The class of tasks relevant in this environment are reaching and pick and place tasks.

Three motion planning queries are done, as shown in figures 6.9 and 6.10, for the two task spaces position and joined position and orientation. For these queries, a general local optimization method that is not specifically initialized for the problem at hand would fail.

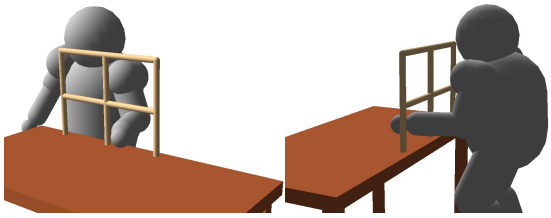

Figure 6.7: Setup A. Humanoid robot motion planning in a human environment, featuring a table and a grid like "cupboard" structure.

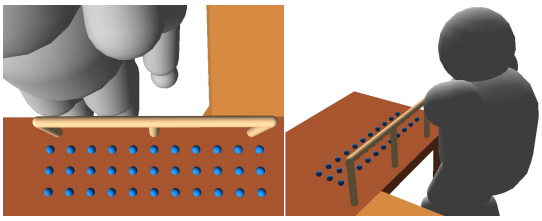

Figure 6.8: Setup B from Behnisch et al. (2010) with a smaller sized grid and a larger number of goal positions, marked by the blue balls.

A local method can only find paths that are in the same homotopy class as the initial path. There are three separate homotopy classes of valid solutions in this setup and a naive straight line initialization would not be in any of them. Thus global planning is indeed necessary, if problem specific tailoring should be omitted. In addition, especially target 1 is characterized by very constrained areas where not much movement variety is possible. This narrow passages are a difficult problem for sampling-based motion planning and they achieved a lot of attention to deal with them. To summarize, the given setup is thus both a relevant and a challenging problem for the targeted application area of motion planning for redundant robots.

Figure 6.8 shows a second setup, already used in (Behnisch et al., 2010). The spatial structure is comparable, but here the focus lies on the reachability of goal positions, tested by defining a whole set of goal positions, shown as blue balls on the table surface. Start position is always the shown posture with the arm in a rest position. The set of valid solution paths is divided into two homotopy classes here.

For the queries of setup A a number of motion planning runs is executed, testing all combinations of two task spaces (3D position and 5D joined position and orientation), two planning algorithms (hybrid planning and classical RRT with task space bias) and two local planning methods (whole body motion control and whole body motion control with redundant space obstacle avoidance).

For the setup B, the same two task spaces and local planning methods are used though. The planning algorithms are different, hybrid planning is less greedy and the configuration space planner operates as an Expansive Space Tree (EST) (Hsu et al., 2002). Because these differences only impact the runtime and not the principle reachability of goal positions, the results are reported here too.

6.5.6 Performance Metric

To compare the runtime of the algorithms, the time needed to find a solution path is measured and arithmetically averaged over all planning runs.

Due to the random nature of sampling-based methods, their completeness is only defined probabilistically, i.e. the probability to find a solution converges to one at infinite time. It is not possible to define an upper time limit at which the search can be terminated because it is always possible that a solution is still found later. In practice it is necessary to set an upper time limit and once it is reached, the search is aborted without a true runtime result, adding some uncertainty to the runtime statistics. To capture this uncertainty, another informative performance hint is the fraction of actually solved planning runs relative to the total number of runs.

A measure related to the runtime is the number of tree extensions. Every tree extension involves a call to the local planner which in turn calls the control framework, distance

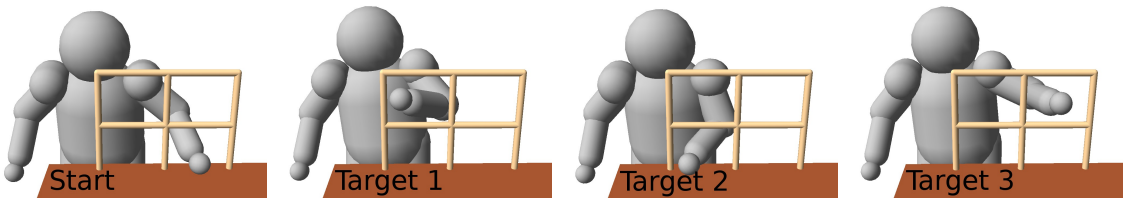

Figure 6.9: Setup A, 3D task space start and goal positions. Planning of a trajectory from the start position (first from left) to three distinct goal positions, target 1 to 3.

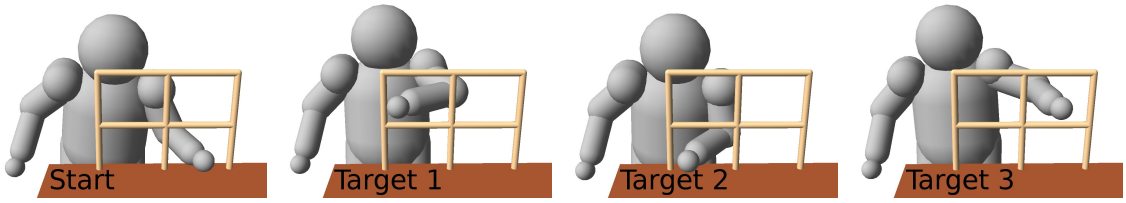

Figure 6.10: Setup A, 5D task space start and goal positions. Planning of a trajectory from the start position (first from left) to three distinct goal positions, target 1 to 3.

computation and collision detection several times. This steps are accountable for a large share of the overall computation needed.

6.6 Results

Figure 6.12 shows two examples of successful planning runs and the computed solutions paths and search trees for setup A and the TaskTree algorithm. Figure 6.13 shows the same for ConfTree. Comparing the resulting movement trajectories of both algorithms, two observations can be made. The TaskTree solution is smoother than the ConfTree solution and appears also to be shorter. While the former creates a continuous natural looking trajectory, the movement of the latter make a jaggy and unnatural impression.

Figure 6.11 and table 6.1 summarize results for planning trials using the 3D hand position task space, showing the three performance measures number of solved queries, runtime and tree size. The statistics are computed by averaging over 100 runs each and are split into groups. There are three groups that separate the three target positions. Each of the target position groups is further divided into four items, showing results for each of the four different algorithm and local control combinations. Starting with the last target position, target 3, this position shows an equal low average runtime and tree size for all algorithm and local planning combinations. This position can be easily and reliably reached. Target 2 shows more difference. Here TaskTree performs significantly better than ConfTree, both runtime and tree size are lower. Comparing the two local control conditions there is no clear advantage visible. The most significant difference between the two algorithms is present for target position 1. ConfTree takes considerable more time and is not always able to find a solution in the given time limit. While the difference between the local control methods is neglectable with ConfTree, the tree size of TaskTree differs with obstacle avoidance resulting in a smaller tree. However this difference is not visible when looking at the runtime.

Results for the 5D hand positon and orientation task space are given in figure 6.14 and table 6.2, again divided into groups of different target positions, algorithm and local planning. Similar to the 3D case, target 3 can be reached without problems. Solution time and tree size is comparable, only ConfTree with obstacle avoidance takes little more time.

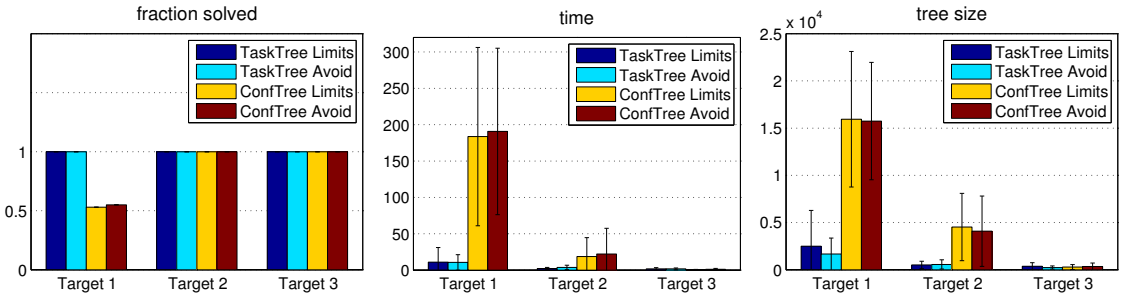

Figure 6.11: Setup A with 3D hand position task space. Fraction of solved queries, average runtime and tree size grouped by target positions (1, 2, 3), planning algorithm (TaskTree, ConfTree) and local control redundancy resolution method (Joint limit avoidance, obstacle avoidance).

Table 6.1: Results for setup A for the 3D hand position task space.

Algorithm	Redundancy	Task Space	Target	Solved	Time	Tree Size
TaskTree	Joint Limits	3D	1	1.00	10.90 (20.19)	2496 (3787)
			2	1.00	2.15 (1.56)	512 (395)
			3	1.00	1.67 (1.66)	367 (386)
	Obstacle Avoidance	3D	1	1.00	10.58 (10.60)	1670 (1692)
			2	1.00	3.73 (2.98)	564 (486)
			3	1.00	1.71 (1.28)	236 (166)
ConfTree	Joint Limits	3D	1	0.53	183.57 (122.67)	15943 (7173)
			2	1.00	18.64 (25.95)	4529 (3566)
			3	1.00	0.53 (0.43)	287 (259)
	Obstacle Avoidance	3D	1	0.55	190.71 (114.44)	15746 (6217)
			2	1.00	22.18 (35.26)	4087 (3726)
			3	1.00	1.17 (1.10)	344 (368)

The second target positon, target 2, shows a clear advantage of TaskTree. The runtime is lower if the joint limit avoidance control is used. For the last target position 1, findings are similar. ConfTree takes more time and larger trees to find a solution and the greater part of queries could not be solved at all. Comparing local obstacle avoidance control to joint limit avoidance control, there is again an advantage of joint limit avoidance control visible, to a greater extend regarding the runtime and less for the tree size, which is almost equal.

Figure 6.15 shows results from (Behnisch et al., 2010) concerning the spatial distribution of the time needed to solve a planning query. The runtime is shown individually for every query position on the table, refer to figure 6.8. For the configuration space with task space bias planning it is visible that the runtime on the left side of the table is higher as with task space planning. Comparing the obstacle avoidance with the joint limit avoidance condition, two observations can be made. The runtime is larger for the 5D task and joint limit avoidance case, which is also reflected by the low count of successful runs. For the 3D joint limit avoidance case, less positions could be solved.

6.7 Discussion

In setup A, there are major differences among the three target positions, thus they are best analyzed individually. Target 3 is easy to reach and planning queries are fast and reliable solved. Depending on the task space, a solution typically takes 2 or 3 seconds and there are

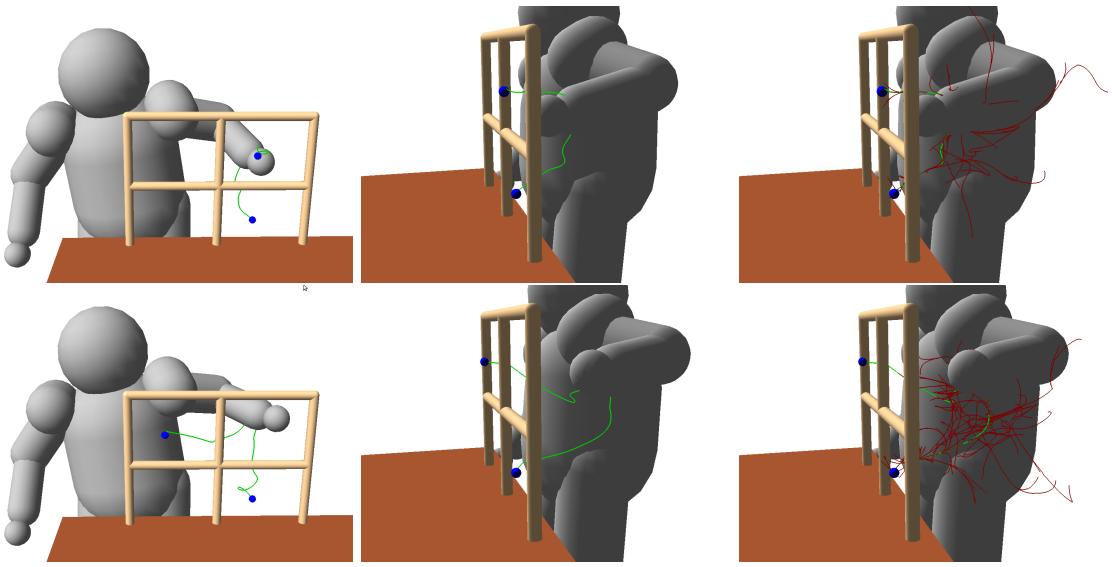

Figure 6.12: Two examples of planned trajectories for target 3 (top) and target 1 (bottom), using the hybrid motion planning algorithm (TaskTree). Task space is the 3D position of the left hand. Solutions are shown in green and the respective search trees are shown in red.

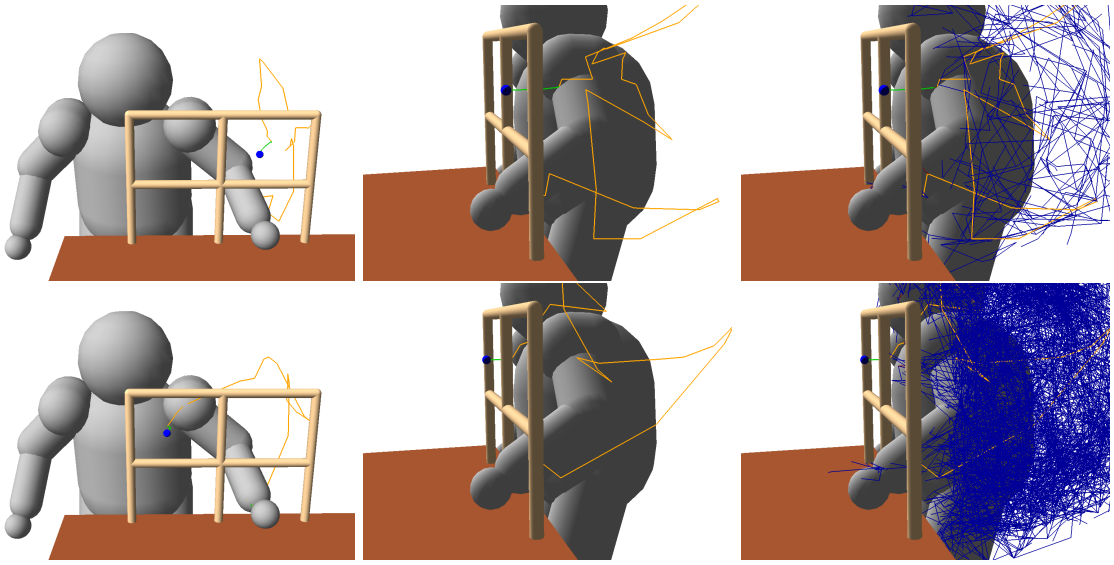

Figure 6.13: Two examples of planned trajectories for target 3 (top) and target 1 (bottom), using the classical RRT based algorithm (ConfTree) with task space biasing in the 3D position space of the left hand. All parts of the solution that are planned in the configuration space are drawn in orange and task space goal biasing parts are drawn in green. In the search tree, configuration space parts are blue and task space parts are red.

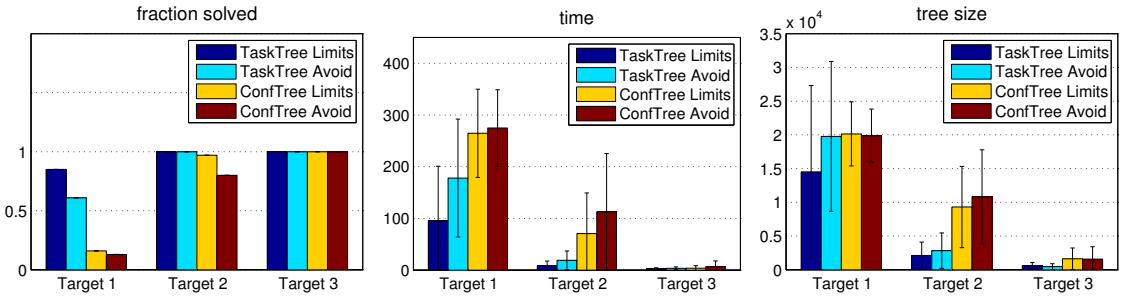

Figure 6.14: Setup A with 5D hand position and orientation task space. Fraction of solved queries, average runtime and tree size grouped by target positions (1, 2, 3), planning algorithm (TaskTree, ConfTree) and local control redundancy resolution method (Joint limit avoidance, obstacle avoidance).

Table 6.2: Results for setup A for the 5D hand position and orientation task space.

Algorithm	Redundancy	Task Space	Target	Solved	Time	Tree Size
TaskTree	Joint Limits	5D	1	0.85	95.49 (105.30)	1452 (1279)
			2	1.00	8.91 (8.58)	212 (198)
			3	1.00	2.64 (1.91)	60 (46)
	Obstacle Avoidance	5D	1	0.61	178.00 (113.98)	1977 (1110)
			2	1.00	18.92 (18.14)	283 (263)
			3	1.00	3.42 (2.91)	47 (43)
ConfTree	Joint Limits	5D	1	0.16	264.58 (85.29)	2014 (475)
			2	0.97	70.64 (78.64)	930 (602)
			3	1.00	3.71 (4.80)	163 (159)
	Obstacle Avoidance	5D	1	0.13	274.73 (74.12)	1987 (393)
			2	0.80	112.79 (112.55)	1082 (696)
			3	1.00	6.51 (11.16)	157 (182)

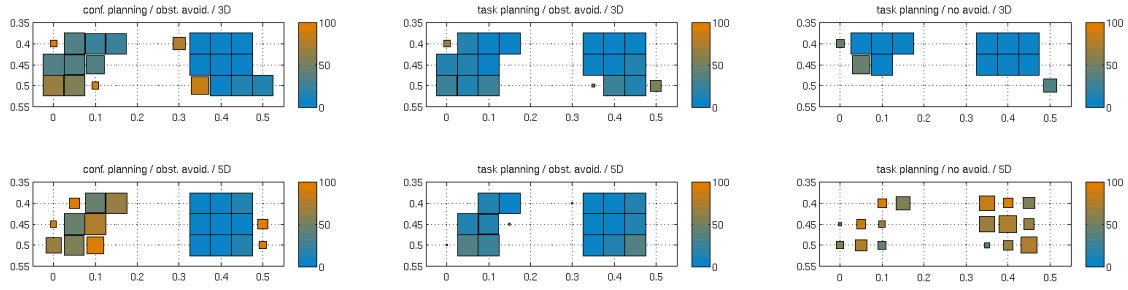

Figure 6.15: Spatial distribution of solution times, setup B, from Behnisch et al. (2010). Time is colorcoded from blue (0 sec.) to orange (100 sec.) and the size of the rectangles encodes the fraction of successful planning runs, smaller means fewer runs are successful. From left to right: Configuration space planning with task space bias, task space planning with obstacle avoidance and task space planning with joint limit avoidance. Top row: 3D hand position task space. Bottom row: 5D hand position and orientation task space.

no big differences between algorithms and control methods. Solutions for targets 1 and 2 take longer time and TaskTree is significantly faster, with bigger difference for the 3D task and less for the 5D task.

The differences between the target positions imply a large variance in planning difficulty. Target 1 is indeed only affected by minor obstacle constraints and the path is relatively short. The solution paths of targets 2 and 3 are longer and also have to get closer to both obstacles and to the robot body itself, only leaving narrow passages of valid motions. The increasing difficulty results in gradually increasing runtime of the planning algorithms. While the runtime of the TaskTree method only increases moderately, the time consumption of the ConfTree shows a steeper increase, up to a frequent exhaustion of the time limit for target 1. If the 3D and the 5D task space are compared for the TaskTree algorithm, the latter takes more effort, visible by greater increase in runtime but still remains below the high time consumption of ConfTree.

Overall the task space search as done with TaskTree is able to find solutions faster, although each tree extension is more costly. The ratio of speedup compared to configuration space search is dependent on the dimensionality of the task space. The 3D task space has the lowest dimensionality and the lowest search times, followed by th 5D task space. These both task spaces result in a significant lowering of dimensionality for sampling, compared to the 9D configuration space, which can be transformed into a planning speedup, as shown by the results. A speedup is possible although each iteration of the search is more costly once the local control framework is involved.

Setup B compares the algorithms on a spatial scale and shows that the differences in runtime are differently distributed. On the left side of the table the advantage of task space search becomes clearly visible, while on the right side differences are smaller. In analogy to the results from setup A, one can see that the behavior of both algorithms varies with the actual planning query. The additional effort for hybrid planning does not always pay off in decreased runtime, but a somehow challenging setup is required. It is important to note however, that even quite simple looking tasks can be challenging enough, for example all the planning queries that involve passing obstacles and the robot itself. The only easy queries in this sense happen if there is only one obstacle, a short distance and a large free space.

The choice of the local planning method has an ambivalent effect. Redundant space obstacle avoidance has a positive effect for Target 1 with the 3D task, where it results in a smaller search tree. Unfortunately, the smaller tree does not result in lower runtime, since every tree extension step has a larger cost. When the 5D task space is used, a negative effect of obstacle avoidance is evident for target 2 and especially target 1, where the search trees a larger, resulting in a higher solution time. However, a clear advantage of redundant obstacle avoidance is visible in setup B, where some positions are only reachable when obstacle avoidance is used and the solutions involving obstacle avoidance tend to be faster for some target positions.

Chapter 7

Bi-Manual Hybrid Planning

How hybrid planning could be used for redundant robots was shown in the previous chapter. The feasibility of the concept was proven with a simulated humanoid robot, but specific capabilities of humanoids were not addressed, in particular, the possibility to utilize two arms for bi-manual interactions. In this chapter, a motion planning experiment is described that realizes a complex bi-manual task for a humanoid robot in a real world lab environment.

Section 7.1 introduces the setup and motivates the use of hybrid planning in comparison to other bi-manual planning approaches, described in section 7.2. The actual task representation and hybrid planning algorithm are explained in sections 7.3 and 7.4, respectively. Sections 7.5, 7.6 and 7.7 deal with the realization and evaluation of the experiment.

7.1 A Humanoid Motion Planning Experiment

Morphological similar to humans, humanoid robots are able to cope with a large class of manipulation problems. Especially, the possibility to utilize two arms and two hands greatly enlarges their handling skills, compared to single arm systems. In this chapter, a hybrid planning solution for a bi-manual motion planning task for humanoid robots is developed and evaluated in a lab setting.

Since planning is done in the physical world now, it is necessary to build a reasonably accurate model of the robot and the environment to be able to test for collisions. While the robot state is known with an adequate precision from joint motion encoders, the acquisition of the world state is a difficult problem. An external solution is used here, where a motion tracking systems computes the position of objects with the help of multiple cameras.

The experiment is designed in a way that introduces a lot of variance into the process. Figure 7.1 shows the experimental setup in the lab. The humanoid is holding a basket in its left hand and the task is to put the flower held with the right hand inside the basket. Locations of the objects are not given beforehand but are instead allowed to vary. The basket can be grasped at any position at the handlebar and the same is true for the flower, there is a large number of possible grasping positions. Also, the target position of the flower inside the basket is not fixed. Before each planning run, a human demonstrator shows where the flower should be placed. Due to these three sources of variance, each planning run has to be solved under different conditions. The ability to handle these changing conditions is a crucial requirement for motion planning methods operating in real world environments.

7.2 Bi-Manual Motion Planning

In the motion planning literature, works about bi-manual manipulation planning often tackle the particular problem of robot arms that form a closed kinematic chain, connected by a jointly grasped object. This problem instance is especially hard to solve, because the chain-closure constraint restricts the set of valid postures to a manifold in configuration

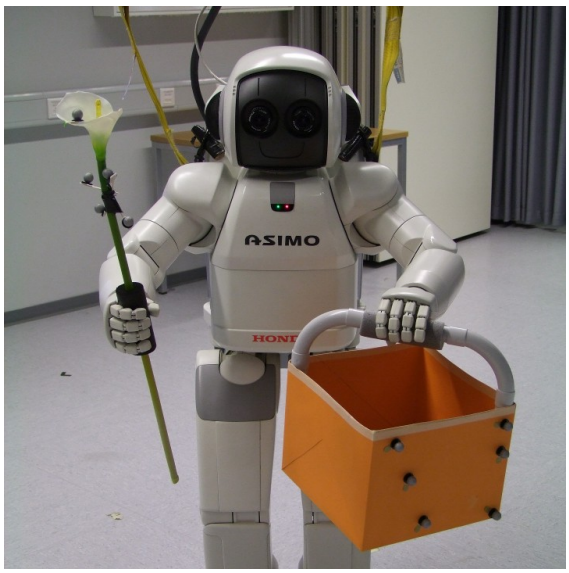

Figure 7.1: Objective of the bi-manual planning task is to place the flower, held with the right hand, into a basket, held with the left hand. The desired goal position of the flower inside the basket is not fixed, but determined by a human demonstrator. Also the grasp positions of the flower and the basket can vary. Once the left hand holds the basket, the left arm and the handle bar constraint the possibilities for collision free paths of the flower into the basket to three homotopy classes of paths. Depending on the goal position, the right homotopy class has to be picked, thus global search is really necessary for this setup.

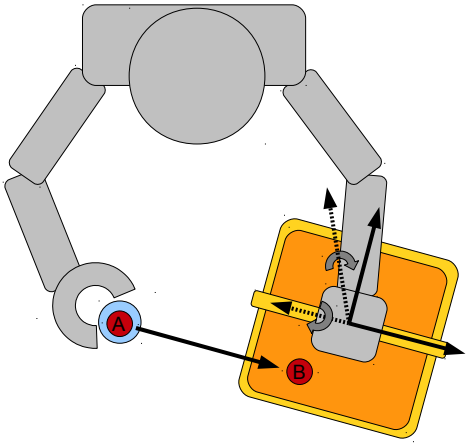

Figure 7.2: Task model: Control the position and orientation (5D) of a reference point fixed to the right hand (A) relative to a point (B) fixed to the coordinate frame of the left hand until the two points coincide. While the right hand can move freely, the left hand movement is constrained. Translations can only be done in a horizontal plane of constant height. Rotations are allowed around the normal axis of this horizontal movement plane and around the grasp axis of the hand, i.e. around the handle bar of the basket.

space. In a classic sampling approach over the whole space, the probability of hitting these valid configurations is zero. In (LaValle, 2006, chapter 7), some specialized approaches for planning with closed loops are summarized.

The problem of bi-manual planning where both arms are not connected is not often explicitly addressed. In this case, there are no chain-closure constraints and planning could in principle be done with a standard configuration space sampling approach. However, since the space contains both arms now, the configuration space often becomes very high-dimensional and directly sampling the full space quickly becomes very expensive. To tackle this difficulty, it is beneficial to have a closer look at the particular problem instance.

Due to the tree-like kinematic structure of multi-arm robots, it is possible to decompose the robot into several independently moving sub-robots. Planning is then equivalent to multi-robot motion planning, a problem often solved with decoupled approaches, (LaValle, 2006, chapter 7). For each sub-robot, an independent partial plan is produced in its individual configuration sub-space, which is typically less complex. Building on the partial component solutions, an overall solution can be orchestrated by considering the interactions between the partial plans.

A planning algorithm for multi-arm systems, operating with this decomposition technique, is presented in (Gharbi et al., 2009). In a first stage, a probabilistic roadmap is build for each robot arm in a preprocessing phase. To solve a specific planning query, the individual arm solutions are coordinated by composing the super-graph over all arm roadmaps in the second stage of the algorithm. It is shown that it is possible to achieve a performance gain with this two-stage approach. The work in (Vahrenkamp et al., 2010) implements bi-manual trajectory planning with a focus on grasping, using an approximate Jacobian transpose task space control method to bias the search. Their method also decouples the problem by planning each arm independently, but no further effort is taken to synchronize both partial solutions, planning is just repeated until two collision free trajectories are found. The approximate task control operates on each arm separately as well.

Although the decoupling approach would be able to solve the bi-manual planning problem of the described experiment, one can assume that the second coordination step can become difficult. The workspaces of both arms are heavily overlapping and the chance of collisions between the arms is high.

To cast the bi-manual planning problem in terms of our hybrid planning approach, a suitable task space representation has to be defined first. An intuitive way to describe the task is to provide a desired position of the flower relative to the basket. Since both objects are rigidly[1] connected to the hands, this can be transformed into the position of one hand relative to the other. Global planning can then be done on the relative hand position task space, leaving the simultaneous arm movement and coordination to be solved on the local planning layer. Thus, the high dimensional planning problem can be effectively reduced to a low dimensional search of the task space. The remaining freedom in the redundant space can be utilized to overcome the severe obstacle constraints in this setup by employing local obstacle avoidance motion optimization. Another advantage of local obstacle avoidance is the ability to counteract possible disturbances by imprecise object localizations.

7.3 Task Representation

In order to employ the hybrid motion planning method for the given setup, a suitable task space has to be defined. Given a control target in this task space, the Whole Body Motion

[1] A perfectly rigid connection can not be guaranteed in practice, but this can be counteracted by tracking the object movement relatively to the hand movement.

Control framework (chapter 2.5) is used to translate the control inputs to full configuration space motions of the upper body.

The setup is shown in figure 7.1. The goal for planning is the placement of the flower inside the basket, what can be described in terms of the position of the flower relative to the basket, equivalent to the position of the right hand relative to the left hand. Figure 7.2 schematically illustrates the spatial relationship. The position and orientation of a reference point described in the coordinate system of the right hand (A) is controlled with respect to a fixed point in the coordinate system of the left hand (B). This relative position defines the task space control input, consisting of the 3D spatial position and the 2D grasp-axis orientation representation, summing up to a 5D task space similar to the space in 6.5.1.

Beside these task constraints, the movement of the right hand is not further restricted. The movement of the left hand however is limited by some constraints. The first constraint fixes the vertical position of the hand, i.e. the hand has to stay in a predetermined horizontal plane. This is illustrated with the two black arrows in figure 7.2, spanning the plane of allowed motion. The second constraint fixes the orientation of the left hand, as denoted by the two dotted axes. Rotations are only allowed around the normal axis of the horizontal movement plane and around the grasp axis of the hand, collinear to the handlebar of the basket.

Due to the relative definition, the task acts on both hands. Imagine that the task involves to move both hands closer together. The controller has three possibilities to achieve this, either to move only the left or the right hand or alternatively to move both hands at the same time. The task itself does not constrain this, the definition leaves some redundancy that can be exploited by the controller. In terms of the concrete setup this means that it does not matter where exactly the task is done, the robot can hold the basket close to the body or further away, more to the left or more to the right, as long as the goal position of the flower relative to the basket is achieved. More freedom in task execution is gained which can be usefully exploited by the control framework by defining secondary motion objectives.

7.4 Planning

Planning is done with the hybrid planning algorithm from chapter 6. The same algorithm can be used because the change of task representation does not change the procedure. It affects the sampling of task positions, task space distances and the local planner, these parts have to be adapted to the task at hand. For the bi-manual task, changes involve mainly the local planner. Sampling and distance metrics can be reused from the 5D position and orientation task from section 6.5.1, the only difference is the coordinate frame of reference.

The difference between the current planning setup and the setup in chapter 6 is the involvement of both arms of the robot. This is reflected in the new task space definition and also by a different configuration space. Adding the right arm, the dimensionality of the space increases from 9 to 14 dimensions. Since the task space is of the same dimensionality, 3 position dimensions and 2 orientation dimensions, the redundant space for the bi-manual task is larger.

Local planning again utilizes the whole body motion control framework of section 2.5, adapted to the new task representation. Redundancy resolution employs the joint-limit avoidance and obstacle avoidance cost functions. The effect of the redundant space obstacle avoidance for the presented task are illustrated in figure 7.3. If the basket gets too close to the robots body or the obstacle, avoidance motions are done as indicated. The high-dimensional configuration space is effectively re-arranged into a low-dimensional relative positioning task space and a large redundant space, leaving room for local obstacle avoidance.

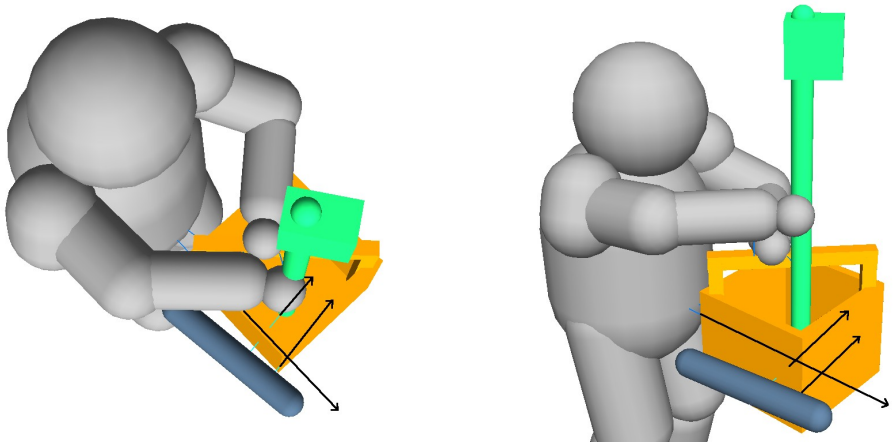

Figure 7.3: Illustration of redundant space obstacle avoidance. Minimizing the obstacle avoidance costs results in a movement away from obstacles. Since the movement is limited to the redundant space, task execution is not affected.

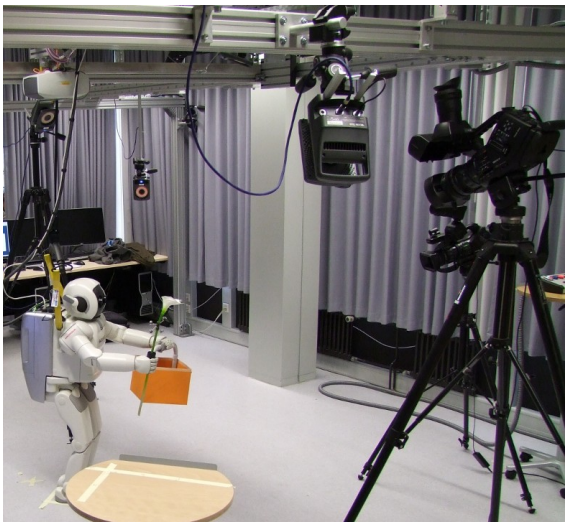

Figure 7.4: Setup of the experiment. The robot is surrounded by 8 Vicon cameras. Two HD cameras are used for recording.

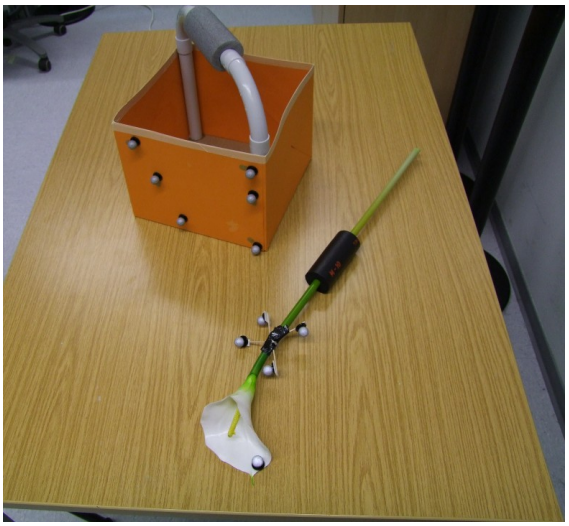

Figure 7.5: The flower and basket used in the experiment, with the infrared reflective Vicon motion tracking markers attached.

7.5 Setup

Figure 7.4 shows the setup in the lab during the experiment. The humanoid robot is holding the two objects, flower and basket, which are tracked with a Vicon motion capturing system[2]. Motion capturing operates based on infrared reflective markers, whose spatial location can be estimated by triangulation from multiple different views. The scene is observed by 8 infrared cameras and once a marker is visible in at least 3 views, its location can be robustly tracked. A closer view of the objects in the setup is given in figure 7.5, where the tracking markers a clearly visible.

The object tracking system can be used in two stages. At first, it is needed for determining the positions of the robot and the objects relative to the robot in a a-priori model building stage. Planning operates on this environmental model. The second stage can be activated during plan execution in order to update object positions and to recognize plan discrepancies,

[2]http://www.vicon.com

for example if objects are slipping in the hand. The updated positions can than be incorporated in the plan execution. To include object tracking into the hybrid planning simulation framework, the "World Model Interface" is enhanced to allow to directly set the transformations of certain objects from outside, here from the Vicon motion capturing system.

The course of the experiment can be divided into five phases, with the first four phases depicted in figures 7.6 to 7.9. Starting in its rest position, the robot grasps the basket in the initial phase, as shown in figure 7.6. The position of the basket is tracked because the handle-bar can be grasped in different ways and thus the position of the basket can substantially vary. Next step is the demonstration where exactly the flower should be placed inside the basket, carried out by a human instructor, figure 7.7. The flower is tracked and its position relative to the left hand and the basket is stored as the current planning goal. Once the target position is determined, the robot grasps the flower with the right hand, while a human operator is handing over the flower, as shown in figure 7.8. Again the flower position is tracked because the location where the flower is grasped is not pre-determined. In this picture, a view of the simulation visualization can be seen on the display in the background, used to monitor the object tracking. The current state of the simulation is then used in the next phase to plan the motion, employing the hybrid motion planning framework as described. Starting point for planning is always a posture similar to the one in figure 7.9. During planning, the object tracking is paused in order to avoid interference. If the plan is executed in the fifth and final phase, the second object tracking stage is activated, updating the positions in an online fashion. Displacements of the flower and basket can be detected and compensated during execution.

7.6 Results

Two examples of the execution of a solution trajectory, the last phase during each run of the experiment, is shown in figure 7.10. The whole process was repeated a couple of times and all solution trajectories were successfully executed. However, due to the large variety in object and target positions in the course of the experiment, there are no meaningful statistics available for the lab experiment. In order to gain insight into the capabilities of hybrid planning for this setup, a small study was conducted in simulation only.

Figure 7.11 show three flower target positions, two on the right hand side of the handle-bar and one on the left hand side. The initial position prior to planning is the same as in the real world experiment. For each of the target positions the hybrid motion planning algorithm is run with two different local planning conditions, once with redundant obstacle avoidance and once without. Also, two different setups of the environment are used, the plain setup as shown in figure 7.11 and a setup with an additional obstacle, placed as in figure 7.3. Example trajectories for target positions 2 and 3 can be found in figures 7.12 and 7.13.

Statistics gathered during 100 runs for each position are summarized in table 7.1. These are the fraction of solved queries, i.e. runs with a runtime lower than the limit time of 200 sec., the average runtime and the average tree size. The averaged values also contain the trials which were not successfully solved in the available time.

The average runtime varies substantially between the three targets. Without the additional obstacle, target 1 is always solved with a relative low average solution time of around 5 sec. Target 2 needs more time to be solved and comparing the two local control approaches, the runtime is lower if no obstacle avoidance is used. The same can be seen for target 3. The runtime is much higher as for the other two targets and local control without obstacle avoidance performs better. Looking at the second setup with an additional obstacle, only

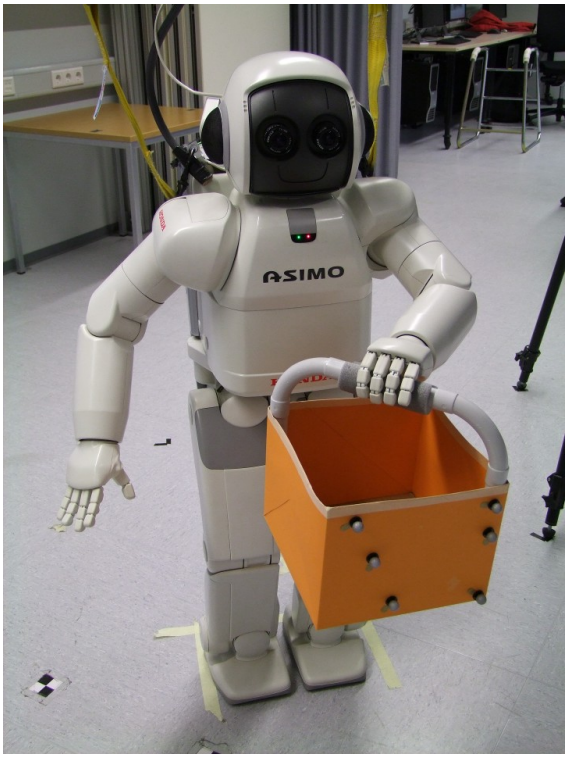

Figure 7.6: Initial phase: The robot holds the basket in its left hand. The position of the basket is tracked.

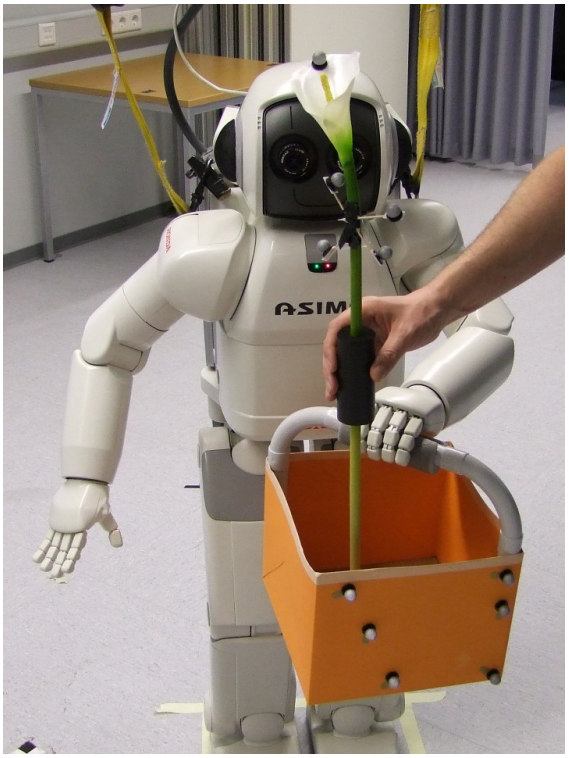

Figure 7.7: Demonstration phase: A human demonstrator shows where the flower should be moved to. The end position of the tracked flower movement is stored as the goal position.

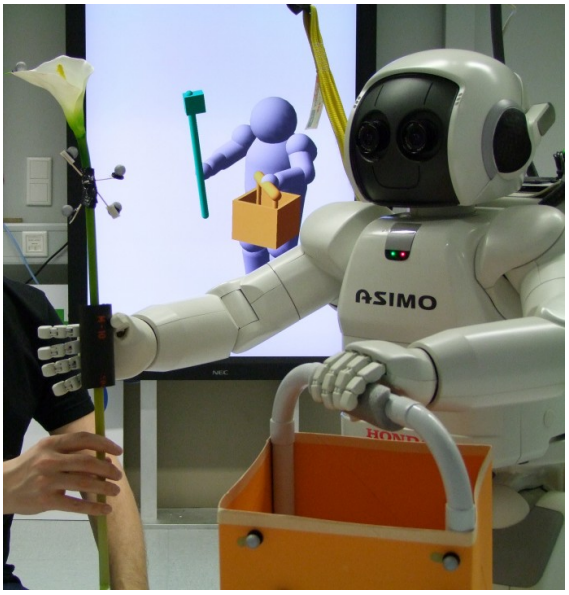

Figure 7.8: The robot grasps the flower, with the human demonstrator holding the flower into the right hand. Again the precise position of the flower is tracked. In order to monitor the tracking, the screen in the background shows the current state of the simulation with realtime position update of all tracked objects.

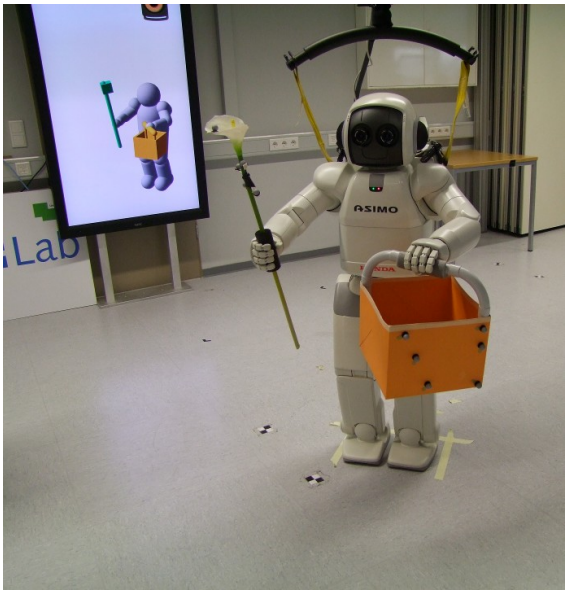

Figure 7.9: Planning phase: Holding the flower and the basket is the start configuration for planning. The simulation, containing the robot and all tracked objects, is used for planning as well. Object tracking is paused during planning but activated again during execution of the planned movement. This way, displacements of the flower and basket relative to the robots hands can be compensated with reactive obstacle avoidance.

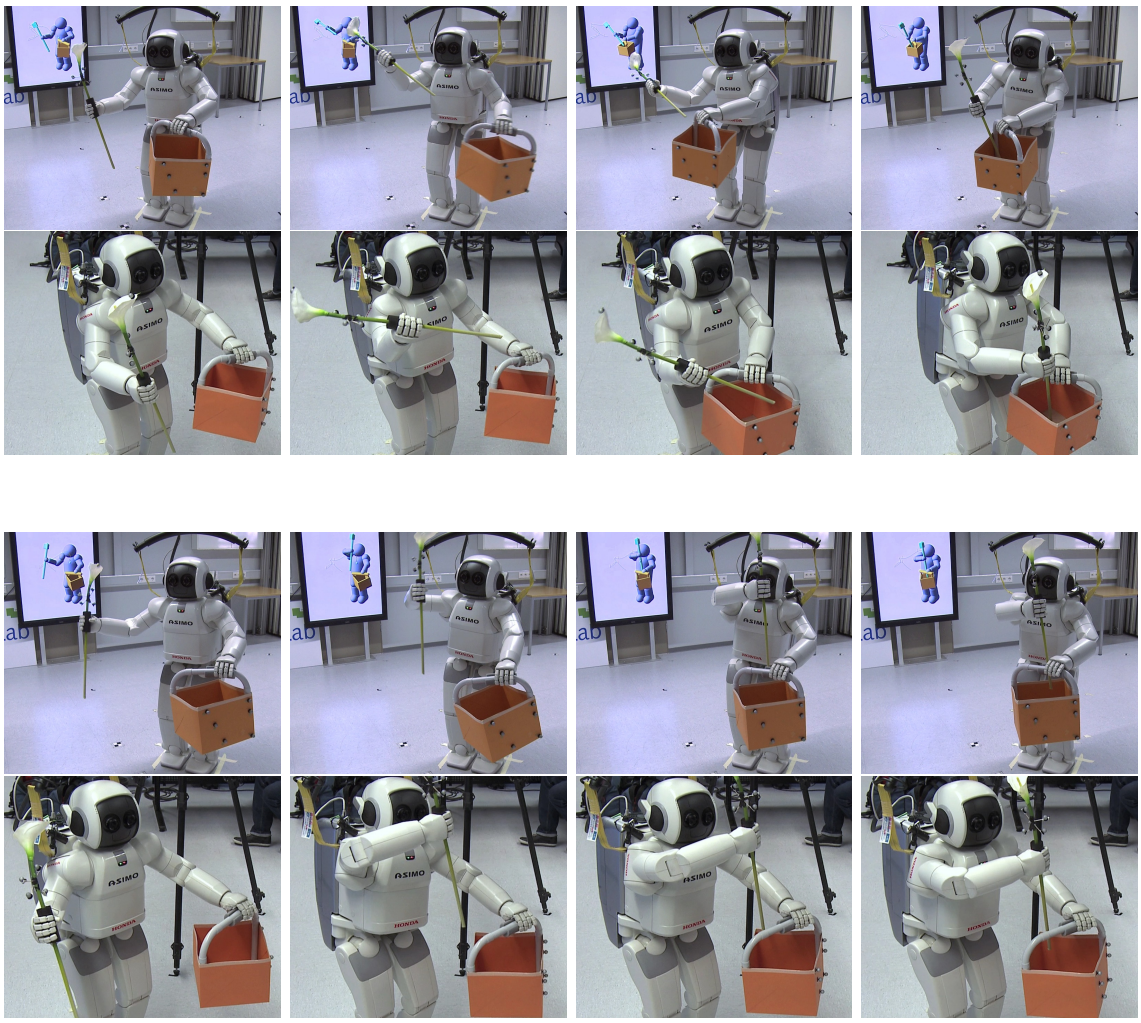

Figure 7.10: The humanoid during execution of two solution trajectories for two distinct flower target positions.

Figure 7.11: Simulated motion planning for three distinct flower target positions. From left to right target positions 1, 2 and 3.

Plain Setup						
Redundant Space Obstacle Avoidance				No Obstacle Avoidance		
Target	Solved	Time (σ)	Tree Size (σ)	Solved	Time (σ)	Tree Size (σ)
1	1.00	5.19 (4.82)	1107 (1000)	1.00	5.47 (4.92)	1266 (1119)
2	1.00	14.73 (17.01)	2963 (2880)	1.00	7.09 (5.86)	1620 (1288)
3	0.91	59.17 (59.63)	8819 (6887)	0.97	48.85 (50.18)	8015 (6406)
Setup with Additional Obstacle						
Redundant Space Obstacle Avoidance				No Obstacle Avoidance		
1	0.95	43.92 (51.64)	6099 (5752)	–	–	–
2	1.00	11.44 (9.31)	2036 (1569)	1.00	7.90 (8.97)	1568 (1652)
3	–	–	–	–	–	–

Table 7.1: Simulation results for three fixed target positions, showing: Fraction of solved queries, average runtime (in sec.) and average tree size.

target 2 could be reliably solved using both local planning methods. Target 1 is only solved if obstacle avoidance control is used. Target 3, in contrast, was not solved at all.

7.7 Discussion

The lab experiment was successfully done as devised. Hybrid planning was able to find solution for most of the desired target positions using the task representation especially defined for the setup. Once a solution was found, it was executed without problems. The large amount of variability in the setup, in consequence of the flexibility of grasp and target positions, was successfully handled.

The results of the simulation study show a large variability, depending on the target position. While some targets are solved with constantly low runtimes, others need considerable more time. Comparing the two local planning methods, the results are again ambivalent, supporting the findings in chapter 6. The use of obstacle avoidance control does not necessarily reduce solution times, in fact it leads to higher times for 2 of the 3 targets. This is reflected in the tree sizes as well, which tend to be larger for obstacle avoidance control. The enhanced local planning abilities increase the probability of successful explorations in the neighborhood of obstacles. Some of these explorations contribute to the solution, but others do not, exploring portions of the space that are not accessible without obstacle avoidance. As a consequence, the overall number of tree nodes gets larger.

Looking at the setup with the additional obstacle however, an advantage of using obstacle avoidance can be seen. While target 2 is again not impacted in a positive way, target 1 can only be solved with obstacle avoidance. Figure 7.14 explains why this is the case. If the redundant space motion is based on joint-limit avoidance only, the optimal posture results in a collision of the basket with the obstacle. When obstacle avoidance is added, a different collision free posture can be found that minimizes the obstacle costs as well.

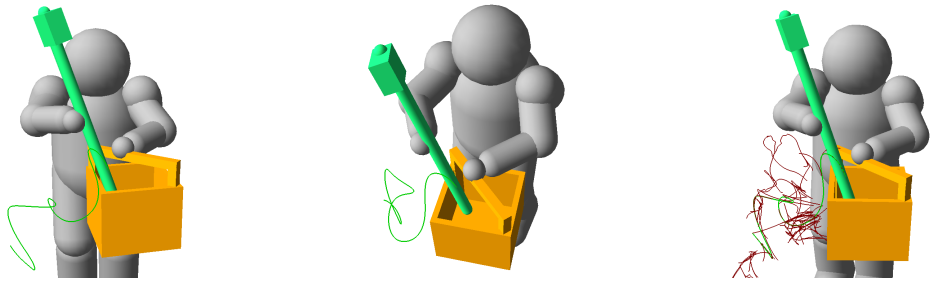

Figure 7.12: Example of a solution path to target position 2. The flower trajectory is shown in green and the search tree in red.

Figure 7.13: Example of a solution path to target position 3. The flower trajectory is shown in green and the search tree in red.

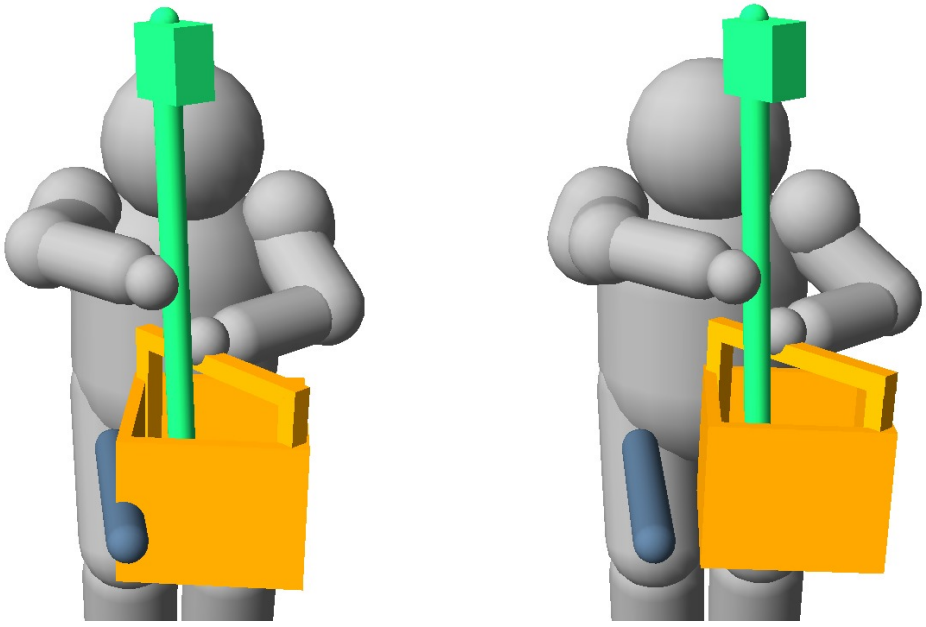

Figure 7.14: Local planning without obstacle avoidance does not succeed in finding a valid solution (left). Joint-limit avoidance as the only secondary motion criteria drives the trajectory into the obstacle. With obstacle avoidance, the trajectory is modified to stay away from the obstacle (right). The task trajectory is identical for both cases, only the redundant part of the motion is changed.

Chapter 8

Relaxed Control Hybrid Planning

In the previous chapters, the applicability of hybrid planning for redundant robots was demonstrated. Local planning utilized the redundant space to avoid joint-limits and obstacles, but task space trajectories were always performing straight target directed movements. For motion planning, task constraints often do not need to be as strict and weakening them can be useful. This is for example the case if a point-to-point movement has to be planned, where the shape of the resulting trajectory is not crucial. In this chapter, the relaxed motion control from chapter 3 is integrated into the hybrid planning framework. This enables the task space local planning process to stronger contribute to overall planning by adapting task trajectories according to additional objectives, like the avoidance of obstacles.

After an introduction in sections 8.1 and 8.2, section 8.3 describes the integration of relaxed control in the previously developed planning framework. Further, specialized sampling and node weighting strategies are developed that aim to utilize specific properties of the new local planning approach. All together, relaxed control hybrid planning is evaluated in sections 8.4, 8.5 and 8.6, in a simulation study for a simple planar manipulator. Section 8.7 discusses possible enhancements to find optimal solutions.

8.1 Task Space Local Planning

The hybrid planning methodology developed in the previous chapters benefits from two key features, global sampling of a condensed task representation and utilization of local optimization capabilities. The impact of choosing different task spaces was shown and local planning successfully exploited the redundant space for the fulfillment of additional motion criteria. Motion generation in the task space was based on a dynamical system approach, producing a strictly goal directed trajectory with the single objective of reaching the target on the shortest path. Other objectives were not included into the task space planning process. The relaxed control method of chapter 3 however illustrated that the task space can also be used for local planning. More specifically, by weakening strict task constraints and allowing deviations from greedy task achievement, for example to avoid obstacles, local planning gains freedom for more powerful motion optimization.

In this chapter, the proposed relaxed control method is integrated into the hybrid planning framework. In contrast to the hybrid planning applications in chapters 6 and 7, local planning modifies both the task space and the configuration space trajectory. The idea is illustrated in figure 8.1. As before, global planning searches the task space and the configuration space motion is generated by employing a local control method, choosing a trajectory according to secondary motion objectives like obstacle avoidance. If relaxed motion control is used as the local planner, not only the configuration space trajectory is optimized but the task space trajectory as well. By regarding the same criteria in both spaces, task space and configuration space local planning can work hand in hand for a greater optimization success. For example, if obstacle avoidance is a secondary motion objective, the task space trajectory is actively adapted to move away from obstacles.

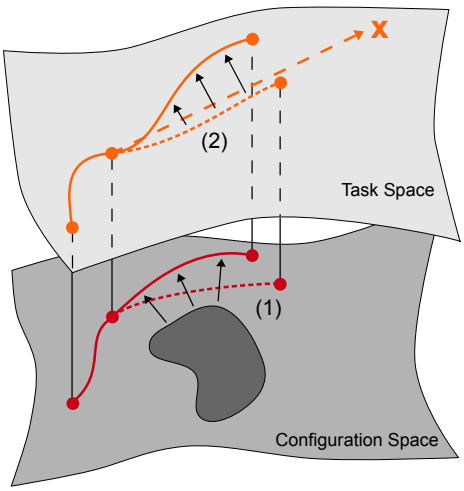

Figure 8.1: Hybrid planning with local trajectory optimization. If the local optimization during configuration space trajectory generation (1) is allowed to go beyond the redundant space, the task space trajectory (2) is also affected. For example if an obstacle distance potential is used, pushing the configuration space trajectory away from obstacles, the task space trajectory is deviated away from obstacles as well. Not restricted to the redundant space, local optimization can have a larger impact, but also can result in a substantial deviation of the task trajectory away from the target.

Using such an enhanced local planning method can reduce the amount of global sampling needed, in particular in setups containing some distinct local structures. Such structures could be narrow corridors for instance, which are difficult to solve with general sampling approaches, as discussed in section 4.5. A local planner with task space obstacle avoidance however is able to locally guide the trajectory through the corridor, while a straight line local planner needs to sample its way through. This can become quite difficult, because depending on the narrowness of the corridor, the probability to place a sample inside the corridor decreases.

Relaxed control can be forced to always converge to the desired target position, if not prevented by constraints, even if costs are increasing. This property is useful in the context of motion planning, because it allows local planning to disrespect cost structures and to plan trajectories that enter higher cost regions. Without this property, regions with higher costs than the starting region would never be explored and it would be impossible to reach a uniform coverage of the space. Consequently, the optimization of secondary movement costs is thus not always converging to lower costs, but dependent on the placement of target points.

The changed task space local planning behavior has an effect on the operation of the global planning component. Although it is possible to include the new control method into the sampling-based hybrid planning algorithm without further changes, an explicit treatment of the specific behavior offers new possibilities. The success rate of the local planner is higher in regions where the local structure can be utilized, for example in the mentioned narrow corridor situations, since the trajectory generation is guided to prevent constraint violations. These regions can then be explored with fewer samples than regions where such a local planning speedup is not possible. This implies that the global sampling process can be improved to focus sampling in regions where it is needed and to prevent an over-sampling of regions where local planning successfully guides the exploration. Another local planning property that can be utilized during global planning is the cost optimization capability. If the search is focused on trajectories with lower costs, the overall solution costs are lower too.

Since the sampling-based framework builds on EST, which is a very flexible tree search algorithm allowing to exchange individual components and to use arbitrary node weighting schemes, it is simple to incorporate different search and sampling behaviors that account for this new possibilities. The search should try to make good use of local planning to speedup

search progress. As described in the next section, the term *exploitation* is often used in the literature to depict the ability to utilize as much local information as possible. *Exploration*, on the other hand, describes the ability to discover the global connectivity of the space, aiming to eventually cover the whole space. Of course exploitation alone does not suffice in cases where a global scope is needed and the challenge is to carefully balance between local exploitation and global exploration.

8.2 Exploration and Exploitation

Three different classes of approaches for motion planning can be distinguished that, either explicitly or implicitly, utilize local exploitation and differentiate global exploration.

The first class of methods are sampling-based roadmap planners, aiming to capture the global connectivity of the free space with a minimum number of graph edges. The resulting roadmaps form a minimum representation of the exploration necessary to cover the space, where all edges not contributing to the global exploration are pruned. These redundant edges only exploit local structure and can be omitted during preprocessing, since it can be assumed that they can be trivially reproduced again in the query phase. The Medial Axis Roadmap (Wilmarth et al., 1999) for example retracts edges to lie on the medial axis in the free space between obstacles, forming a minimal skeleton representation using the geometric property of obstacle distance. During planning, a query node can be easily connected by exploiting the free space while all global exploration is already captured. This relatively simple geometric argument is taken further in (Jaillet and Simeon, 2008) by defining smooth higher order deformations between edges as local exploitation. By filtering a roadmap to only contain edges not deformable up to a certain order, a minimal exploration representation can be obtained.

While these roadmap approaches filter local exploitation to reduce the number of edges to a minimum necessary to capture the space, another family of methods operates by suppressing an excess exploitation during the growth of a single-query sampling tree. Some algorithms that prevent an over-proportional sampling of individual nodes (Cheng and LaValle, 2001; Yershova et al., 2005) or regions (Ladd and Kavraki, 2005; Sucan and Kavraki, 2009a) were presented in section 4.5. By maintaining different measures of localized state space coverage, these methods actively guide exploration towards less covered regions.

In contrast to the former two classes that focus on state space exploration, a third class of methods follows the reverse approach of focusing on exploitation and only performing exploration when necessary. The decomposition approach of Rickert et al. (2008) (refer to section 6.1) exploits local workspace properties by following a potential function defined in a free space tunnel towards the goal. If this exploitation fails, the search is gradually shifted to exploration by sampling inside the tunnel. If exploration is successful, the search mode is shifted back to exploitation again. The Transition RRT (Jaillet et al., 2010) treats planning as the optimization problem of finding a low cost path on a globally defined cost function. It uses a temperature parameter commonly found in stochastic optimization to control the acceptance or rejection of states based on a Boltzmann distribution. It is desired to keep the path in low costs regions with a low temperature and only if that fails the temperature is raised and higher costs states are accepted too. The temperature is automatically balanced as a function of state rejections, with the two objectives of maintaining low costs and of preventing an excessive refinement of regions not at the frontier of the tree. Berenson et al. (2011) observes that the approach of cost minimization by sampling has difficulties finding narrow low-cost chasms, because sampling those passages directly is relatively unlikely. They propose to combine the Transition RRT with local gradient descent steps, exploiting the cost

Algorithm 8.1 Hybrid Task Space Tree Growing with Greedy Goal Bias (TaskTree)

tree with nodes $p_i = (x_i, q_i, u_i, t_i, w_i) \in T = (\mathcal{X}, \mathcal{C}, \mathcal{X}, \mathbb{R}, \mathbb{R}^d)$
metric $\rho(x, y), x, y \in \mathcal{X}$
while not goal reached **and not** time over **do**
  if goal bias **then**
    p_s = nearest tree neighbor to goal with $\min_{x \in T} \rho(x_{goal}, x)$ not yet goal biased
    $u_{tgt} = x_{goal}$
  else
    p_s = select tree node based on weights $w_i \in T$
    u_{tgt} = sample in neighborhood of p_s
  end if
  $(x_{tgt}, q_{tgt}, t_{tgt})$ = local planner from (x_s, q_s) towards u_{tgt} with max. duration t_{max}
  if $t_{min} \leq t_{tgt} \leq t_{max}$ **then**
    add new tree node $(x_{tgt}, q_{tgt}, u_{tgt}, t, w_{new})$ to T, branching from p_s
    update tree weights $w_i \in T$
  end if
end while

function by pulling the tree extension towards a low-cost area if the extension fails because of too high costs. This way the tree growth is more greedily focusing on exploitation. If the solution needs to traverse large high cost areas however, exploration can be substantially slowed down.

The next section presents a modified version of the sampling-based random-tree planning approach developed in chapter 6, incorporating relaxed control local planning. Unlike the previous exploitation-heavy methods, the focus of the global task space search lies on exploration, the rapid and coarse scale discovery of global connectivity. This is achieved by a suppression of over-refinement, similar to the described techniques of guiding the search towards an uniform coverage, by including a new node weighting and target sampling scheme, biasing exploration towards less covered regions.

Although focused on exploration, local planning also performs exploitation by modifying trajectories according to the local costs, as long as not conflicting with the goal of reaching the target. By distinguishing trajectories that are better in line with the local costs, it is possible to bias the search towards low cost regions. An approach to measure and incorporate local costs into node weighting is developed.

8.3 Hybrid Planning with Relaxed Motion Control

In chapter 6.2, a sampling-based algorithm implementing the hybrid planning framework was developed (TaskTree). The relaxed motion control approach of chapter 3 can be immediately included by replacing the control method used in the local planning component. Algorithm 8.1 again shows the sampling-based hybrid planning algorithm. In order to incorporate relaxed motion control, the local planner is adapted, algorithm 8.2. Note that the inputs remain the same: The current state given by task and configuration positions (x_s, q_s), the task target u_{tgt} and the maximum control iteration time t_{max}. The outputs are the reached state (x_{tgt}, q_{tgt}) with control time t_{tgt} and in addition the cost reduction δc, indicating the amount of cost reduction done during local planning. As described in chapter 3, task velocities are mixed according to the weighting coefficients α and β, equation 3.1, with β being automatically adapted to ensure target convergence, equation 3.4. The

Algorithm 8.2 Relaxed Motion Control Local Planner

local planner from $(x, q) \in (\mathcal{X}, \mathcal{C})$ towards $u \in \mathcal{X}$ with max. control duration time t_{max}
while not max. duration time t_{max} exceeded **and not** constraints violated **do**
    compute cost-function $H(q)$ and gradient $\nabla H(q)$
    get target velocity $\dot{x}_{tg}$ and cost minimization velocity $\dot{x}_{cf}$
    mixing of task velocities according to α and β respectively β'
    compute joint velocities $\dot{q}$ given configuration q and mixed task velocities $\dot{x}$
    update current task position x and joint position q
    check constraint violation of updated q
end while
estimate trajectory cost reduction δc

Algorithm 8.3 Dispersion Reduction Target Sampling

sample in neighborhood of p_s :
for $k = 1...K$ **do**
    x_k = sample from $\mathcal{X}$ in neighborhood of node p_s
    d_k = nearest node from neighbors n of node p_s with distance $d_k = \min_n \rho(x_k, n)$
    L_S = insert pair (x_k, d_k) into candidate samples list L_S
end for
SL_S = sort list L_S by decreasing distances d_k
u_{tgt} = first target sample from sorted list SL_S

baseline proportion between the task velocities is fixed here with constant coefficients.

The other components, namely goal biasing, tree node selection, target sampling and node weighting do not necessarily need to be changed for relaxed motion control. However as already mentioned, relaxed motion control offers new possibilities needing different search and sampling strategies in order to utilize them. The following sections introduce novel heuristics replacing the node selection, target sampling and node weighing components.

8.3.1 Exploration Heuristic

With the sampling-based incremental tree growing approach of algorithm 8.1, nodes can be weighted according to different heuristics. One commonly used heuristic in sampling-based planning is a bias towards less explored regions of the space. The EST (Hsu et al., 2002) for example estimates how well the space is covered around a node, with the intention of biasing growth towards nodes in less covered regions. This heuristic is also used in the implementation of OOPSMP, see chapter 6, where the weight of a node is set inversely proportional to the number of outgoing edges of the node. If the spatial distribution and length of edges can be assumed to be uniform on average, the number of outgoing edges is a reasonable local coverage estimate. Although this assumption does only hold to a certain extend in actual motion planning, where some edges are blocked or shortened by constraints, some global exploration is always maintained.

Once this edge uniformity degrades to a larger heterogeneity in spatial edge distribution and edge length, the coverage estimate and the bias towards less explored regions degrades as well. With the increased local motion optimization ability of relaxed control, especially for better obstacle avoidance, such a larger heterogeneity becomes more likely. The local planner is more successful in creating motions because trajectories are forced away from obstacles, resulting in edges with a longer duration, covering larger distances. In other

situations local planning fails, either due to limitations of local optimization or due to constraints. The overall distribution of edge lengths can thus have a larger variance. In combination with the outgoing edge heuristic, exploration would be biased towards regions with short edges and dense node distributions, which are the regions where local planning did more frequent fail. Regions with successful local planning and longer edges, where nodes are less dense, would receive less attention on the contrary.

To counteract the over-proportional exploration in regions with a already dense node distribution, a different coverage estimation method is developed, useful for both node selection and target sampling. It is based on the idea of dispersion, as described in section 5.3. Dispersion measures the sampling density of a set of points, intuitively given by the radius of the largest possible ball that can be fitted into the free space between points. If the dispersion of a motion planning tree with respect to the state space is computed, one can identify the region of the space which is least covered by the tree. In section 5.3, this was used for ensuring rapid space exploration by greedily focusing the tree to achieve the largest possible dispersion reducing effect in every iteration. The Simultaneous Task and Configuration Space Dispersion Reducing Tree (algorithm 5.2) computes the dispersion in both the task and the configuration space and uses this information to identify the tree node with the largest configuration space dispersion and grows it towards the task target with the largest task space dispersion reducing effect.

This task space dispersion reduction method is now transfered into the sampling-based algorithm used here by replacing the target sampling component with a dispersion reduction target sampling, algorithm 8.3. Instead of just sampling one random target point in the neighborhood of the selected node, K points x_k are sampled and for each x_k, the nearest tree neighbor node with distance d_k is determined. This nearest neighbor search is limited to the local region of the node, including only the node itself and all directly connected nodes. The sample with the largest distance represents the approximated local dispersion around the node and growing the tree towards this sample has the largest dispersion reduction impact. Figure 8.2 illustrates this process.

With the new target sampling, local exploration now takes edge length and distribution into account and tries to reach an uniform local coverage. In order to equalize the global coverage as well, the node weighting scheme is adapted to incorporate the dispersion estimate captured during target sampling. Nodes with a lower remaining dispersion are assumed to lie in better covered regions than nodes with higher remaining dispersion and putting a bias towards this high dispersion nodes steers the exploration towards uncovered regions. Figure 8.3 shows how the weights are computed. The initial weight of a newly created node x'_i is set in relation to its distance d'_1. This does not reflect the new nodes local dispersion, but could be interpreted as the contribution in dispersion reduction of the new node. The weight of the extended node is set to the second largest distance d_2, corresponding to an estimate of the remaining local dispersion after the new node is added to the tree. Thus a new node gets a larger weight if the local planner explored further and the weight of an old node gets lowered, if the density of nodes in the neighborhood increases. The list of distance sorted samples SL_S is kept in algorithm 8.3 for being able to access the second largest distance d_2 later during node update.

The Simultaneous Task and Configuration Space Dispersion Reducing Tree estimates the configuration space dispersion as well to identify less covered configuration space regions. Although it is possible to incorporate this additional information into the node weighting scheme, the expected impact is negligible here. With a greater local planning autonomy, the main focus lies on most profitable task space sampling and exploitation.

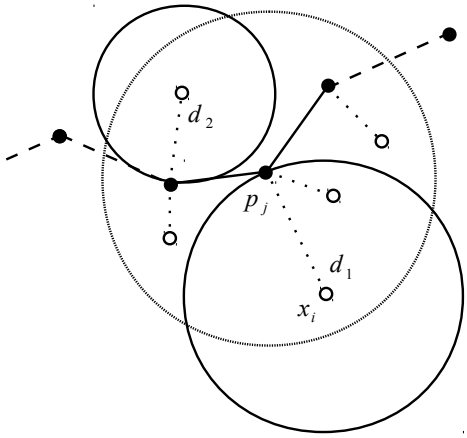

Figure 8.2: Local dispersion estimation in the neighborhood of node p_j as the largest nearest neighbor distance $d_1 = \max_{x_k} \min_{p_n} \rho(x_k, p_n)$ of samples x_k towards direct neighbor nodes p_n of node p_j.

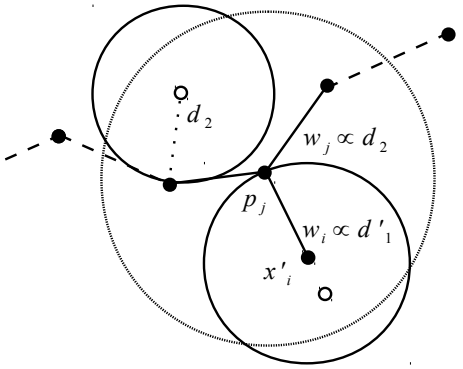

Figure 8.3: After local planning, a resulting node at x'_i receives an exploration weight w_i equal to the distance d'_1. The exploration weight w_j of the extended node p_j is set to the second largest dispersion estimate d_2.

8.3.2 Exploitation Heuristic

The exploration heuristic from the previous section is targeting a rapid and uniform coverage of the task space. In this section, another heuristic is developed, aiming to achieve a better minimization of local costs. Local planning with the control methods of chapters 2 and 3 includes optimization of additional cost functions, encoding secondary motion objectives. These objectives are minor objectives, subordinate to the objective of reaching the target position. The amount of optimization possible is dependent on the actual sampled target position. Some targets can be reached on trajectories with decreasing costs, while others can not. Figure 8.4 shows some example trajectories and the evolution of the obstacle avoidance costs. It can be seen that depending on the target position, the cost difference between the start and end positions varies. While the costs are steadily decreasing for trajectory A, costs are increasing towards the end for trajectory B. Looking at the absolute difference, costs are only slightly less for trajectory B at the end, while the cost decrease is more prominent for trajectory A. It is even possible that costs are increasing beyond the initial value.

Since the amount of optimization possible depends on the ability of the local planner to utilize the local cost structure, the term exploitation is used here. Good exploitation refers to a trajectory with successful cost minimization. Bad exploitation depicts a trajectory forced to move towards increasing costs and thus ignoring local structure. By focussing the global search towards trajectories with better success in cost minimization, a bias towards regions with good exploitation abilities is introduced. This has two benefits. First, focusing the search in regions with larger exploitation means focusing on regions where local planning was successful and is also more likely to continue to be successful, if some local uniformity of the cost structure is assumed. Global sampling can be potentially faster, if there are solutions possible that stay in low cost regions. Secondly, the overall costs of a solution trajectory are more likely to stay low, since the growth is biased to favor low cost nodes.

There are several possibilities to estimate cost decrease and thus exploitation. The one adapted here in the local planner component (algorithm 8.2) is very simple. It is computed by just taking the difference between the first and the last cost measure, ignoring the

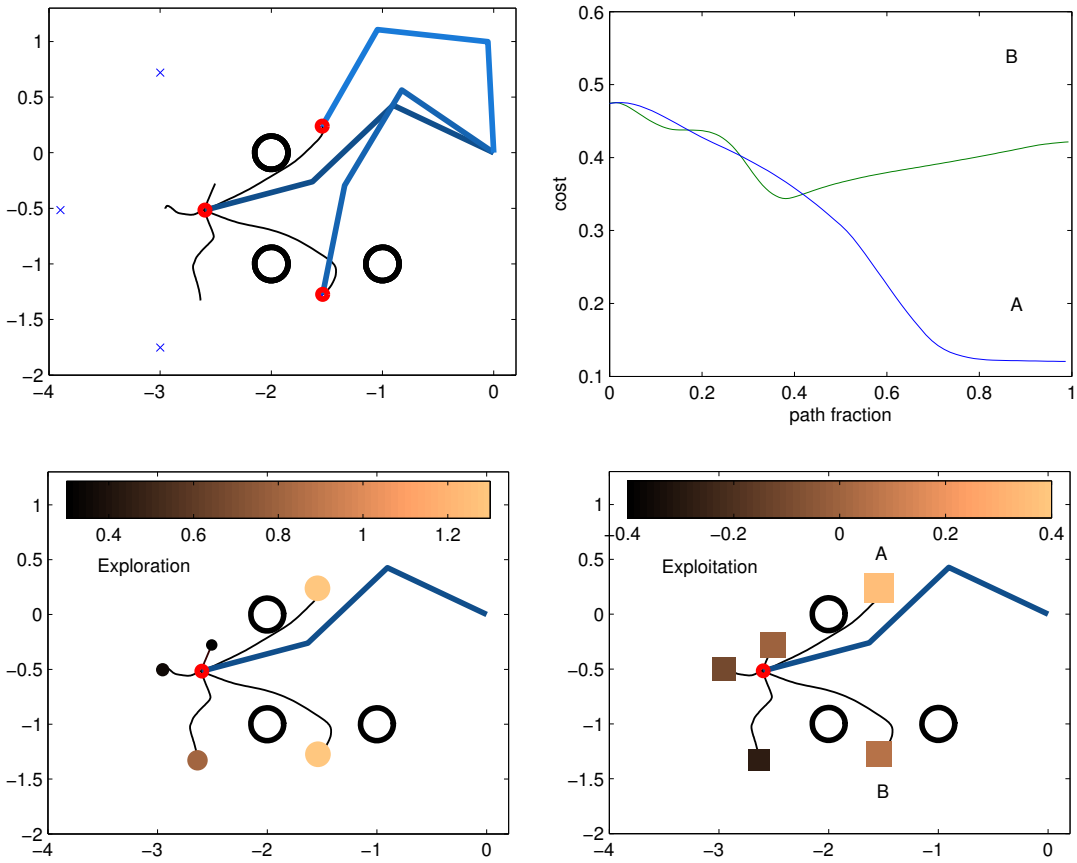

Figure 8.4: A simple three joint manipulator performing movements in the plane (top left). Starting at the central position, five targets marked by the crosses are approached with relaxed motion control, optimizing obstacle avoidance. The exploration weight of the resulting tree nodes is shown (bottom left). To estimate the success of exploitation, the absolute cost function decrease (top right) is used to set corresponding exploitation weights (bottom right). Since the cost decreases more towards node A, its exploitation weight is larger than the exploitation weight of node B.

behavior between these points. It can be computed very fast but is of course only a very rough estimate. A way to include the whole behavior for example would be to sum up all portions with monotonically increasing costs to get a notion of work necessary to follow the path (Jaillet et al., 2010). How this exploitation estimation is included into the node weighting scheme is subject of the next section.

8.3.3 Node Weighting

The exploration and exploitation heuristics are both included into the node selection of the hybrid sampling-based planning algorithm. The probability of a node to be selected is proportional to its overall weight and by construction of a suitable weighting function, both heuristics can be considered.

In principle, the desired node weighting scheme could be outlined as follows. If a new tree branch covers a longer distance, it was able to explore the space to a greater extend and should receive a high exploration weight. Conversely, shorter distances receive a lower exploration weight. Also, if a node is selected more than once, its exploration weight should lower proportionally to the level of further exploration in its neighborhood. This is captured by setting the weight according to the local dispersion estimate after a node was extended.

Algorithm 8.4 Weight Initialization

initial node weight, parent node s and local planning cost estimate c :
exploration weight of new node $w_{ex,new} = \rho(x_{tgt}, x_s)$
exploitation weight $w_{pl,new} = 1/(1 + \exp(-b\ \delta c))$

Algorithm 8.5 Weight Update

update tree weight of selected node p_s :
exploration weight of selected node $w_{ex,s}$ = second largest distance d_2 from SL_S

To achieve this behavior, the exploration weight is set to the covered distance for new nodes and to the local dispersion estimate else in a linear fashion, as shown in algorithm 8.4 and 8.5.

The exploitation weight is set relative to the cost decrease estimate. It is desired to have a high weight if the cost is decreasing and a low weight if the cost is increasing. In order to emphasize the difference between decreasing and increasing costs even stronger, a non-linear relationship is used to compute exploitation weights, as shown in figure 8.5. Slight cost changes around zero receive proportionally more attention than cost variations on either high or low level. Once computed for every new node, exploitation weights are fixed. Algorithm 8.4 shows the equation, with b being proportional to the slope of the function at zero cost change.

Finally, exploration and exploitation weight are multiplied to form the overall node weight, algorithm 8.6. Figure 8.6 shows the combined weight in relation to exploration and exploitation. Node weight is linearly increasing with larger exploration distance, but the slope is dependent on the amount of exploitation cost decrease. With a positive cost decrease, the weight is raising faster for better exploration and slower for less exploration. Overall, this weighting function discriminates between exploration with high exploitation and with low exploitation and puts a bias on the former.

8.4 Evaluation

To test the impact of relaxed motion control and the proposed exploration and exploitation heuristics, the performance in a simulated motion planning task is evaluated. Three variants of the global planning scheme are compared. The approach from chapter 6 using the EST edge count coverage estimate, the exploration heuristic explained in section 8.3.1 and the combined exploration and exploitation heuristic as described in section 8.3.3. In addition, local planning with the new relaxed local planner and with the previously used local planner with redundant space optimization is compared.

Algorithm 8.6 Node Selection

select tree node from $p_i \in T$:
p_s = choose randomly according to weight distribution $\{\ w_{ex,i} \cdot w_{pl,i} \mid i = 1...N\ \}$

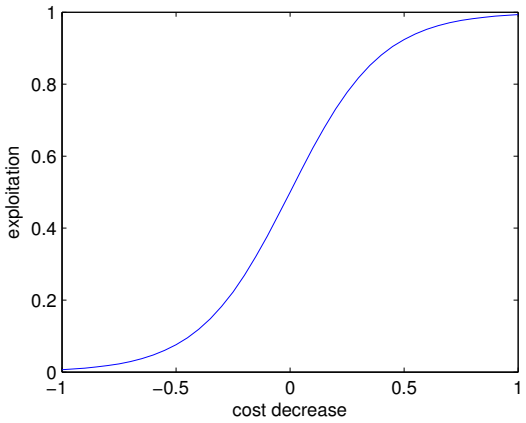

Figure 8.5: Non-linear relationship between cost decrease and exploitation weight ($b = 5$).

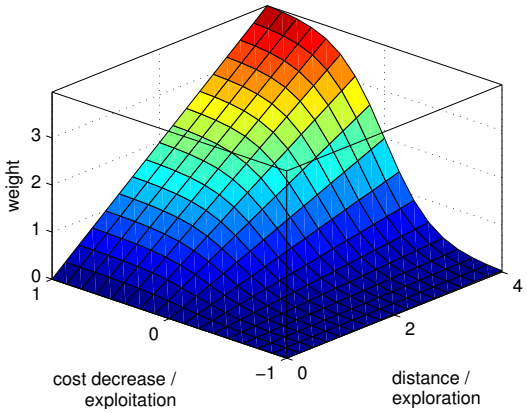

Figure 8.6: Overall node weight in dependence of exploration and exploitation.

8.4.1 Algorithm Implementation

The simulation environment used for evaluation of the approach presented in this chapter is completely implemented in Matlab[1]. The three main components are the sampling-based global search (algorithm 8.1), local planning incorporating the underlying control framework (algorithm 8.2), and the simulation of the robot arm, including kinematics, collisions and obstacle distance computations.

In the sampling-based global search component, distance computations are needed for goal-biasing and dispersion reduction target sampling. These distances are defined in the task space simply as the euclidean distance $\rho(x, y) = ||x - y||,\ x, y \in \mathbb{R}^2$. While distance computations for large point sets are a major concern for sampling-based planning, see (LaValle and Kuffner, 2000) for example, these problems do not arise here. Nearest tree neighbors of a candidate sample set are computed during target sampling, but since the scope is limited to a small local sub-tree and candidate set ($K = 5$), costs remain bounded. For goal-biasing, the goal distance of each node is calculated after local planning and stored in the node. The nearest neighbor to the node is then found in linear time by searching the current minimal goal distance.

An expensive part of the current implementation is local planning, with the iteration of the control equations and evaluation of the potential functions. If the obstacle avoidance is used, computation of the potential can quickly become the most expensive part of the approach, depending on the number of obstacles involved. In the present simulation environment, the robot workspace is planar with the robot segments modeled as lines and obstacles modeled as circles, refer to figure 8.7. Although the individual calculation of distances and closest points between planar lines and circles can be implemented very efficiently, the number of these calculations increases quadratically with the number of robot segments and obstacles. Also, closest-point Jacobians have to be assembled for every obstacle inside the reactive range, another costly operation during the potential function computation[2].

Other simulation components, like forward kinematics of the robot arm and collision detection are less critical. The collision detection in particular can be very efficient if the results of the distance computations are reused.

[1] http://www.mathworks.com/products/matlab

[2] For Setup B, the iteration of the control equations needs about 95% of the overall computation time, with Jacobian assembly (about 40%) and distance computation (about 30%) using the largest share.

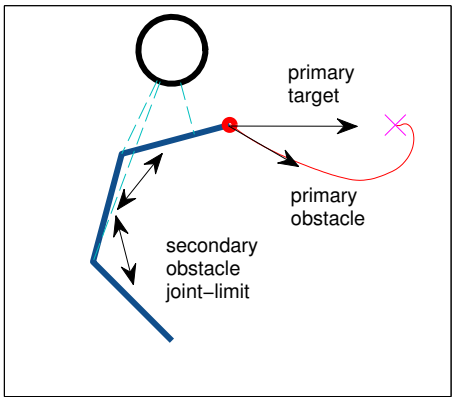

Figure 8.7: Local planning with relaxed motion control incorporates two hierarchical motions: Primary task-space motion and secondary redundant-space motion. The primary task motion consists of a target directed and an obstacle avoidance component. The secondary redundant motion performs joint-limit avoidance and obstacle avoidance.

8.4.2 Local Planning

Local planning builds on the relaxed motion control approach from chapter 3. Besides target reaching, two secondary objectives are incorporated, joint-limit avoidance and obstacle avoidance. Section 2.4 describes potential functions encoding these two objectives. Secondary motions influence local planning in two ways. First, the primary target directed motion is mixed with obstacle avoidance, resulting in task trajectories able to circumvent obstacles but still converging to the target. Second, both joint-limit and obstacle avoidance movements are projected into the redundant space, locally optimizing the robots posture. Figure 8.7 schematically shows the interplay of the different objectives.

By setting the β parameter in the relaxed control equation, it is possible to switch between full task space local planning with relaxed motion optimization and local planning with redundant space optimization. An example comparing these two local planning approaches was previously given in section 3.2, illustrating the ability of relaxed control to avoid obstacles by modifying the task trajectory and to produce movements with lower obstacle costs.

8.4.3 Setup

Several motion planning tasks are defined in two different environments, both featuring a planar multi-link robot in a planar workspace, populated by circular obstacles. The two setups (A and B) are shown in figure 8.8 and 8.9, respectively. Setup A has a regular structure with six goal positions all located at the end of different corridors. Figure 8.10 illustrates that indeed a global planning scope is necessary, because local optimization alone does not succeed. Also, the goal positions at the right side (4,5 and 6) can not be directly reached due to joint constraints, as shown in figure 8.11. The joint range is limited to stay in the interval from $-\pi$ to π, i.e. the manipulator can not fold upon itself. In addition obstacle collision constraints are active, but note that self-collisions of the manipulator are not prohibited. The spatial structure in setup B is less regular. Some corridor like structures exist, but to a lesser extent. Also, some goal positions can be reached on different paths, not homeomorphic to each other. The task space in both setups is the two dimensional position space, while the configuration space in setup A is of dimension 4 and the configuration space in setup B of dimension 8.

8.4.4 Performance Metric

There are two areas of interest for comparing the simulation results. One is the difference between the local planning methods, between redundant space motion optimization and

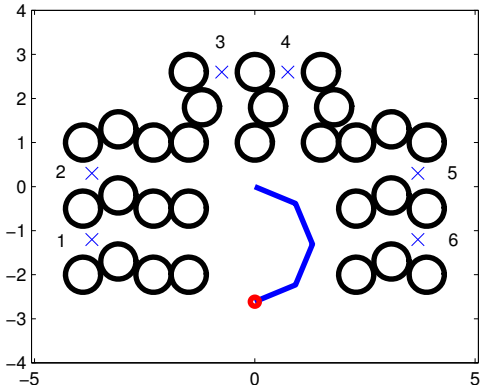

Figure 8.8: Setup A: Planar robot arm with 4 joints, fixed at the origin, the red circle represents the end-effector. 6 different task goals are defined in the 2D position task space.

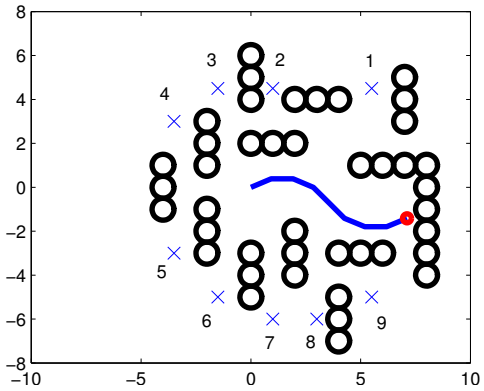

Figure 8.9: Setup B: Planar robot arm with 8 joints, fixed at the origin, the red circle represents the end-effector. 9 different task goals are defined in the 2D position task space.

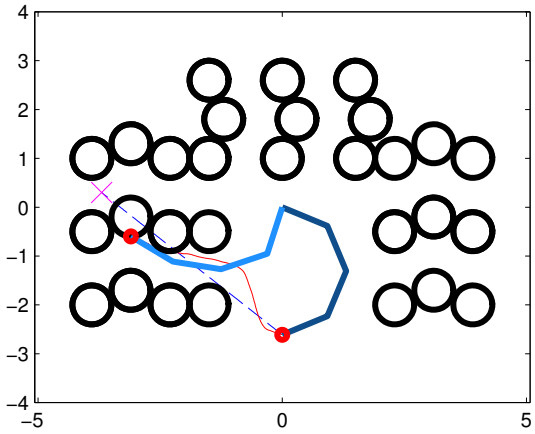

Figure 8.10: Setup A, approaching target 2 with relaxed obstacle avoidance control. The limited scope of local optimization leads the motion into a dead-end situation.

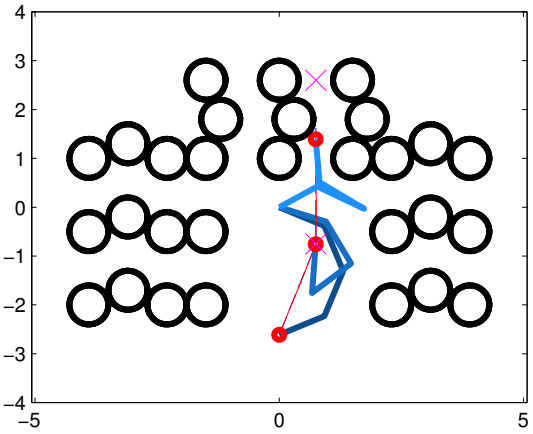

Figure 8.11: Setup A, approaching target 2 directly. Again caused by the limited scope of local optimization, the arm moves towards a joint-limit, preventing further progress.

relaxed optimization incorporating the task space. The second area is the impact of the three global sampling weighting heuristics, EST coverage estimation, exploration weighting and combined exploration and exploitation weighting. In order to compare these methods, a suitable performance metric has to be used that measures the effort of planning.

Previously in chapter 6 and 7, the overall runtime was used as such a measure of effort. The overall runtime is determined by two factors, the number of control cycles needed to create a local plan and the number of local plans generated until the goal is found. However, the runtime of local planning with relaxed motion control is in general higher. A higher number of control cycles are needed for planning a trajectory with the same start and target position, because the trajectory can be longer and is generated with smaller steps, especially in situations with conflicting task objectives, where the resulting target velocity is low. However, the actual runtime difference of local planning is merely an implementation issue and could be counteracted by controlling the control step sizes and equalizing the number of iterations needed.

Assuming that the runtime of local planning under the two control schemes can be equalized, the overall time is then governed by the number of calls of the local planner. Consequently, the number of local planning calls, or correspondingly the number of tree growing attempts, is used here as the performance measure. It enables to measure the planning effort independently of local planning costs, making it easy to compare the impact of the different tree weighting heuristics.

The maximum number of iterations is bounded to 500 in the present setup. If planning runs do not succeed with less than 500 iterations, they are terminated and reported as not solved. For the performance evaluation, it is thus necessary to look at the fraction of solved planning queries out of the total number of queries. Unsuccessful runs account to the overall iteration number statistics with a constant value of 500, introducing a distortion of the average towards a value lower than the true iteration average, if every run would be iterated until a solution is found.

Local planning with relaxed control is able to locally lower the obstacle avoidance cost function to a greater extend than with strict task constraints, as previously shown in section 3.2. An interesting question for evaluation is how the obstacle avoidance costs of solution paths behave on the global level. In particular, the influence of the exploitation node selection heuristic has to be analyzed. To do so, the individual contribution of path segments in decreasing the overall path cost is estimated the same way as during estimation of the exploitation weight, as explained in section 8.3.2. All individual path segment contributions are then averaged over the whole path to form a solution cost optimization performance estimate. This way, the cost performance measure uses the same information as used during planning, sharing the same inaccuracy regarding the actual cost evolution between local planning end-points and the length of path segments. However, this inaccurate estimate is sufficient to assess the differences between the different node weighting heuristics.

8.5 Results

For every goal position of both setups and for every combination of local planning (relaxed optimization control with $\beta = 1$ and redundant optimization control with $\beta = 0$) with node weighting (EST, exploration weight and combined exploration/exploitation weight) 50 motion planning runs were simulated. Example solutions are shown in figure 8.12 and figure 8.13. In both cases global planning used combined exploration/exploitation weighting, while local planning was done with redundant optimization control in figure 8.12 and with relaxed control in figure 8.13. Comparing both local planning variants, it can be seen how

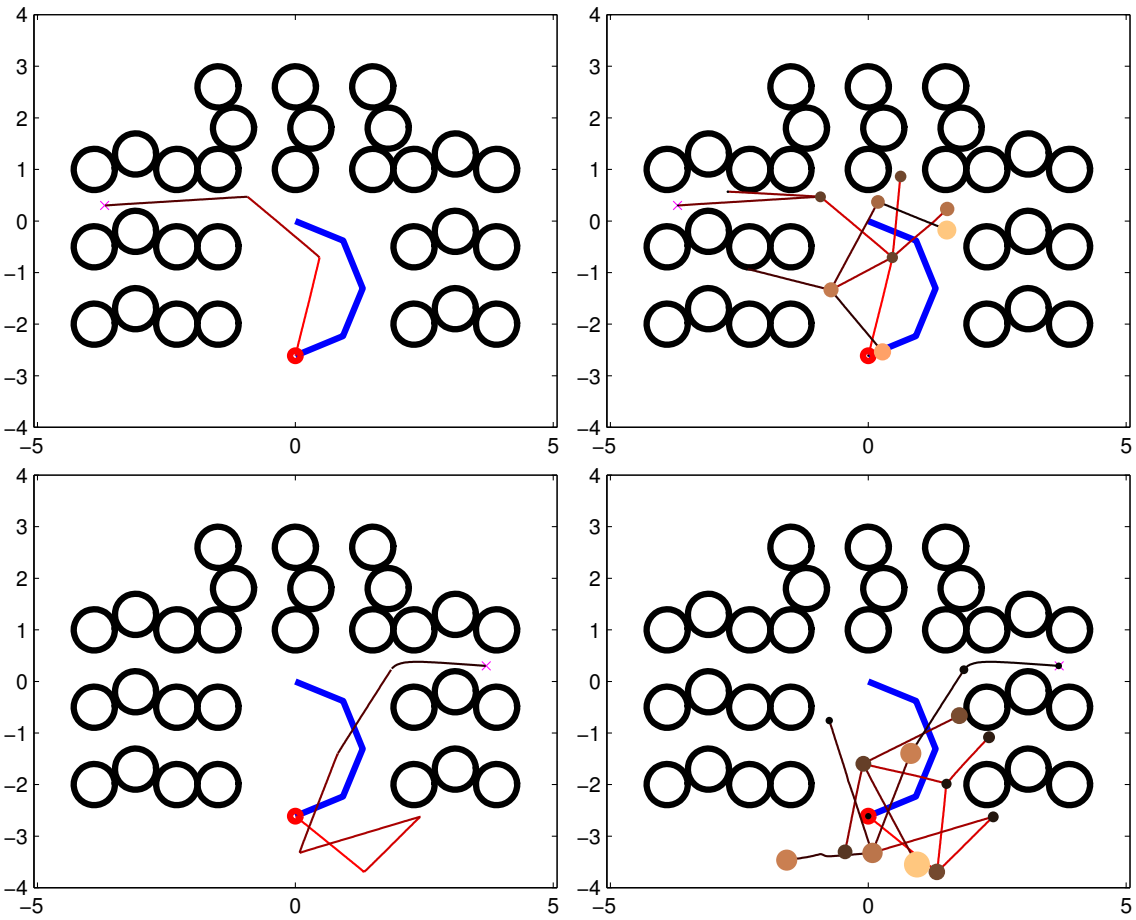

Figure 8.12: Setup A: Example solutions and corresponding search trees for two different goal positions, marked by the crosses. Local obstacle and joint-limit avoidance is limited to the redundant space ($\beta = 0$). Global planning uses the combined exploration/exploitation weighting scheme. Nodes with higher weights are shown with larger and brighter circles.

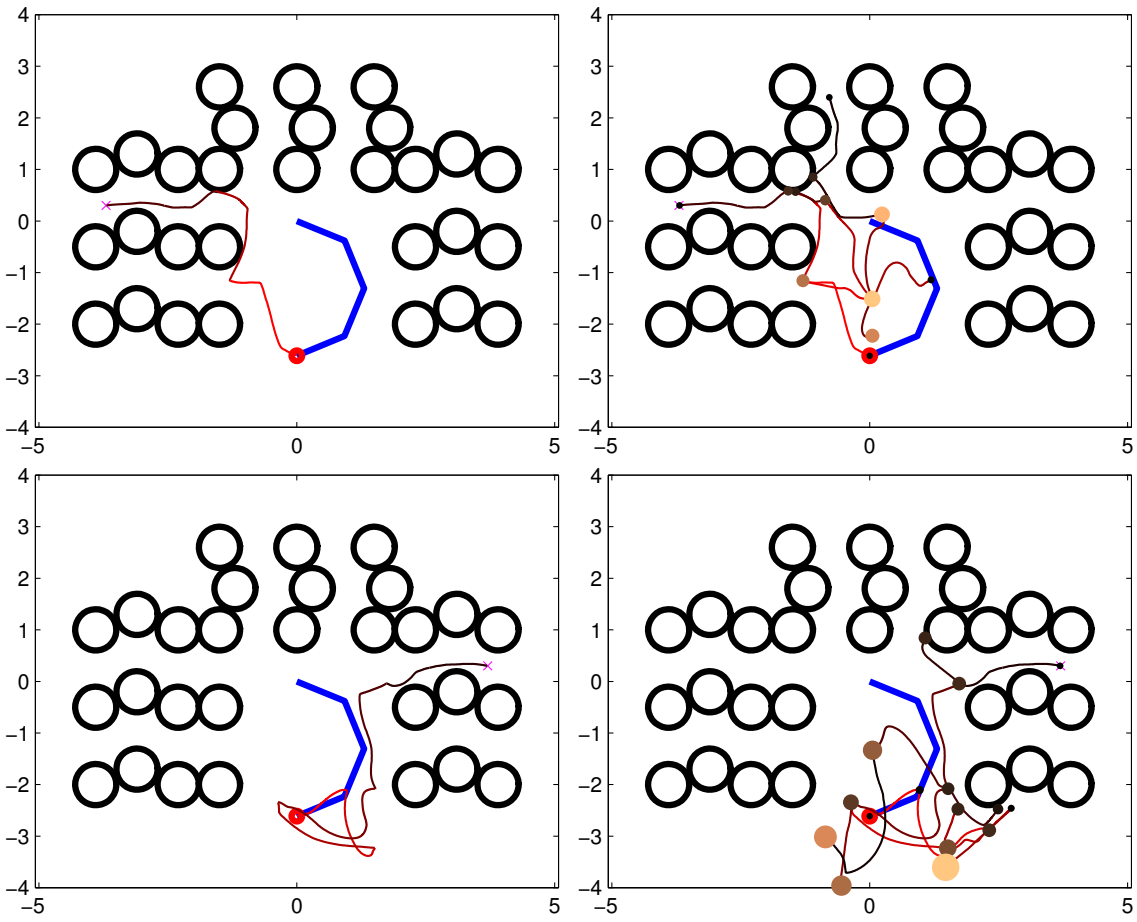

Figure 8.13: Setup A: Example solutions and corresponding search trees for two different goal positions, marked by the crosses. Local planning with relaxed control, optimizing obstacle avoidance in the task space and combined obstacle and joint-limit avoidance in the redundant space ($\beta = 1$). Global planning uses the combined exploration/exploitation weighting scheme. Nodes with higher weights are shown with larger and brighter circles.

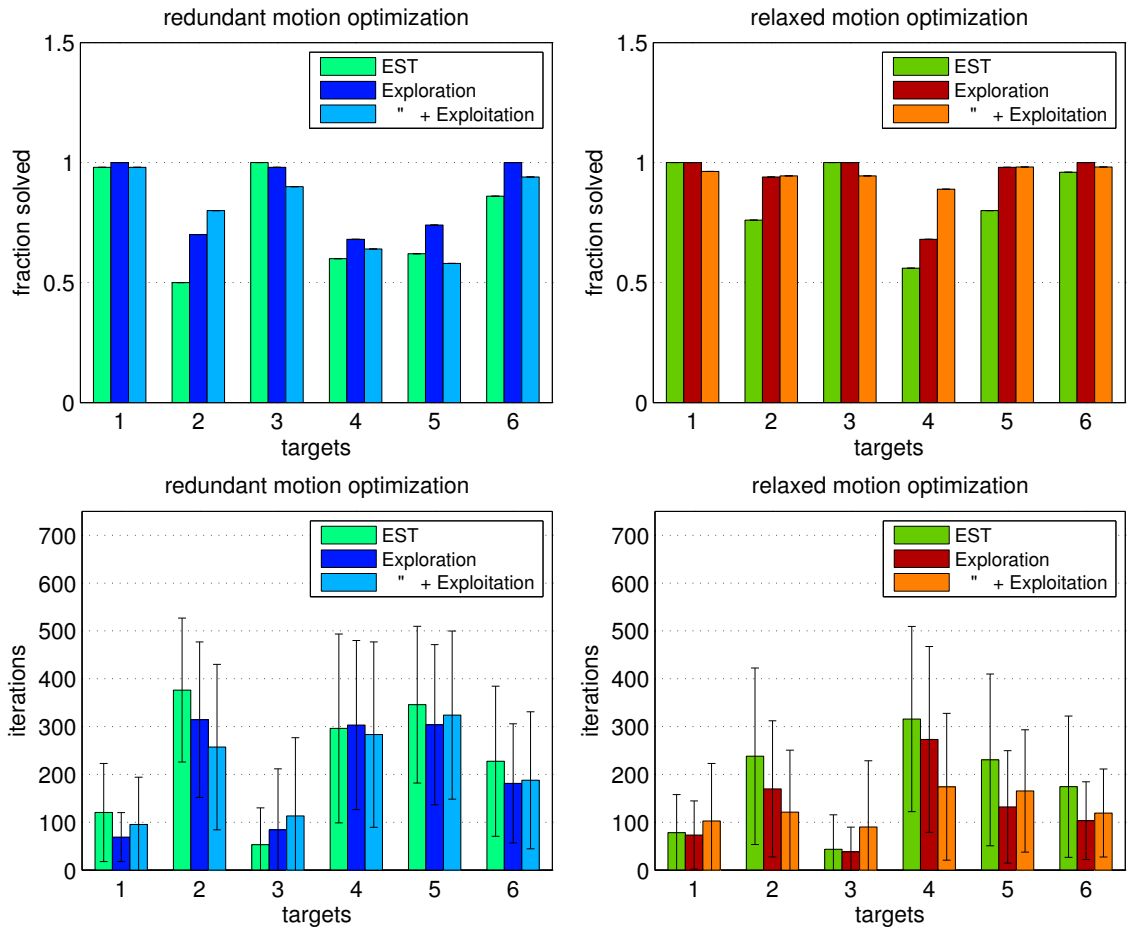

Figure 8.14: Setup A: Fraction of solved planning queries (top row) and average number of iterations (bottom row), 50 runs per target. Redundant motion optimization ($\beta = 0$) on the left and relaxed optimization control ($\beta = 1$) on the right. For each target, results for EST weighting, exploration weighting and combined exploration/exploitation weighting are shown.

the relaxed control trajectories are deformed to locally avoid obstacles.

The results for setup A and B are shown in figures 8.14 and 8.15, respectively. They are divided according to the two different local planning methods and the three node weighting heuristics and grouped together for every target. The fraction of solved planning queries out of 50 trails is depicted in the top row.

Depending on the actual target, the individual performance varies in a nonuniform way, with some more and some less distinctive differences. Starting with the less distinctive, one can identify some target positions which are constantly and reliably solved with a low number of iterations, regardless of the used control method and weighting heuristic. Targets A1, A3 and B1 (setup A, targets 1 and 3, setup B target 1) are solved with approximately 100 iterations or less on average. Targets B7 and B8 took around 200 iterations, but do not show any pronounced variations across the different methods either.

More interesting are the cases that show clearly distinctive differences. Such a observation is that for every target, excluding the previously identified low distinctive ones, the lowest number of iterations is always achieved with relaxed control. How much lower the number of iterations is, varies between the targets as well as between the three weighting heuristics. For targets A4, A6, B3, B5, B6 and B9, EST weighting does not reduce the number of iterations while for targets A2, A5, B2 and B4 EST weighting is able to reduce it. The

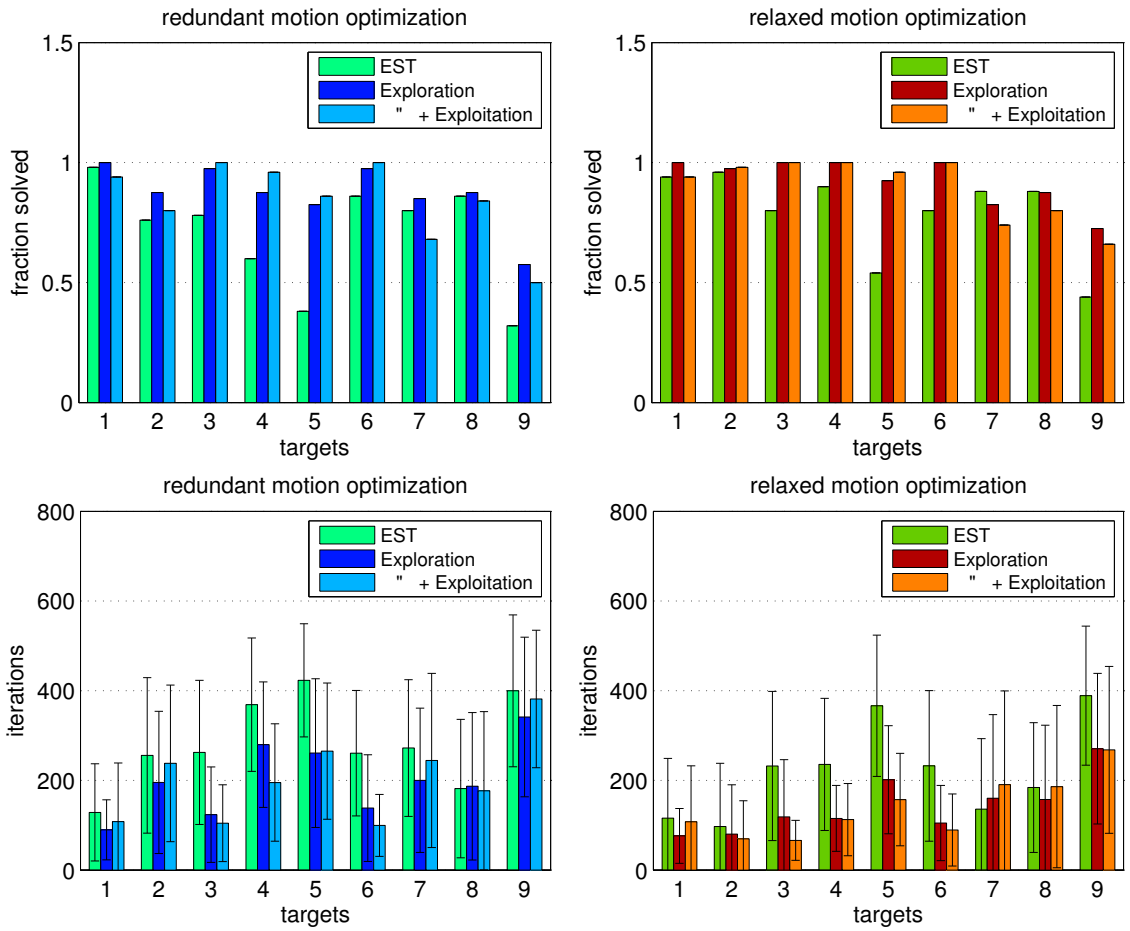

Figure 8.15: Setup B: Fraction of solved planning queries (top row) and average number of iterations (bottom row), 50 runs per target. Redundant motion optimization ($\beta = 0$) on the left and relaxed optimization control ($\beta = 1$) on the right. For each target, results for EST weighting, exploration weighting and combined exploration/exploitation weighting are shown.

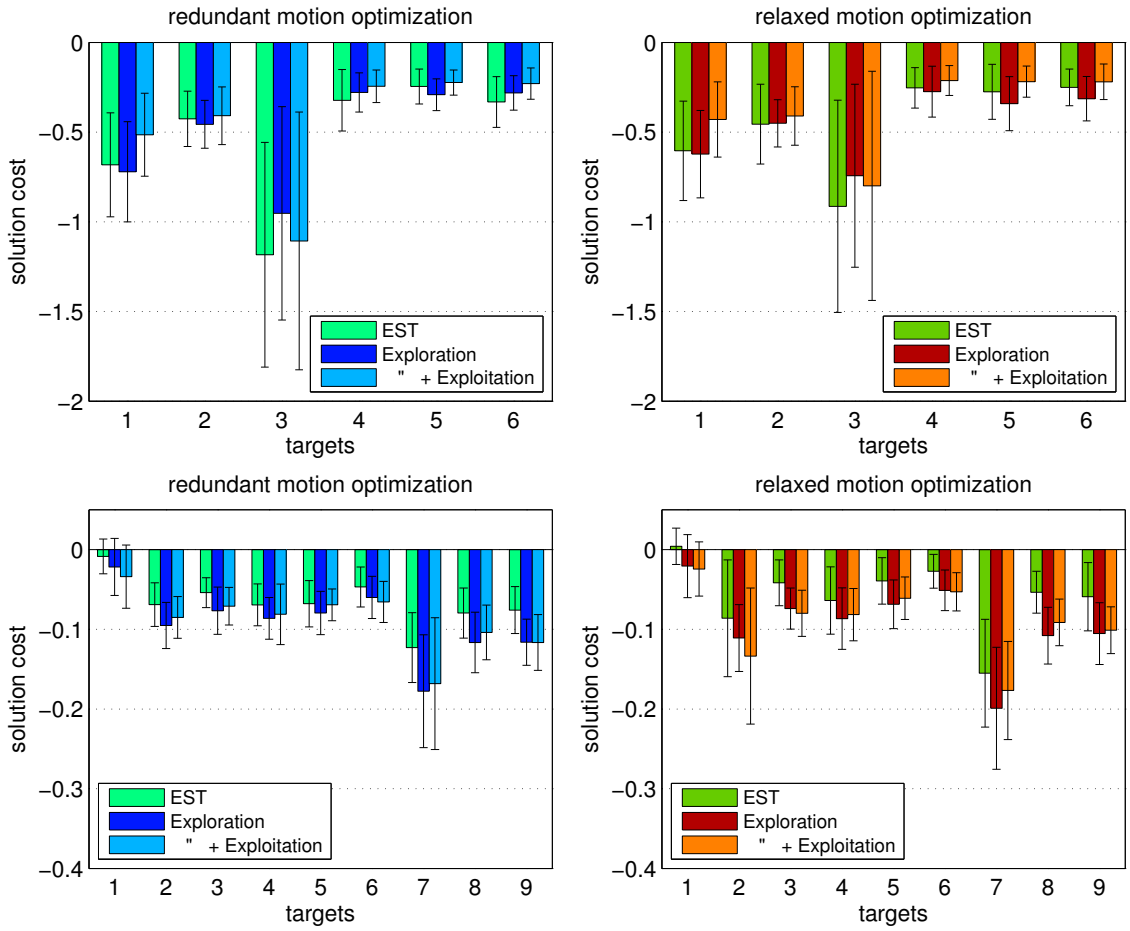

Figure 8.16: Average obstacle avoidance costs of solution trajectories for setup A (top row) and setup B (bottom row). Redundant motion optimization ($\beta = 0$) on the left and relaxed optimization control ($\beta = 1$) on the right. For each target, results for EST weighting, exploration weighting and combined exploration/exploitation weighting are shown. Note that larger values are better, representing a stronger decrease of obstacle costs.

exploration and the combined exploration/exploitation weight on the other hand reduce the number of iterations when relaxed control is used, for almost all targets except B3 and B6, where no differences between both control methods are visible. Compared to the EST heuristic, this two heuristics always need less iterations.

Lets have a closer look on the exploration heuristic. Like already mentioned, the number of iterations needed is lower as with the EST heuristic, clearly lower if relaxed control is used and also to a lesser extend for redundant motion optimization. For several cases, namely for targets A5, A6, B3, B4, B5, B6 for relaxed control and targets B3, B5 and B6 for redundant optimization control, the number of iterations drops by 50%.

Regarding the impact of the combined exploration/exploitation heuristic, there are some examples where the additional exploitation weight further decreases the iteration number. For relaxed control, these are A2, A4, B3, B5 and B6 and for redundant optimization A2, B4 and B6. Conversely, for the remaining targets the iteration number remains the same or is even increasing.

Figure 8.16 shows the obstacle avoidance costs averaged over all solution trajectories. Comparing both local planning methods, it can be seen that for all target positions the average solution costs for relaxed motion optimization are not substantially better. Regarding

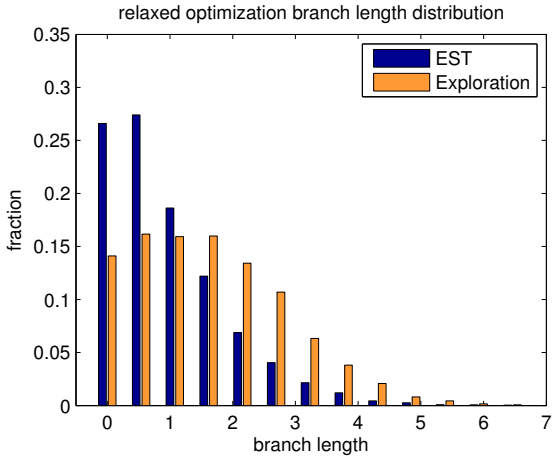

Figure 8.17: Distribution of tree branch lengths for EST and exploration node weighting. Longer branches correspond to a more successful local planning and a faster exploration of the space.

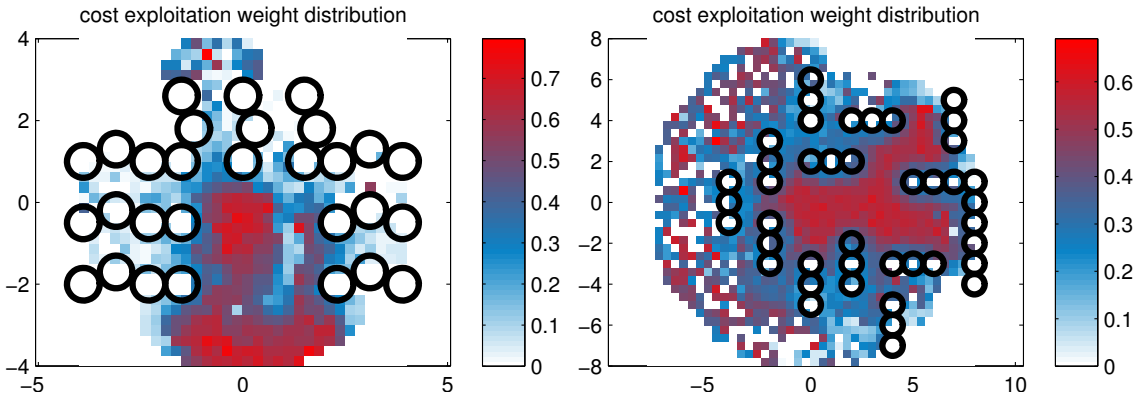

Figure 8.18: Spatial distribution of exploitation node weights w_{pl}, for setup A (left) and setup B (right). Weights are averaged over all nodes in a cell.

the three different node weighting schemes, no consistent relation is visible. In setup A, the single slight difference can be seen for target A, where the combined exploration/exploitation heuristic performs the best. Conversely in setup B, the EST weighting scheme behaves better than the other two heuristics.

8.6 Discussion

The results for both setups show the advantage of using the relaxed motion control scheme as a local planner in the hybrid motion planning framework. For most target positions, relaxed motion control reduces the number of global planning iterations needed to find a solution. The enhanced capability to avoid obstacles by modifying the task space trajectory increases the probability of the local planner to find a valid trajectory, reaching the sampled task targets more often and exploring the space faster. In regions with an easy to exploit local structure, the effect is stronger than in regions with less easy structure and accordingly, the ability to reduce the iteration number varies across the different targets. However, there are no targets where relaxed motion control has a negative impact.

The global sampling scheme plays an important role too. Using the EST node weighting heuristic as in chapters 6 and 7 is possible, but in general a larger number of global planning iterations is needed. Using relaxed control reduces the number of iterations in some cases, but not to the same extend as with the other two weighting heuristics.

The exploration node weighting heuristic on the other hand is more successful in reducing the global sampling effort. Although this can be seen for both local planning approaches, the benefit of this weighting scheme is most obvious if relaxed motion control is used. The novel exploration weighting scheme utilizes the capabilities of relaxed control to a greater extend. Looking at the distribution of lengths of the tree branches in figure 8.17 shows a significant difference between the EST and exploration heuristics. A large number of relatively short branches are present in the EST tree, while the exploration tree has a wider distribution of branch lengths, with considerable less short branches and more longer branches. By focusing on nodes with a large exploration rate, the exploration heuristic hinders an over-exploration of regions where local planning is successful but not progressing very much, as occurring frequently with relaxed control. The EST heuristic however treats all regions the same, eagerly focusing exploration even if little progress is made.

With the combined exploration/exploitation node weighting scheme, results are ambivalent. In some cases a further speedup of the global search is possible, but in other cases not. The spatial distribution of the additional exploitation weights in figure 8.18 indicates why not all planning runs benefit. High weights can be found in the center and the boundaries of the workspace, where relatively large areas of free space are located. In setup B another region with high weights is on the top right, where the corridor between obstacles is wide. If corridors get narrower, weights are quickly decreasing. Nearly all of the goal positions also lie in regions where low weights dominate. Thus, biasing the search towards higher exploitation weights drives the search away from narrow corridors towards more open regions. While this behavior has a positive effect if large parts of a solution trajectory lie in the free space, it makes it more difficult to find solutions that have to get closer to obstacles.

However, the negative impact remains rather small in the current setups, although narrow corridors have to be traversed for most goal positions. One explanation why this is the case is given by the obstacle avoidance ability of relaxed motion control, helping to traverse corridors and making exhaustive sampling inside the corridors unnecessary. For different motion planning setups, it could be worthwhile to tune the relationship between exploration and exploitation weights. Figure 8.19 illustrates the evolution of node weights during global planning. Equal node weights are shown with isolines in the space spanned by the two components exploration, given by the dispersion estimate d, and exploitation, given by the cost decrease δc. Due to the non-linear mixture of both components, nodes with higher exploitation are over-proportionally favored, while low exploitation nodes receive less attention. By modifying the weighting function, it is possible to implement a wide range of behaviors, adapted to certain problem instances. For example, the slope b can be increased for an even stronger bias towards high exploitation. Conversely, the bias can be shifted towards lower exploitation by decreasing b, up to negative values of b, reversing the relationship by favoring low exploitation over high exploitation. For setups where solutions are known to primarily traverse high cost areas, this could be advantageous.

Regarding the second performance criteria of lower obstacle avoidance costs, a positive effect of relaxed motion control and exploitation weighting can not be shown. Local planning with relaxed motion control does not consistently lower the overall costs of the solution trajectories. Also, the EST weighting scheme tends to perform better, especially for setup B. Adding the exploitation weights to measure the achieved cost reduction does not result in finding low cost solutions.

To get an insight into this results, it is important to keep the precedence of all involved motion objectives in mind. The local planning modules used in the current setup primarily try to reach the sampled target positions, even if the secondary objective of low obstacle avoidance cost is counteracted. Only if both objectives are not conflicting, obstacle costs can be equally considered. This way a rapid exploration of the task space can be achieved,

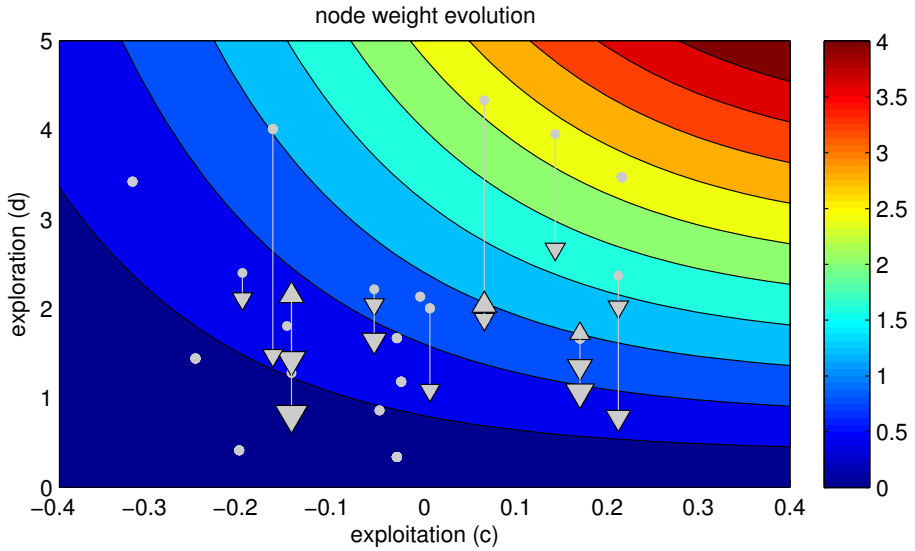

Figure 8.19: Relationship between exploration (y-axis), exploitation (x-axis) and node weights (color) and its evolution over time. Dots represent tree nodes and weight changes of nodes are indicated by arrows, either decreasing weight (down) or increasing weight (up). The arrowheads grow with successive selection of the corresponding node.

but reducing obstacle avoidance costs plays a subordinate role. Assuming an identical global sampling process, the overall cost of a solution trajectory can only be lower if sub-trajectories can be optimized without these conflicts. Consequently, randomized global sampling without any dependence on optimization success can only result in lower costs under fortunate preconditions or by random chance.

The idea of incorporating the achieved optimization into the node selection with the combined exploration/exploitation weighting is to actively bias the search towards solutions with lower costs. If better sub-trajectories are encountered that benefit from fortunate local conditions they are favored. To be able to choose these better trajectories, sufficiently many of them have to be created, capturing at least parts of the local cost variations. A dense sampling of alternative trajectories however contradicts the premise of a rapid exploration with few iterations. Looking at the example search trees in figure 8.13 indeed confirms the relative sparseness of samples. Comparing the node placement with the local cost structure, as indicated in figure 8.18, shows that it is likely that solutions are already found before cost minimas are encountered.

Figure 8.20 shows the spatial distribution of solution path nodes over all planning runs for one target. Comparing the EST and the exploration/exploitation heuristic one can see that in the latter case, solution trajectories cover a larger area of the space. The exploration bias results in longer paths, spreading out faster into the space until constraints are encountered. Consequently, solutions found with this approach tend to be longer too and to cover larger areas. Depending on the cost structure of the setup, also shown in figure 8.20, the wider distribution of solutions results in parts of the trajectories traversing regions with higher costs. Thus, the overall costs can indeed be larger if the exploration heuristic is used, explaining the finding that EST tends to produce solutions with lower costs.

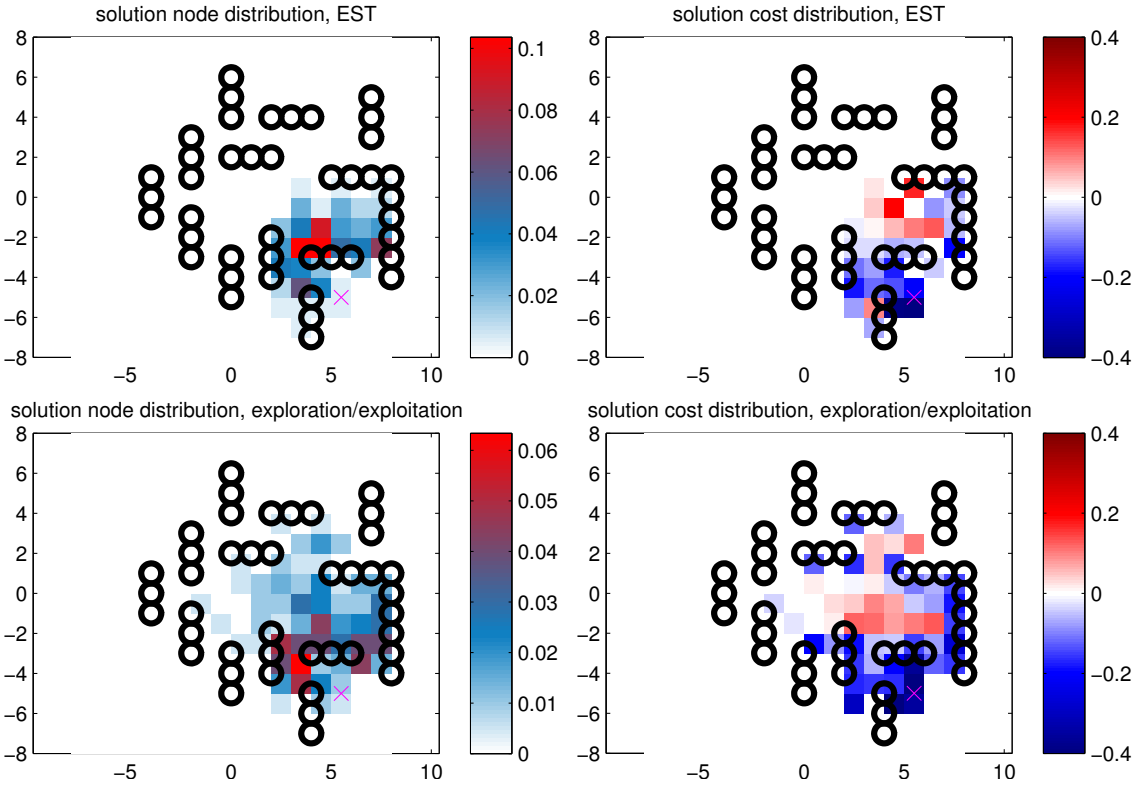

Figure 8.20: Spatial distribution of solution trajectory nodes for target 9 of setup B, showing the percental node count (left) and averaged obstacle avoidance cost decrease δc (right). Distributions for EST (top) and exploration/exploitation (bottom) heuristics are shown.

8.7 Optimal Solutions

The developed exploration heuristic is good for a fast discovery of solutions, using a limited number of local planning calls. However, the solutions found are not optimal, caused by two reasons: Motion optimization is only done locally and is dominated by exploration in conflicting situations. For practical motion planning applications it would be useful to have, at least close to, optimal solutions, for example minimizing the execution time or minimizing energy costs. Building upon the presented hybrid approach, two paths can be followed.

One possibility is to post-process the solutions. Taking the result of hybrid planning as an initial guess, trajectory optimization methods incrementally refine the solution until a minimum is found. The method of Toussaint et al. (2007) would be a promising approach, since it is able to operate on the same task representation as hybrid planning.

A second way is to combine exploration and optimization in a holistic approach. Berenson et al. (2011) propose to interleave exploration with gradient based optimization steps in order to keep the tree in low cost areas. This idea can be utilized for hybrid planning as well. While task space exploration proceeds as described, intermediate task targets could be incrementally changed towards lower costs with gradient descent steps. Although this would be relatively costly because it would involve a re-evaluation of the modified trajectories, the higher optimization costs could be counteracted by lower global exploration costs. Heuristics could be included that heavily bias paths with good optimization characteristics, concentrating the exploration towards optimal paths and effectively pruning less optimal parts of the tree. With respect to the exploration/exploitation weighting scheme, this means that exploitation weights of nodes are not fixed but can be updated by incremental

optimization steps. In figure 8.19, nodes can also move horizontally, which changes their weight according to optimization success.

It is important to note that these two optimization approaches can not guarantee to find the true global optimum. For instance, if multiple topological different solution paths are possible, exploration would randomly choose one and optimization would not be able to consider other possibilities. To overcome this limitation, all alternatives have to be discovered and considered, as for example practiced in a recent sampling-based approach for optimal motion planning (Karaman and Frazzoli, 2010). It relies on a constant re-wiring of tree nodes and selection of the shortest paths, a costly process, counteracting our premise of keeping the exploration and local planning calls low. Thus, for applications where true optimality is not the biggest concern, we argue that the solution of optimizing a solution found by sampling-based exploration is sufficient, if this enables us to find the solutions faster.

Chapter 9

Conclusion

This final chapter summarizes hybrid motion planning as introduced and evaluated in this thesis. Also, an outlook of possible further developments is given.

9.1 Hybrid Motion Planning

Hybrid motion planning combines motion control methods and ideas from motion planning. The key element is the use of a task representation layer, on which planning operates by specifying task level targets just as a general global motion planning approach. The role of motion control is to handle local planning between individual task targets, simultaneously optimizing the resulting trajectories locally. This way it is possible to overcome limitations of both local and global methods and to benefit from their strengths.

Widely used techniques for motion control, especially for redundant robots, were described in chapter 2. Although these methods are successful in generating locally optimal motions very fast, they might fail if subjected to local minima. Seen from the motion control perspective, hybrid motion planning adds a global planning layer to overcome this intrinsic limitation of local control approaches.

Motion planning, introduced in chapter 4, views the problem from a different perspective. Instead of treating the generation of movements as an optimization problem, it is formulated as a search process. While this solves arbitrary movement planning problems in a very general way, the high costs of an exhaustive search among all possible solutions render the approach impractical for many applications. From the motion planning perspective, hybrid planning counteracts the prohibitive costs of searching high dimensional state spaces by shifting the global search to a high level task representation. With respect to the underlaying state space, this high level representation constitutes a strong task induced heuristic that is able to rigorously reduce the searched space to a meaningful low-dimensional subset. By exploiting local optimization capabilities of control based local planning, the search can be effectively guided and planning can proceed on a coarser scale.

In chapter 5, two different strategies to search the task space according to sampling-based motion planning were evaluated. The first strategy (DRTask) adapts the popular rapid exploration heuristic in order to search the task space instead of the full state space. It can be shown that this strategy might lead to a unsuccessful exploration if local planning exhibits non-holonomic motion constraints. As a solution, an alternative sampling strategy (DRSim) is proposed, incorporating a measure of state space coverage into the exploration heuristic. The results of a small case study for a simple robotic manipulator reveal that the alternative strategy indeed shows a more stable exploration behavior and is thus better suited for the specific requirements of hybrid planning.

These insights into the interplay of global sampling and local planning influenced the implementation of hybrid planning for the application domain of redundant robots in chapter 6. In order to avoid the problems of exclusively biasing the search towards task space progress, a simple space invariant coverage heuristic is used in a hybrid random-tree

planning algorithm (TaskTree). Comparing the performance of the hybrid solution against a configuration space sampling algorithm (ConfTree) in a simulated motion planning setup shows the advantage of hybrid planning. Although a single exploration step is more costly because of the involvement of motion control methods, the reduced sampling effort due to the lower dimensionality of the sampling domain outweighs this and the overall runtime of TaskTree is lower.

The task definition of the simulation from chapter 6 was useful for the problem of reaching towards a fixed target. Chapter 7 demonstrated that task spaces can be used to describe more complex tasks as well, as shown for a bi-manual object placement job for a humanoid robot. This setup illustrates the benefit of employing a carefully chosen task representation, allowing to significantly lower the sampling effort. The redundant and not actively searched dimensions can be effectively exploited using local motion optimization, for instance to avoid obstacles. The experiment was done in a lab environment and showed the ability of hybrid motion planning to address real world problems.

Besides limiting global search to the task space, another strength of the hybrid approach is the possibility of exploiting local optimization abilities of motion control. Simulation results of chapters 6 and 7 showed that redundant space obstacle avoidance indeed helps to increase the set of problems that can be solved by increasing the reachability of local planning. The time needed to find a solution however was not generally lower. An explanation why this is the case is given by the increased probability of successful tree extensions if obstacle avoidance is used. The search tree can cover a larger portion of the state space and gets larger, resulting in an over-exploration of certain regions.

While this behavior was not respected in the space invariant exploration bias, a novel sampling heuristic, explicitly considering a variable exploration success, was developed in chapter 8. This is even more important if the relaxed motion control technique from chapter 3 is used for local planning. It allows to optimize a motion according to multiple primary motion objectives. Mixing obstacle avoidance and target reaching results in a local planner with an increased ability to circumvent obstacles, accelerating the planning process by guiding trajectories autonomously through problematic structures like narrow corridors.

Based on the TaskTree hybrid random-tree planning algorithm, principles developed during the discussion of task sampling strategies in chapter 5 are incorporated. The novel sampling heuristic systematically explores the task space by reducing the task dispersion locally and maintains a simultaneous global exploration of the state space. Combining exploration with an additional exploitation measure, a heuristic is created favoring trajectories with a greater exploration effect and with a larger contribution to reduce optimization costs. Evaluation shows the advantage of relaxed motion control to guide the search through corridors and demonstrates that the proposed exploration heuristic outperforms the previously used simpler heuristic. The impact of the combined exploration/exploitation heuristic however remains ambivalent. Planning can only be accelerated for some queries, for which relatively easy low cost solutions exist. Target sampling tends to be too sparse to capture less obvious cost minima.

The following list collects all key properties of hybrid motion planning, summarizing this thesis and describing a frame for possible future applications:

- Planning problems that need a global scope can be solved, where local methods are not sufficient. For example, problems involving local minima or possessing complex free space topologies can be addressed. It is not necessary to tailor problem specific solutions for specific use-cases, using a global planning layer enables to find solutions in a very general way.
- It is well suited for motion planning for redundant systems, where state spaces are

high-dimensional and the planning objective can be achieved with more than one or a whole continuum of states. It is not necessary to pre-determine a set of possible goal states, hybrid planning can discover any feasible goal during the search.

- Prohibitive costs of global planning can be reduced by limiting the search to high-level task specific representations. Depending on the task, these representations can be of much lower dimensionality than the full state space.
- It is possible to include motion optimization techniques that help local planning to avoid constraints or to achieve other objectives, like creating more efficient trajectories.
- A flexible hybrid motion planning framework based on sampling-based random-tree planning is proposed. The developed sampling strategies and weighting heuristics focus on an efficient exploration while being robust to sub-optimal or failed local planning and unmodeled non-holonomic behavior.

To be able to utilize hybrid planning, it is important to consider the following challenges for a successful application of the method:

- A sufficiently complete and precise model of the robot and the environment is needed. In comparison to plain local motion control, the horizon of available sensory data has to be larger since planning goes beyond the immediately visible. It is no longer possible to instantly react to changes like obstacle movements or localization errors without potentially invalidating the solution. Although a certain level of perturbations can be tolerated, coping with more substantial changes needs to re-evaluate and re-plan the solution.
- It is difficult to guarantee completeness of hybrid planning. The global search can be complete with respect to the high-level task representation, but the coverage of the underlying state space is subject to the local planning process. Dependent on the actually used method, it might be possible to show the fitness of local planning regarding a particular problem setup, but is appears unlikely that this is always possible.
- Due to the involvement of motion control, the implementation of hybrid motion planning is usually more challenging than classical motion planning. However, for many robotic platforms, motion control methods do already exist and it is relatively easy to include them into the framework.

9.2 Outlook

The framework as presented in this thesis can be used very generally for different applications, given there exist suitable local control methods and a model of the environment that can be used for simulation. The use for single-chain and dual-chain redundant robots was demonstrated. Another promising application would be multi-chain manipulators, for example robotic hands. Typically consisting of three or more fingers, hands have high-dimensional state spaces and complex constraints. With respect to typical tasks however, hand structures are highly redundant. The task of grasping an object for instance can be solved with different hand postures and contact points. It is difficult to treat grasp planning as an optimization problem, since many constraints are active: Constraints from the hand morphology and the object geometry as well as grasp contact constraints. Assuming that some limited control abilities are available, the hybrid approach might be a way of

incorporating a global planning component, enabling solutions to more complex grasp problems. For instance, a global exploration of finger contact points could be done on a lower dimensional task representation, while local planning generates matching finger and hand postures, obeying hand and contact constraints.

Throughout this work, well studied control methods were used as local planning components. They are generally very fast but need a precise model of the system. In practice, such an explicit modeling can become too expensive or impractical. Machine learning techniques can then be used used to estimate a control model based on empirical data. The task representation might be given in a less explicit way, but nonetheless it is possible to use hybrid planning. In particular, the additional global planning layer can help to counteract failures of local planning, which are likely to happen with an often imperfect learned model. Sampling on learned or only implicitly defined task manifolds can be more complicated, but there exists methods addressing this issue, for example see (Porta and Jaillet, 2011).

In other cases, a fixed model is not desired, sometimes variety is included on purpose. If motion models are derived from human movement data for example, variations during execution of a task should be captured as well. Corresponding task representations can then be defined in a probabilistic way, containing both the means and the variances, refer to (Mühlig et al., 2009). These models could be used as importance probability distributions, guiding the global exploration process according to observed behaviors. The search space can be heavily narrowed down in low variance areas, while other areas with higher variance can be explored more generously. As a result, the search space is effectively reduced based on the meaningfulness with respect to an observed task and by biasing motion planning to this space, more natural behaviors can be created.

Appendix A

Integrating the Attractor Dynamic

The second-order attractor dynamic constraint

$$\ddot{y} + \beta\dot{y} + \alpha y = \alpha u(t) \tag{A.1}$$

can be treated as an ordinary second-order differential equation. A homogeneous solution is then given by

$$y_h = C_1 e^{-t} + C_2 t e^{-t} \tag{A.2}$$

choosing $\alpha = 1$ and $\beta = 2\sqrt{\alpha} = 2$. For the right hand term $u(t)$ two cases have to be differentiated, the moving attractor $u(t) = t$ for $0 \leq t \leq 1$ and the steady attractor $u(t) = 1$ for $t > 1$. Two general solutions of the differential equation are then

$$y_1 = C_1 e^{-t} + C_2 t e^{-t} + t - 2 \qquad 0 \leq t \leq 1 \tag{A.3}$$

$$y_2 = C_3 e^{-t} + C_4 t e^{-t} + 1 \qquad t > 1 \tag{A.4}$$

Setting the initial values $y_1(0) = 0$, $\dot{y}_1(0) = \nu$ and $y_2(1) = y_1(1)$, $\dot{y}_2(1) = \dot{y}_1(1)$ yields

$$y_1 = 2e^{-t} + (\nu + 1)te^{-t} + t - 2 \qquad 0 \leq t \leq 1 \tag{A.5}$$

$$y_2 = (2 - e)e^{-t} + (\nu + 1 - e)te^{-t} + 1 \qquad t > 1 \tag{A.6}$$

These two equations represent an explicit solution of the constraint equation, describing the resulting trajectory depending on the initial conditions $y_1(0)$ and $\dot{y}_1(0)$. Since the initial velocity ν is present in the solution of the dynamic constraint, it is not possible to express the trajectory without knowing ν. The constraint can not be completely integrated to a velocity independent form.

Bibliography

M. Behnisch, R. Haschke, and M. Gienger. Task space motion planning using reactive control. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2010. 72, 74, 76

M. Behnisch, R. Haschke, M. Gienger, and H. Ritter. Deformable trees - exploiting local obstacle avoidance. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2011. 25

D. Berenson, T. Simeon, and S. Srinivasa. Addressing cost-space chasms in manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011. 91, 110

D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006. 63

C. Bishop. *Pattern recognition and machine learning*. Springer New York, 2006. 40

S. Brin. Near neighbor search in large metric spaces. In *21th International Conference on Very Large Data Bases*, 1995. 71

O. Brock and L. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2001. 62

O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031, 2002. 24, 34

B. Burns and O. Brock. Single-query motion planning with utility-guided random trees. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3307–3312, 2007. 38

S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(2):286–298, 2007. 6

P. Cheng and S. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Proceedings of the IEEE-RSJ Conference on Intelligent Robots and Systems*, pages 43–48, 2001. 41, 91

S. Chiaverini, G. Oriolo, and I. Walker. Kinematically redundant manipulators. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 245–268. Springer, 2008. 8

H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005. 37

S. Dalibard and J. Laumond. Linear dimensionality reduction in random motion planning. *The International Journal of Robotics Research*, 30(12):1461–1476, 2011. 41, 63

A. De Luca and G. Oriolo. Nonholonomic behavior in redundant robots under kinematic control. *IEEE Transactions on Robotics and Automation*, 13(5):776–782, 1997. 19, 21

R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner. Bispace planning: Concurrent multi-space exploration. In *Proceedings of Robotics: Science and Systems IV*, 2008. 63

A. Escande, N. Mansard, and P. Wieber. Fast resolution of hierarchized inverse kinematics with inequality constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3733–3738, 2010. 23

N. I. Fisher, T. Lewis, and M. E. Willcox. Tests of discordancy for samples from fisher's distribution on the sphere. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 30(3):pp. 230–237, 1981. 70

M. Gharbi, J. Cortes, and T. Simeon. Roadmap composition for multi-arm systems path planning. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 2471 –2476, 2009. 81

M. Gienger, H. Janssen, and C. Goerick. Task-oriented whole body motion for humanoid robots. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2005. 7, 14, 15, 69, 70

M. Gienger, H. Janssen, and C. Goerick. Exploiting task intervals for whole body robot control. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2006. 24

M. Gienger, M. Toussaint, and C. Goerick. Whole-body motion planning – building blocks for intelligent systems. In K. Harada, E. Yoshida, and K. Yokoi, editors, *Motion Planning for Humanoid Robots*, pages 67–98. Springer London, 2010. 11, 12, 14

K. Gochev, A. Safonova, and M. Likhachev. Planning with adaptive dimensionality for mobile manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2944–2951, 2012. 63

K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa. An analytical method on real-time gait planning for a humanoid robot. In *Proceedings og the IEEE-RAS International Conference on Humanoid Robots*, 2004. 12

P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107, 1968. 36

D. Hsu, R. Kindel, J. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233, 2002. 38, 50, 64, 72, 93

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, 15:1523–1530, 2002. 6

L. Jaillet and T. Simeon. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. *The International Journal of Robotics Research*, 27(11-12):1175, 2008. 91

L. Jaillet, J. Cortes, and T. Simeon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635 –646, 2010. 91, 96

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011. 34

S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems*, June 2010. 111

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*, 12(4):566–580, 1996. 40

C. C. Kemp, P. Fitzpatrick, H. Hirukawa, K. Yokoi, K. Harada, and Y. Matsumoto. Humanoids. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, chapter 56, pages 1307–1333. Springer, 2008. 12

O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1):90, 1986. 33

J. J. Kuffner, Jr and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, 2000. 39

A. Ladd and L. Kavraki. Fast tree-based exploration of state space for robots with dynamics. *Algorithmic Foundations of Robotics VI*, pages 297–312, 2005. 41, 91

E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. *Technical Report TR99-018, Department of Computer Science, University of North Carolina*, 1999. 16

J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991. 17, 33, 35

S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Technical Report, Computer Science Dept, Iowa State University*, 1998. 37

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. 18, 35, 36, 37, 40, 81

S. M. LaValle and J. J. Kuffner, Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000. 37, 98

Y. Li and K. Bekris. Balancing state-space coverage in planning with dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3246–3253, 2010. 41

A. Liegeois. Automatic supervisory control of configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man and Cybernetics*, 7(12):861–871, 1977. 9, 11

S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing voronoi bias in rrts. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004. 46, 47, 48, 49

J. Maycock, B. Bläsing, T. Bockemühl, H. J. Ritter, and T. Schack. Motor synergies and object representations in virtual and real grasping. In *1st International Conference on Applied Bionics and Biomechanics*, 2010. 6

M. Mühlig, M. Gienger, S. Hellbach, J. Steil, and C. Goerick. Task-level imitation learning using variance-based movement optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1177–1184. IEEE, 2009. 116

Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Co., Inc., 1991. 7, 8, 34

J. Pan, L. Zhang, and D. Manocha. Collision-free and curvature-continuous path smoothing in cluttered environments. In *Proceedings of Robotics: Science and Systems VII*, 2011. 34

J. Phillips, N. Bedrossian, and L. Kavraki. Guided expansive spaces trees: a search strategy for motion-and cost-constrained state spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004. 38

E. Plaku, K. Bekris, and L. Kavraki. Oops for motion planning: An online, open-source, programming system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007. 64, 68

J. Porta and L. Jaillet. Path planning on manifolds using randomized higher-dimensional continuation. *Algorithmic Foundations of Robotics IX*, pages 337–353, 2011. 116

N. Ratliff, M. Zucker, J. A. D. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, May 2009. 34

M. Rickert, O. Brock, and A. Knoll. Balancing exploration and exploitation in motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2812–2817, 2008. 63, 91

J. Salini, V. Padois, and P. Bidaud. Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1283–1290, 2011. 24

S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. *Adaptive motion of animals and machines*, page 261, 2006. 6

J. Scholz and G. Schöner. The uncontrolled manifold concept: identifying control variables for a functional task. *Experimental Brain Research*, 126(3):289–306, 1999. 6

G. Schöner, M. Dose, and C. Engels. Dynamics of behavior: Theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, 16(2-4):213–245, 1995. 6

L. Sciavicco and B. Siciliano. *Modelling and control of robot manipulators*. Springer Verlag, 2000. 6, 8

T. Shamir and Y. Yomdin. Repeatability of redundant manipulators: Mathematical solution of the problem. *IEEE Transactions on Automatic Control*, 33(11):1004–1009, 1988. 21

A. Shkolnik and R. Tedrake. Path planning in 1000+ dimensions using a task-space voronoi bias. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009. 64

J. Steffen, R. Haschke, and H. Ritter. Towards dextrous manipulation using manipulation manifolds. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 2738–2743, 2008. 6

I. Sucan and L. Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *Algorithmic Foundations of Robotics VIII: Selected Contributions of the Eighth International Workshop on the Algorithmic Foundations of Robotics*, volume 57, page 449, 2009a. 41, 91

I. Sucan and L. Kavraki. On the performance of random linear projections for sampling-based motion planning. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2009b. 63

H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time self collision avoidance with whole body motion control for humanoid robots. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, 2007. 15

H. Sugiura, M. Gienger, H. Jannsen, and C. Goerick. Reactive self collision avoidance with dynamic task prioritization for humanoid robots. *International Journal of Humanoid Robotics*, 7(01):31–54, 2010. 15, 24

M. Toussaint, M. Gienger, and C. Goerick. Optimization of sequential attractor-based movement for compact movement representation. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007. 14, 34, 110

N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann. Integrated grasp and motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2883–2888, 2010. 63, 81

J. van den Berg. *Path planning in dynamic environments*. PhD thesis, Utrecht University, The Netherlands, 2007. 40

G. Van den Bergen. A fast and robust gjk implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999. 16

M. V. Weghe, D. Ferguson, and S. Srinivasa. Randomized path planning for redundant manipulators without inverse kinematics. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007. 63, 66

S. Wilmarth, N. Amato, and P. Stiller. Maprm: a probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999. 91

Y. Yang and O. Brock. Elastic roadmaps–motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 2010. 62

A. Yershova, L. Jaillet, T. Simeon, and S. LaValle. Adaptive tuning of the sampling domain for dynamic-domain rrts. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 2851–2856, 2005. 41, 91

J. Zhang and A. Knoll. An enhanced optimization approach for generating smooth robot trajectories in the presence of obstacles. In *Proceeding of the European Chinese Automation Conference*, pages 263–268, 1995. 34