Motion Planning for Manipulators with Many Degrees of Freedom – The BB-Method

Boris Baginski

Fakultät für Informatik
der Technischen Universität München

Lehrstuhl für Robotik und Echtzeitsysteme

Motion Planning for Manipulators with Many Degrees of Freedom – The BB-Method

Boris Baginski

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. M. Paul
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. H.-J. Siegert
2.	Univ.-Prof. Dr. Dr. h.c. W. Brauer

Die Dissertation wurde am 24. Juni 1998 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 31. August 1998 angenommen.

Preface

The topic of this book – automatic path and trajectory planning – is an important area of robotics. In this decade robotics – similar to some other areas of computer science – was shaped by striving for systems with more intelligence, more autonomy, and more freedom of choice. The conception of such inherently complex systems posed new challenges: in research as well as in the reorientation process of the often traditionally shaped application domains.

In industrial manufacturing these efforts were strengthened and supported by the target of an integrated computer support comprising all steps from design to product. Due to these efforts more and more products and their manufacturing production lines are completely modelled with CAD-tools.

Due to this, product properties cannot be determined only when a prototype is available, but already during the development process by using computer simulation. Examples from the car manufacture are driving characteristics, crashtests, design of manufacturing cells or digital mock-up. This extensive use of computers leads to cost reduction and shorter development cycles.

Within the industrial area we thus find robots, which operate in a well-known – because described by CAD-models – real or simulated environment. In the real environment up to today the robots usually still move only on predefined paths. Unexpected errors cause a halt in the production. In the future autonomy will be used to handle unexpected error situations. Among other tools path planning is mandatory for this. In the simulated environment, for example during the design of the manufacturing cells or within mock-up simulation, suitable collision-free paths have yet to be found. Here computer-aided path planning, for example with the algorithms developed by Mr. Baginski, is important.

Computer-aided path planning is helpful or even indispensable in many other application domains. In order to show the diversity of the domains, the following scenarios are mentioned:

- A multilegged and highly articulated robot executes inspection work in a sewer system (a partially well-known environment).
- A service robot in a hospital delivers meals. It knows the static objects and recognizes moving objects by sensors, e.g. a camera. The robot must find its way to the target and then move his arms to place the meal correctly.
- Virtual humans or other objects with joints, move in a virtual environment to a goal position to grasp an object. The movement of the human altogether and the movement of the individual parts of the body have to be calculated with the constraint "the movement must look natural".

In many scenarios pertinent to industrial manufacturing, paths for robots with six or more joints can now be calculated within a short time by the BB-algorithm presented in this book.

Nevertheless there are still many other path planning problems without satisfying algorithms to solve them. An example is path planning in very tight and complex obstacle spaces, which are typical for digital mock-up simulation. Another example is path planning for hyperredundant robots under many, also nongeometrical boundary conditions. In the last section of this book Mr. Baginski shortly points to some unsolved problems and his ideas for extending the BB-method to their solution.

I would be happy to see a rapid transfer of Mr. Baginski's algorithms for path planning into the industry and hope that many new impulses for research in path planning are triggered by this book.

Munich, October 26, 1998 Hans-Jürgen Siegert

Abstract

A new approach for manipulator motion planning - the BB-method - is introduced. A collision-free path is found by incrementally modifiying an initial, usually colliding, path. The method described alters the robot's motion in its physical workspace, utilizing *virtual* rating functions that are based on *reducing* and *expanding* the robot's geometry model. The BB-method is able to plan motions for redundant and hyperredundant manipulators. With this, it stands in contrast to most other motion planning methods usually developed in an abstract configuration space. Due to their complexity, they are not applicable to real robot tasks.

The experimental experiences with the BB-method prove the main thesis of this work: an efficient and effective motion planning scheme can be build upon careful examination and utilization of the *spatial properties* of the motion of a robot manipulator. The algorithmic complexity of collision free path planning is only linear in the number of degrees of freedom, and the effective computational effort is not necessarily increased for an increased number of joints. The BB-method is local and heuristic, therefore it is an incomplete motion planning algorithm. Within this work, all major properties of the BB-method are characterized and analysed in detail.

This work is a basis for further explorations of the various properties, possibilities, extensions and further developements of the BB-method. But nonetheless, efforts were taken to identify all necessary requirements of a motion planning system and to introduce the pradigm of *virtual geometry modification* in a way all aspects can be handled. The existing prototype implementation allows real-time planning (up to a few seconds) of motions for typical six degree of freedom industrial manipulators in practical cases, as well as fast planning for multiple kinematic devices with several tens of joints. The planning allows the assertion of safety distances to handle control and modelling uncertainties, and a local length reduction of the path, yielding motions that require less time and less energy if they are executed in reality.

Kurzfassung

Das BB-Verfahren, ein neuartiges Verfahren zur Bewegungsplanung für Manipulatoren, wird vorgestellt. Eine kollisionsfreie Bahn wird gefunden, indem eine initiale, üblicherweise kollisionsbehaftete Bahn schrittweise verändert wird. Das dargestellte Verfahren modifiziert die Roboterbewegung im Arbeitsraum, unter Ausnutzung *virtueller* Bewertungsfunktionen, die durch *Verkleinerung* und *Vergrößerung* des Geometriemodells des Roboters bestimmt werden. Das BB-Verfahren ist auch für redundante und hyperredundante Manipulatoren anwendbar. Es unterscheidet sich damit ganz wesentlich von den meisten anderen aus der Literatur bekannten Ansätzen. Diese basieren überwiegend auf der Planung in einem abstrakten Konfigurationsraum und sind aufgrund ihrer Komplexität nicht für realistische Anwendungen geeignet.

Die Ergebnisse der Versuche mit dem BB-Verfahren belegen die Kernthese dieser Arbeit: ein leistungsfähiges und praktisch anwendbares Bewegungsplanungssystem kann durch geeignete Auswertung und Nutzbarmachung der *räumlichen Eigenschaften* der Bewegungen eines Manipulators entwickelt werden. Die algorithmische Komplexität der Planung kollisionsfreier Bahnen ist nur linear in der Anzahl der Freiheitsgrade, und der effektive Planungsaufwand wird durch Hinzufügung zusätzlicher Gelenke nicht notwendigerweise vergrößert. Das BB-Verfahren ist ein lokales und heuristisches Planungsverfahren, es ist daher unvollständig. In dieser Arbeit werden alle wesentlichen Eigenschaften des Verfahrens beschrieben und detailliert analysiert.

Diese Arbeit versteht sich als Basis für die weitere Untersuchung der Eigenschaften, Möglichkeiten, Erweiterungen und Ergänzungen des BB-Verfahrens. Nichtsdestotrotz wurde versucht, alle notwendigen Teilaspekte eines Bewegungsplanungssystems zu identifizieren und das Paradigma der *virtuellen Geometrieverformung* in einer Weise einzuführen, daß alle relevanten Aspekte berücksichtigt werden können. Die vorliegende prototypische Implementierung ermöglicht die Planung von Bewegungen für typische sechsgelenkige Industrieroboter in praktischen Anwendungsbeispielen in Echtzeit (maximal wenige Sekunden), und ebenso sehr schnelle Planung für komplexe (auch gegabelten) kinematische Strukturen mit beliebig vielen Freiheitsgraden. Dabei können Sicherheitsabstände eingehalten werden, die Modellierungs- und Steuerungsunsicherheiten berücksichtigen, und die Bahnen können lokal verkürzt werden, was in der Praxis zu verringerten Verfahrzeiten und verringertem Energieaufwand führt.

Acknowledgements

First, I would like to thank Prof. Dr. H.-J. Siegert for his encouragement and support during my years at his chair. I enjoyed the hours of discussion very much.

I gratefully acknowledge the support of my colleagues and friends at the institute (in no specific order): Andrea Baumann, Andreas Koller, Oliver Schmid, both Stöhrs, Dr. Gerhard Schrott, Stefan Riesner, Thomas Weiser, Martin Eldracher, Till Brychcy, and Dr. Max Fischer.

Special thanks go to the students who spent months with me, listened to my ideas and realized them with enthusiasm: Qiudong Ji, Alex Diebald, Matthias Riesch, Reinhard Haslbeck, Dirk Gottschling, Manfred Hockauf, Richard Lechner, Spyridon Kolssouzidis and the others whom I forgot to mention.

Kristina Degn deserves special thanks for her tireless work and skills in proofreading this thesis.

Finally, I thank Toni for her patience, love, and inspiration of different points of view when discussing all aspects of this work.

Contents

List of Figures

List of Tables

List of Symbols

Formula Symbols

The following convention is used: vectors and sets of vectors are written in a lower case bold typeface ($\boldsymbol{\nu}$), matrices and vector-valued maps are written in an upper case bold typeface ($\mathbf{M}$), and scalar values are written in italics (d).

$\mathbf{u}$, $\boldsymbol{\nu}$, $\boldsymbol{w}$	Cartesian points
G_i	Geometry model of link i
G^{+d}	Model isotropically expanded by d
$\mathbf{R}$	Rotation matrix
$\mathbf{t}$	Translation vector
${}_i\mathbf{T}_j$	Homogeneous transform
${}_i\mathbf{T}_j(\mathbf{q})$	Configuration-dependent transform
${}_i^s\mathbf{T}_j$	Scaling transform, scaling factor s
n	Number of joints / links
n_i	Number of joints preceeding link i
v_i	Joint preceeding joint i / link preceeding link i
I_i	List of joints preceeding link i
W	Workspace
W_{free}	Free workspace
q_i	Joint value of joint i
$\mathbf{q}$	n-dimensional configuration
$\mathbf{q}_{start}$, $\mathbf{q}_{goal}$	Start/goal configurations
C	Configuration space
C_{free}	Free configuration space
C_{coll}	Colliding configuration space
$\mathbf{P}$	Path
$\bar{\mathbf{P}}$	Linear path segment
$\hat{\mathbf{P}}$	Polygonal path
$\delta_{\min}$	Smallest displacement for path planning
$\delta_{\max}$	Largest displacement for path planning
$\Delta_i(\mathbf{q}_a, \mathbf{q}_b)$	Estimated cartesian motion for link i
$\mathbf{c}^a$	Replacement candidate set for configuration $\mathbf{q}_a$

f_{step}	Relative displacement step size
f_{divide}	Path segment subdivision factor
z_{random}	Maximum number of random subgoals
$d_{\max_i}$	Desired safety distance for link i
$\mathbf{d}_{\max}$	n-dimensional vector of desired safety distances
$\mathbf{d}_{\bar{\mathbf{p}}}$	Safety distances of a path segment
ρ	Threshold ratio for path shortening
λ	Segment length threshold for path shortening
τ	Tolerance for collision detection
γ_{scale}	Approximation precision for scaling based rating
γ_{dist}	Approximation precision for expansion based rating
t_{scale}	Planning time for collision-free path
t_{dist}	Planning time for safety distances
t_{short}	Planning time for path length reduction
m_{dist}	Approximate quality of safety distances
m_{short}	Length ratio achieved by length reduction

Algorithms

To clarify the presentation, prototype algorithms are used throughout this work. A PASCAL-style notation is used for the control structures.

Q_{scale}	Scaling based path rating (p. 51)
ModifySegmentScale	Scaling based path modification (p. 61)
DivideSegmentScale	Path segment subdivision (p. 62)
PlanScale	Collision free path planning (p. 63)
PlanGlobal	Random global planning (p. 74)
Q_{dist}	Expansion based path rating (p. 79)
ModifySegmentDist	Expansion based path modification (p. 83)
DivideSegmentDist	Path segment subdivision (p. 85)
PlanDist	Planning safety distances (p. 87)
CutTriangle	Local path length reduction (p. 99)
PlanShort	Length reduction planning (p. 100)
SegmentColl	Collision test for a path segment (p. 109)
SweptObjColl	Collision test for a moving object (p. 114 and 118)
SweptStepColl	Collision test for an intermediate segment (p. 117)
ObjColl	Collision test for two objects (p. 118)
BisectScale	Approximation of scaling (p. 120)
SweptObjGetScale	Scaling of a moving object (p. 121)

Chapter 1

Introduction

This chapter shows the necessity of manipulator motion planning. The objectives are developed by considering the relevant aspects of manipulator motion and possible planning applications. The chosen approach is informally introduced. A summarizing overview of the whole work and the achieved results is given.

1.1 Motivation

The motions of robot manipulators in industrial applications are still programmed by human operators. This is usally done by manual *teach-in* of a sequence of positions, either off-line using a simulation system or by moving the robot itself with a manual control device. The teached positions are connected by a fixed approximation scheme at execution time. This method requires skilled operators, is time consuming, and costly. It is only justified for repeatedly executed manufacturing tasks.

To allow a more flexible use of robots, it is desired to extend the capabilities of the simulation systems to enable them to automatically plan collision-avoiding motions. This should happen at interactive rates, enabling the operator to program complex production tasks in short time.

In the future an even higher level of automatization is intended. In the manufacturing domain, *task-level programming* is the ultimate objective. This means that the task that shall be fulfilled by a robot – or even the task of a complete manufacturing system consisting of several robots and other devices – will be specified by the results only. The *autonomous manufacturing system* will decompose the task by means of planning and scheduling, creating a complex sequence of operations. By taking into account the knowledge of its own abilities, the input of sensors that supply information from the environment, the desired results, and additional global constraints, the whole production process needs only little human interaction.

For this development one essential low-level planning system is the *motion planning system*. Given information of the objects surrounding it, a robot has to be able to plan and execute its own motions without collision, without other interferences, and with high efficiency.

Looking at the motion of a human arm that avoids obstacles with unconscious ease, the problem appears to be a very simple one. On the technical side, there exist a surprisingly large number of control parameters to make a manipulator move, and so the problem turns out to be of high complexity and is almost incalculable. This is the reason why there is no motion planning system – at least no satisfactory one – in practical use for manipulators with more than 4 or 5 joints. Of course, motion planning gets more complex when the robot has more joints. On the other hand, more joints allow the robot to move more freely, and motion planning should use this freedom. The thesis to be verified in the following is:

An efficient motion planning algorithm can be developed based upon a consequent examination and utilization of the geometry of the robot's motion among the obstacles within its workspace.

To point out this idea more clearly: Motion planning shall be addressed from the viewpoint of *real motion*, the spatial sweeping of kinematically chained rigid bodies among stationary bodies. The *traditional* search space of motion planning, the parameter or configuration space, is just seen as means of visualization and understanding. In its turn, this work follows Latombe's challenge: to produce "incomplete methods with well-defined and reasonable failure conditions" (Latombe (1991) p. 588). It is desired to find an algorithm – although local, heuristic, and due to fail in certain circumstances – that quickly solves *most of all* practical tasks with extremely low average complexity, has a solid theoretical foundation, and knows a *back door* to overcome its own locality.

1.2 Objectives

This work aims at motion planning for manipulators in known, obstructed environments. To be useful in practical applications, a motion planner has to fulfil the requirements that arise from the available data of the robot and its environment, the possible tasks it has to solve, and the desired results it has to obtain. These aspects are discussed to set up the objectives of this work.

1.2.1 Sample Robot Applications

The possible area of application for motion planning as addressed within this work can be summarized as follows: the environment is known and *static* for a single planning task, but it is not *fixed* for an arbitrary large number of planning tasks. The activities of the manipulator itself modify the environment as loads are transported, closures are opened etc. Thus, each planning task may be based on a different environment, but within the task itself, the obstacles are at known positions and do not move.

Two relevant application domains are described as examples for the intended area of application. These domains are industrial manufacturing and hazardous

environments, typical domains where automated motion planning is desired, as it can increase the usability and flexibility of the employed robots.

Robots in Manufacturing Tasks

Nowadays, manipulators are already widely used in automobile manufacturing. There, the typical loads (arbitrarily shaped metal parts) and the used tools (e.g. welding devices, usually with an inherent degree of freedom) are very large and often lengthy shaped. In addition, the workcells are quite tight. Therefore, the required motions are very complex.

The "traditional" manipulator operating in industrial manufacturing is a six-joint robot (6 degrees of freedom = 6 DOF), where typically the first three links have a long extent and contribute mostly to the position of the tool. The last three links have very short radii and are mainly responsible to set up the orientation of the tool. To solve complex manufacturing tasks the orientation of the tool is important. Thus it is required to plan motions in all degrees of freedom (a decomposition, as e.g. suggested in Brooks (1983), is not sufficient).

One problem of the "classical" use of manipulators in industrial manufacturing is the fixed arrangement of the workcells with one or several manipulators and additional handling devices. For more flexible manufacturing, mobile platforms with manipulators have been proposed. These systems are able to transport material and do different and changing manipulation tasks at different locations. Motion planning for mobile manipulators consists of planning transfer movements of the mobile platform in three degrees of freedom and planning for the manipulator (in 6 or more DOF) itself.

The "traditional" manufacturing is mass production. The same motions are repeated over and over again for a long time and manual teach-in is used for programming. Flexible manufacturing or mobile manipulators require automated motion planning to be operated. The robots have to plan motions among static, known obstacles. All geometry data needed for automatic motion planning is available. The use of computers has led to a situation where all parts that are handled and all parts of the workcells – even the robots – are constructed with CAD-systems. The precision of this model data is extremly high.

Robots in Hazardous Environments

In certain locations, robots are a better choice than humans. It may be either too dangerous or too expensive for people to work in these locations. For maintainance within nuclear power plants manipulators that are integrated in the most dangerous locations have been proposed (see e.g. Paljug et al. (1995)). To be able to operate freely in environments full of obstacles, robots that are developed for these applications have more degrees of freedom than industrial manipulators.

So-called redundant (up to 8 DOF) and hyperredundant manipulators (with even more joints) allow operations around corners, through openings and on all

sides of an object of interest. With 10, 15 or more DOF and a less intuitive arrangement of links and joints than the industrial manipulators, these robots can hardly be teleoperated, therefore motion planning is needed. As in the manufacturing domain, the required geometry data is available with high precision.

Quite similar conditions can be found in space laboratories, another area of application where robots are – or will be, due to intended cost reduction – preferred to humans (see e.g. Hirzinger (1993), Ackermann and Hirzinger (1996)). All environment geometry is known. The required motions depend on the ongoing experiments and maintainance tasks. A motion planning system that avoids any possibly damaging collision and allows a safe operation of the space laboratory is as important in this application as its use in the power plant mentioned above.

1.2.2 The Context of Motion Planning

Motion planning is just one component of a future intelligent robot system. Figure 1.1 sketches the possible context of a motion planner module. Any task the motion planner has to fulfil is projected by a higher level planning system. The task is described by a start and a goal posture of the robot and, possibly, supplemental information. It is fed into the motion planning system. Here it is solved using the geometry and other required data supplied by the world model. The solution is passed to the trajectory generation and motion execution module that controls the robot itself. The identified components and their connections to the motion planner will be described in the oncoming paragraphs. The concepts are presented in a simplified fashion. The underlying intention is to create a reasonable framework that allows to derive required features of a motion planning system. Additional components, like grasp planning or the planning of sensor-guided activities, can be added easily, but are not considered in this work (for an extended discussion see section 9.2, p. 158). A comparable scheme is presented in Hörmann and Rembold (1991), for example.

Higher Level Planning

The tasks are supplied by a higher level planning system. This may either be an automated task-level planner or – a human operator. The higher level planning system maintains the world model. It commands the motion planner and uses its responses for further planning. Possible tasks and responses are:

- **Plan motion**
  The motion planner connects a given start and goal posture. The typical task, either transportation of loads or manipulation of objects starts and ends very close or even "in touch" with obstacles. These kind of tasks are typical for the applications outlined above. The response for the higher level system is either success or failure. In case of successful planning, the result can be passed to the trajectory generation.

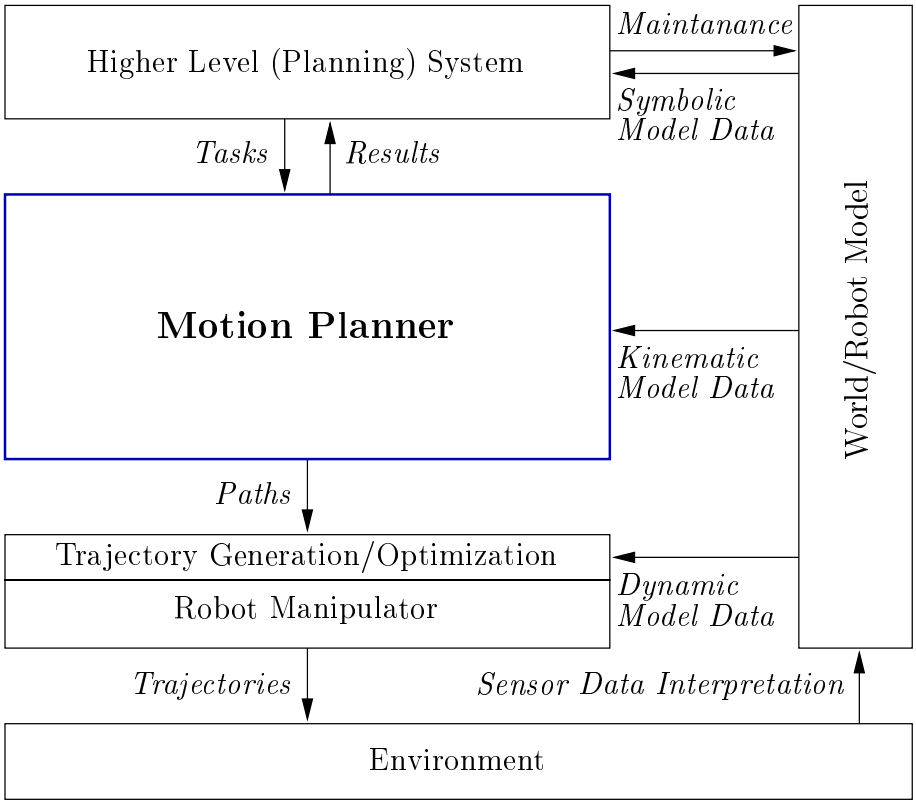

Figure 1.1: Principle embedding of a motion planning component in the context of an intelligent or interactive robot system.

- **Plan motion with constraints**
  There are tasks that are not completely specified by start and goal. Constraints limit the motion of the robot and have to be considered in planning. Typical constraints are:
  - **Task constraints implied by the load** – If, for example, an open filled container has to be transported, it has to be kept in an upright position at any time.
  - **Task constraints related to the environment** – An example for this kind of constraint is the supervision of an object with a camera that is fixed at the robot's tool. The object has to be kept "in sight" during the whole motion.
  - **Dynamic constraints** – All kinds of constraints that explicitly involve time to be expressed are called dynamic. These constraints can either be implied by the robot, e.g. keeping certain velocity bounds for the joints, or they are implied by the environment, e.g. another, independently moving robot is to be considered.

  Constraints have to be specified by the higher level planning system. Typically, certain tolerance intervals for constraints are allowed for the resulting motion. The response of the motion planner will be either success, failure or a certain rating of the degree of constraint-satisfaction that has been achieved for the planned motion.
- **Plan motion using partial/incomplete/uncertain solutions**
  The higher level planning system should be able to influence the planning by supplying partial or incomplete solutions. For example, a future intelligent planner might infer some kind of rough motion sketches from its "fuzzy" topological knowledge to add it to the planning input (this is an active area of research, see e.g. Mukerjee (1996) or Liu (1996)).
  It may as well be possible that the higher level system stores (partial) solutions and the motion planning component should include them in the planning, as this might simplify the planning process. Or the stored solutions have been planned for a possibly different arrangement of obstacles (or a different load) and are to be retested and adapted to the new situation.
  If the motion planning system is part of an operator-controlled robot programming system, the possible solution might be obvious in part and it may be desired to guide the planning process. Task extensions like this can be commanded by a sequence of not binding intermediate positions or path segments, supplied in addition to start and goal. The response is again success or failure.
- **Interactive planning**
  The task description consisting of partial/incomplete or uncertain solutions may be extended to full interactive planning. Motion planning, especially if constraints are involved, may turn into a "dialogue" between the higher level planning system and the planner. The higher level may attempt to

solve a certain task with a low-toleranced constraint. If this fails, the constraint is relaxed and planning is restarted. This implies the capability of "interactive planning" that allows to use previously achieved results if the task is modified slightly, as complete replanning is very expensive.
Interactive planning is of interest for human operated systems as well. The operator may want to modify failed tasks, use only partial solutions etc. The responses of the planning system have to report the relevant aspects of the internal state.

This list is just a rough sketch, but it illustrates the scope of possible tasks for a basic motion planning system. If several robots or other devices are involved, task descriptions get more complex. Additional aspects are discussed at the end of the work (chapter 9, p. 153).

World Model

The world model contains all relevant infomation about the environment and the robot. The higher level system manages it, typically in a symbolic fashion. Sensors can be used to update or modify the contained data. The motion planning component uses just the information it needs: mainly geometry data, describing the robot and the obstacles, and kinematic information, modelling the robot. For the application in the areas described above, the geometry is usually available as CAD-data and inherently very accurate. If parts of the obstacles are sensor scanned the modelling is less accurate. In this case, the motion planning system should be able to take the respective *modelling uncertainty* into account. This can be done by keeping a certain distance between the moving robot and the obstacles, which decreases the probability of collision down to a degree that is acceptable for the higher level planning system.

Trajectory Generation

The result of the motion planning system is passed to the trajectory generator to be performed by the robot. This result is typically a (more or less) close sequence of *via-points*, the so-called path, which is postprocessed (usally on-line) to control the manipulator's motion.

From a technical point of view, a manipulator is a heavy physical device equipped with (usually electric) motors and encoders to move the joints. The power applied to the motors is controlled through feedback of the position encoders. In principle, the trajectory generator takes three via-points at a time and calculates, using a control function, a time sequence of joint positions (with frequencies of 40 Hz up to 1 kHz) that approximates the path. Within each step, the current positions and the desired positions are compared and the power of the motors is adapted. Among the aspects considered by the control function are joint limits, velocity limits, acceleration limits, and inertial and gravitational forces. More advanced control functions can e.g. minimize motion time, minimize

the overall energy consumption or reduce the forces applied to the joints. The differential equations to formulate such an optimal control can be found e.g. in Craig (1986) or, updated, in Sciavicco and Siciliano (1996), newer results are reported in Stryk and Schlemmer (1993).

The manipulator starts to move at the start position and finally stops at the goal position. Normally, the via-points are not reached, as the motion passes them in a certain distance. This means that the robot does not move *exactly* to where it was intended to move by the motion planner. This divergency is referred to as *control uncertainty*. It depends on the speed of motion and the directional change implied by the via-points. To achieve a short motion time as usually desired, the planner should preferably avoid unnecessary detours and sharp bends within the sequence of via-points that make up the solution path.

The motion planner has to allow for the trajectory generator which varies the planned motion – without yielding collisions. Therefore it has to be supplied with an overestimation of the magnitude of control uncertainty and keep a respective distance between the robot and the obstacles.

1.2.3 Objectives and Requirements

From the context of a practical motion planning system as sketched above, a set of requirements, objectives and desired extension possibilities were selected for this work. The following requirements are reasonable design decisions to assure the practical usability of the proposed planning system:

- **Requirement R1 – use the available geometry data**
  For the desired areas of application, CAD-like data is available. Motion planning should be based on this kind of geometric world model. The planning system should be able to use the available data immediately and not depend on a world model that requires complex preprocessing or expansive transformations.
  The main reason for this decision is rearrangement of obstacles through the activities of the manipulator itself, and the changing loads that are to be transported. Immediate planning does not leave time to create a complete new world model.
  Typical tasks include long range motions in relatively free space as well as small, precise motions close to obstacles, especially close to start and goal positions. Therefore, the motion planning system should be able to use the supplied geometry data at its inherent resolution and precision.
- **Requirement R2 – plan independent of a specific kinematic structure**
  To be able to work with any kind of robot, the planning system should be independent from any specific kinematic structure. Of course, it may be of interest to develop high quality planning systems for a specific (family of) manipulators, but at the current state of research it is impossible to

forsee the mechanisms to come for the desired applications. Therefore, a basic system should be able to work with arbitrary physically reasonable kinematic structures.

The following objectives are set up for this work as they describe the most important features of a basic motion planning system for practical use:

- **Objective O1 – plan a collision-free motion among static obstacles**
  The fundamental task for any motion planning system is to connect a given start and goal position with a motion that does not lead to a collision of the robot, neither with the static obstacles within its workspace nor with itself. If no collision-free motion can be found, all other aspects turn irrelevant. Therefore, this is the *primary* objective.
  The restriction to static obstacles simplifies the problem, as the time a motion requires does not have to be considered by the planning system. If additional constraints are to be included, time has to be modeled and movable obstacles can be taken into account (see extention **E2** below).
- **Objective O2 – keep a specified safety distance to the obstacles wherever possible**
  To allow for modelling and control uncertainties, the planning system should be capable of keeping a safety distance to the obstacles in the workspace. This objective considers the fact, that the motion planning system itself cannot work on an exact model of the environment and the robot's dynamic behaviour – as this model does not exist. The magnitude of the required safety distance is assumed to be known and given as a task parameter to the planning system.
  The planning system may find a collision-free motion, but fail to achieve the full safety distance for the complete path (this typically happens, if start and goal are close to the obstacles). The higher level planning system has to decide if the solution is acceptable and can be performed by the robot, possibly by adapting control parameters like speed. A slower motion can typically be executed with smaller control uncertainties.
- **Objective O3 – prefer short motions**
  The planned motion should be preferably executed within short time and with little energy consumption. On the lowest level this is realized through the trajectory generation of the robot itself, but it is the planning system which is responsible to generate motions that avoid unnecessary detours.
  Together with the safety distances, short paths are considered as a kind of basic optimization that should be handled by the geometric path planning module, i.e. the planned paths should be of a high "quality".
- **Objective O4 – plan fast**
  The motion planning system developed in this work is intended for intelligent task-level systems or interactive robot (simulation) systems. To be useful within practical applications, it has to be as efficient as possible, i.e.

to find a result in the shortest possible time – or fail in a short time as well if no solution can be found.

The method to be developed should not be inherently limited to the above objectives. Therefore, two important extensions are set up that should be possible to be fulfilled by further developments. These extensions are beyond the scope of this work, but discussed (see chapter 9) due to their high relevance:

- **Extension E1 – allow interactive planning**
  The motion planning module should be extendable to a fully interactive component, allowing user interaction with the planning process in every stage. There should not only be the possibility to give *hints* to guide or ease the automatic planning (like promising intermediate positions or path segments that are fed into the motion planner as supplemental task information), but it should be possible for an operator to influence the planner on-line. This implies that the planning process is intuitively understandable and enables a meanigful visualization.
  The ability of interactive planning is as of interest as well if motion planning is part of an intelligent task-level operated robot system where topological and symbolic knowledge is handled on the higher levels that use the motion planning system.
- **Extension E2 – allow planning with task constraints**
  As discussed in the previous sections, several kinds of task constraints can be relevant. In a future general motion planning system, it has to be possible to consider them within the planning process. Their integration must not be limited, but simplified as much as possible in the context of the basic planning system.

1.3 Approach of the BB-Method

The principle idea of the BB-method is to consider an *entire path* as the subject of the planning process. Rating functions are employed to measure the quality of a path with respect to the different objectives. Planning itself becomes the *reshaping* of the path to improve it with respect to the rating function.

To plan collision-free motions, the rating function is based on the *reduction* of the size of the robot's geometry model. This allows to qualify paths that in reality move through collision. Starting with an initial path, the size of the robot is reduced until it can *virtually* move from start to goal. By repositioning intersection configurations, the path is modified incrementally to let a larger robot model pass. This is repeated until finally the robot in its real extent can move in real space. This is the naming scheme of the BB-method: ***B****low up* the robot and ***B****end* the trajectory.

By only considering local information along the current path, the planning process may get trapped in a local maximum of the rating function. To escape

such a "dead end", random subgoals are employed that decompose the planning task into subtasks and explore the full search space.

To keep desired safety distances, this scheme is continued in modified form *beyond* the real size of the robot to allow an *expanded* robot geometry model to move without collision. To achieve short trajectories for the real robot, the rating is based on the path length. Again, intersection configurations are repositioned to improve the rating and thus the "quality" of the path.

1.4 Overview of this Work

The work is structured as follows:

Chapter 2 classifies and summarizes the state of research in manipulator motion planning in general and especially in relation to the BB-method in special. The most prominent planning approaches are critically analysed with respect to the objectives of this work, and with respect to their practical usability in general. The ZZ-scheme, a combination of local planning and random global exploration, is given more attention. Within the BB-method, a simplified form is used in chapter 4 to embed the local planning of collision-free paths. It serves as a possibility to overcome "dead-ends".

Chapter 3 formally introduces the terms and objectives of this work. Geometry models, homogeneous transforms, joints, manipulator systems and all other important terms in relation to manipulator motion planning – configuration space, paths and workspace – are defined in a general way. The specific terms used for the BB-method are added, especially expanded geometry models and scaling transforms. The terms defined are used to formalize and discuss the objectives **O1**, **O2** and **O3**. Based on this, the approach is derived and the principle idea of rating functions and path modification is explained to structure the elements presented in the following chapters. Sample tasks are introduced which are used in turn to illustrate the planning schemes.

Chapter 4 is dedicated to collision-free path planning, the primary objective **O1** of this work. First of all, the rating function for colliding paths is developed. The desired properties are set up and the scaling of a single body in collision is analysed, to finally form a well-defined rating of path segments based on the reduction of the robot's size. Using this rating function, the planning scheme is developed. To improve a path, possible candidates to replace current intersection configurations are calculated and alternative paths are rated. If the rating increases, the path is modified. The candidate positions are calculated within a plane approximately workspace-orthogonal to the current path to possibly gain the highest rating increase. The mechanisms are integrated into a compact planning algorithm. Planning for the sample tasks is visualized. The complexity, the required computational effort, the topological behaviour and the influence of the heuristic parameters are examined. The planning is local and may fail. Possi-

ble conditions of failure are identified. To overcome failure situations, random subgoals are used, creating subtasks that are planned separately.

Chapter 5 handles the planning of safety distances, objective **O2** of this work. The rating based on expanded geometry models is developed analogous to the rating based on scaling. The different possible safety distances for the different links of the manipulator are considered, and a link-wise rating function is defined. Using this rating the candidate creation and the path modification are developed, extending the mechanisms employed for collision-free path planning. The planning of safety distances cannot "fail" completely, as it is based on collision-free paths. The required computational effort and the topological behaviour of the planning are discussed.

The length reduction of collision-free paths with safety distances is presented in chapter 6, addressing objective **O3**. The path is shortened, based on implicit rating: if a triangular detour can be replaced by a straight line connection without yielding collisions or reducing the achieved safety distance, an overall shorter path is found. The length reduction works local and does not find the overall shortest path. The influence of the control parameters and the behaviour of the shortening are examined and discussed.

The computational effort in local path planning is mostly spent for collision detection between geometry models. As an efficient possibility for collision detection and the rating functions using size-reduced and expanded models a novel scheme, the *swept volume collision detection*, is developed. An approximation of the spatial region that is swept by the moving robot is explicitly modeled and tested for interference. Chapter 7 presents this scheme. An algorithm to calculate intermediate positions close enough to allow the workspace-linear sweeping of slightly expanded geometry models is developed. The calculation of expanded models based on polyhedrons is described and used to create an efficient collision detection algorithm. The realization of the rating functions built upon this scheme is explained. The collision detection and the rating functions are analysed under the aspects of complexity and computational effort.

Additional results of the prototype implementation beyond the scope of the examples presented in the previous chapters are reported in chapter 8. These examples with up to several tens of joints were chosen to show the characteristics of the BB-method, and to allow a comparison with other results reported in literature. From these results, the efficiency and reliability of the BB-method becomes obvious.

Chapter 9 summarizes and discusses the results of this work. The requirements, objectives and desired extensions developed above are picked up to analyse the achieved results in general and with respect to their practical usefulness within the frame set up for a motion planning system. In addition, promising ideas for future work with the BB-method are presented.

1.5 Summary of Results

The BB-method is presumably the most powerful manipulator motion planning algorithm developed so far. It is able to handle arbitrary manipulators with any number of degrees of freedom successfully. The planning is based on the CAD-data of the robot and its surroundings, and does not require specific preprocessing or limiting preconditions.

The principle can be described as incrementally increasing the spatial volume that is available for the motion of the manipulator. The planning is incremental and monotone: it either increases the space that is available to move (by expanding the robot's geometry model) or – it fails. The planning is local, i.e. it does not consider the whole space of possible motions.

The approach is driven by practical objectives and its realization is a heuristic solution. There is a theoretical foundation, but there are also several parameters and algorithmic aspects that are developed empirically. But nonetheless – or for this reason – the BB-method is extremely effective and efficient. Its prototype implementation is able to plan motions for realistic 6-DOF industrial manipulators operating in complex environments with obstacles, in near real-time (in the order of a few seconds or below on a standard workstation), and it is also able to plan motions for redundant and hyperredundant manipulators in the order of seconds or minutes. The effective algorithmic complexity of collision-free path planning is linear in the number of joints. Experiments show that the computational effort spent for planning is mainly dependent on the complexity of the geometric modelling of the robot and its environment.

The quality of the resulting motions is very high. Although no well defined measure of motion quality is employed (as it is difficult to define at all), the planned motions are *straight* and *short* – thus usually fast and cheap to be moved along. They keep a safety distance to obstacles wherever possible – thus allowing to compensate for modelling and control uncertainties.

The BB-method can be seen as a base layer for advanced intelligent motion planning systems. Its results can be passed to sophisticated optimization and motion execution layers, and it can be used by higher level intelligent robotic systems. It can easily be extended to fulfil advanced requirements – namely planning under task constraints and the utilization within an interactive planning system that allows operator interaction at every stage. The performance of the prototype implementation proves that all these possibilities are within reach.

Chapter 2

State of Research

The state of research in motion planning is classified and summarized in general, with special consideration of the BB-method. The focus is set on collision-free path planning for manipulators, the primary objective of this work. The most important published planning approaches are critically analysed with respect to their practical usability. Similarities and differences to the BB-method are outlined. The ZZ-scheme, a combination of local planning and random global exploration, is given more attention, as it is employed to embed the BB-method as a possibility to overcome "dead-ends" of local planning.

2.1 Introduction

The planning scenery considered can be summarized as follows: A manipulator, consisting of n rigid bodies connected by an equal number of joints moves in a space of static obstacles, called the robot's workspace. The possible postures of the manipulator are described by the joint variables. All joint variables form an n-dimensional vector, called configuration. Within the configuration, the joint values are labeled upwards, i.e. the first component describes the base joint, the second component the following joint and so on. All possible configurations span the configuration space, short c-space. The planning task is described by two specified configurations, start and goal. Given the robot's kinematics and the environmental obstacles, an implicit description of the collision-free and colliding regions of the c-space is available. A possible solution is a collision-free path connecting start and goal. A formal definition of these terms as they are used within the description of the BB-method is given in chapter 3.

An enormous spread of publications with regard to robot and manipulator motion planning has been published in the past decades, with ever increasing interest in this subject. Nonetheless, the overall problem is far from being solved. With the increasing capabilities of the controlling and planning instances, the complexity of the manipulators has also increased, and of course the demands have shifted as new sensor technologies and new areas of application come to reach. The best surveys are almost outdated (Latombe (1991) and Hwang and Ahuja (1992)), a more recent one is announced for the near future (Summer 1998) by Gupta and del Pobil.

This review concentrates purely on the manipulator motion planning problem in known environments and with more than just two or three joints. It does not consider the general motion planning problem with all its aspects and diversifications (mobile robots, multiple robots, moving obstacles, assembly and grasp planning). But some theoretical results that hold for manipulators as well as for the general motion planning domain are of interest. They show the enormous inherent complexity of motion planning, and this influences the possible approaches, especially for manipulators with many joints.

2.2 Theoretical Results

In Reif (1979), a lower bound for planning free paths in a configuration space of arbitrary dimension is established. It is summarized by Latombe as follows (Latombe (1991), p. 34): *Planning a free path for a robot made of an arbitrary number of polyhedral bodies connected together at some joint vertices, among a finite set of polyhedral obstacles, between any two given configurations, is a* PSPACE-*hard problem.*

The consequences of this result become obvious if the meaning of the terms *PSPACE*-hard and *PSPACE*-complete is briefly reviewed (on the general terms and concepts of computational complexity, see e.g. Aho et al. (1983)). The complexity of a problem is measured by the time and space consumption in relation to the input size that an idealized computer (Turing Machine) requires if running an idealized algorithm.

Classes of problems are differentiated in a hierarchy of complexity. The classes of interest in this context are P and *PSPACE*. P denotes the class of problems that can be solved in polynomial time with respect to the input size. *PSPACE* is the class of problems were the processing space requirements are polynomial with respect to the input size. It is known that there are problems in *PSPACE* that require more than polynomial time, i.e. $P \subset PSPACE$. The best known algorithms for problems of the class *PSPACE* run in exponential time. Up to now it is unknown whether there are faster algorithms, but their running time is certain to increase faster than any polynomial. A problem is called hard with respect to a class if it is as difficult to solve as all other problems of this class, but it does not necessarily belong to this class. A problem is called complete if it is hard and belongs to the respective class.

The cited result proven by Reif indicates that there is no possible polynomial algorithm able to generally solve the motion planning problem.

An upper bound on path planning complexity was first established 1983 by Schwartz and Sharir (see in Schwartz et al. (1987)). It can by summarized as follows: *Planning a free path in a configuration space with n dimensions has a time complexity that is exponential in n and polynomial in the size of the description of the geometry models.* This result was drawn from a planning algorithm that is twice exponential in n. Canny (1988) presented an algorithm that is just singly

exponential in n. Both algorithms are based on a reduction of the planning problem to a decision problem in the first order theory of reals (see e.g. Hwang and Ahuja (1992) for a summary). But as there are no existing implementations of decision algorithms that are fast enough, these methods cannot be used in practical cases and are of pure theoretical interest (Latombe (1991), p. 35).

These results suggest that the complexity of manipulator motion planning is increasing exponentially in the number of degrees of freedom n. There may be faster algorithms, but they are certain to be more complex than polynomial, and their existence is doubtful. This enormous complexity hinders complete planning for manipulators with more than a few joints (more than four to six, depending on the complexity of the geometry description). This means, within the full resolution and dimension of the problem description, it is impossible to definitely find a solution if one exists. And it is equally impossible to decide that no solution exists if none can be found. Therefore, all planning approaches for manipulators are inherently incomplete and attempt to solve the planning task either heuristically or using a certain discretization.

2.3 Attempting a Classification

The complete planning problem is intractable for realistic manipulators with more than a few joints. To be able to plan successfully anyway, i.e. at least solve most tasks with reasonable effort, a lot of very different planning approaches were developed. A classification is very difficult, but two major aspects can be differentiated, namely global and local planning. Within the most approaches, both aspects are considered and very often intermix.

Global Planning

Within this context, *global* means global within the c-space. To be able to find a solution, the whole c-space has to be explored – approximately, as this is too complex to be done completely. There are two major distinctions possible to classify global planning:

- **Building a representation of the c-space**
  An approximate representation – or at least a representation of the collision-free regions – is constructed *a priori* to the planning itself. Planning for a given task is in turn reduced to connect start and goal to the representing *c-space map*, and to plan a path within this representation.
- **Opportunistic planning in the c-space**
  Opportunistic planning means that a c-space representation is constructed incrementally while attempting to connect start and goal. If the planning task is "easy", only small parts of the representation may be required to solve the task.

To capture the incapable high dimensionality of the c-space random schemes are often employed, either to build up a representation or in an opportunistic fashion. They promise to explore a high dimensional space with limited effort and with a scalable probability of success.

Local Planning

Almost any global planning scheme requires a local component as well. Within this context, *local* means to plan based on configuration-dependent information only, without considering the whole c-space. Typically, local planning delivers partial information required by the global planning. It avoids the high complexity of the full c-space but may be very powerful on its own, depending on the employed methods. Local planning may fail, the failure conditions of local planning are referred to as *local minima*.

Based on these considerations, the approaches presented below were roughly put in order. A quite similar classification scheme was presented by Latombe (1993) for the context of general motion planning.

2.4 Resolution Complete Global Planning

The most attractive property of global planning is its inherent *completeness* – a possible solution is found, and failure indicates that no solution exists. The major disadvantage is its extreme complexity. Complete planning is well suited for mobile robots with 3 DOF (translation and rotation in a plane). One possible solution for manipulators with more degrees of freedom is to create a representation based on a fixed resolution discretization. Planning schemes like this can find a solution or indicate failure down to the level of this fixed resolution, therefore these algorithms are called *resolution complete*.

Approaches of this kind are often called "approximate cell decomposition", and there are numerous publications based upon them, see e.g. Lozano-Pérez (1986), Fink and Wend (1991), Zhang and Münch (1993), Zhang (1994). A full map of the c-space is constructed based on "cells" of a fixed size. Each joint range is partitioned in fixed intervals. The resulting grid of potential nodes is evaluated in the workspace, i.e. the individual configurations at all grid positions are tested for collision. Care has to be taken to assure that 1-neighbouring nodes can be connected by a motion in exactly one joint (or higher order neighbours are considered accordingly). A possibility to do so is to test isotropically expanded geometry models that assure a collision-free motion. Another possibility is to apply distance computation between the robot and the obstacles, and to use a conservative estimation of the resulting free region in the configuration space to decide about the connectivity between neighbouring nodes.

When a manipulator collides with the first link, any movement in any upper joint cannot remove the collision, as these joints do not influence the position

of the first link. Similar conditions are given for any colliding link unless it is the last one. This can be used to reduce the computational effort of the grid evaluation enormously, as the state of nodes within complete subspaces of the c-space becomes known upon the detection of a collision in a lower link.

The construction of the approximative map with its set of collision-free nodes and the respective adjacency information is a preprocessing step. Global planning itself turns into graph searching, and different schemes can be employed. Note that the grid is of considerable size and may contain "local minima" with respect to the goal distance. Its adjacency is n-dimensional and it is non-trivial to search a connection. In Lozano-Pérez (1986), rectangloid intervals of collision-free nodes are grouped into larger cells, this reduces the graph complexity. To find a path, A*-algorithms were employed.

Another way to evaluate the calculated grid is presented in Ralli and Hirzinger (1994, 1996). Within the n-dimensional grid, a wavefront is propagated from the goal configuration, flooding the complete c-space with approximate distance information. The path can now easily be found by descending from one grid node to an adjacent closer node repeatedly.

Warren (1989, 1990) developed another interesting method. The c-space map is constructed completely, i.e. collision-free, and colliding nodes are represented. The next phase of preprocessing propagates the distance from free space *into* the obstacles. Planning is very different to the above mentioned schemes. A complete path, i.e. a sequence of nodes connecting start and goal more or less in a linear way, is evaluated by checking the *degree of intersection* with the c-space obstacles. Based on local decisions, the path is modified to pass through *better* regions of the c-space. This bears a resemblence with the BB-method, as the path as a whole is the subject of planning, but unlike the approach presented in this work, the rating and modification employed by Warren is based on complete *a priori* c-space evaluation.

The major disadvantage of all above approaches is the enormous computational cost of preprocessing. The required number of collision detection tests and the resulting number of collision free grid nodes that have to be stored is huge, i.e. the effort in time and the required memory is extremely high. Lozano could not plan for more than three joints, Warren even limited his work to two joints, and Ralli planned for a 6-joint robot, but neglected the influence of the last joint and used a coarse resolution. This marks a technical upper limit: even with increasing computer power, it will not be feasable to represent a complete c-space with 6 or more dimensions with an acceptable resolution.

Planning becomes reasonably fast (order of seconds) once the representation is built (order of hours). Within the areas of application targeted within this work, this is not suitable as the space may change between each task. Regarding the objectives, the resulting paths will typically be quite short in c-space, but will not keep clear of the obstacles. Further processing to improve the quality of the resulting path using the c-space representation is not obvious and not reported.

2.5 Random Global Planning

A randomly built representation can be used instead of a representation based on fixed discretization. The invested effort can be limited instead of exploding with the number of joints. The completeness with respect to the resolution gets lost, but the *probability* to create a sufficient c-space representation can be increased by increasing the invested effort. Therefore, the random-based algorithms are called *probabilistic complete*.

Random-based c-space maps have become very popular in recent years, see e.g. Eldracher (1994, 1995), Kavraki and Latombe (1994), Kavraki et al. (1994), Amato (1996), Horsch et al. (1994). The basic scheme is quite similar in these works, the following description follows Overmars and Svestka (1994). The preprocessing begins with a certain number (order of thousands) of random collision-free configurations created in the c-space. For each of these nodes, a certain small number (order of 30) out of all other nodes near by in the c-space is selected as *possible neighbours*. The connections between each node and its possible neighbours are planned using the local planner. This is just a c-space straight line collision detection, the most simple possible local planner. Creating edges for each successful local planning results in a graph that covers the c-space.

But typically the quality of this graph is not sufficient. It consists of several unconnected components and requires improvement. Based on their connectivity, the nodes are rated, and calculating a probability using this rating makes it possible to select candidates randomly among the nodes that are attempted to be expanded. New nodes are created in the vicinity and attempted to be connected to the current node and to other graph components. The effort spent in the node expansion is limited by preset parameters; either time to spend or number of nodes to be tested.

As a result, a random graph spans the c-space that *hopefully* covers every region sufficiently without falling apart into unsuitable components. The planning itself, given a task, is reduced to connecting start and goal to the graph with local planning and applying graph search. The latter is done with standard graph search technologies, the former uses distance-based heuristics to select nodes and components that are used for the local connection of start and goal respectively. If the subgoal graph is "good enough", planning is very efficient (down to fractions of a second), as only very few and short local plannings are required, because all path segments drawn from the graph do not need to be replanned.

The random subgoal graphs can be applied to robots with arbitrary numbers of joints. But they require a preprocessing to create the map that can hardly be adapted to modified environments or loads. The computation times reported for preprocessing (order of seconds to minutes) were typically achieved with extremely simplified robots (line segments) among only a few polyhedral obstacles. It cannot be expected that the preprocessing is effective for more complex sceneries. It is also reasonably doubtful that preprocessing is successful at all in *complex*

sceneries. The random exploration scheme tends to create large components of similar configurations in free space. Even with the node expansion, it may fail to connect especially those regions that later on are the ones of interest. If a manipulator has to move through a tight opening, unconnected graph components may be constructed on either side. To overcome this, the only possibility would be to increase the preprocessing effort.

The resulting paths can be expected to be quite short in c-space if the graph search honours the c-space distance between nodes. The distance to the obstacles is typically not considered, and the c-space graph is not suited to improve a found solution.

2.6 Opportunistic Resolution Complete Global Planning

One of the main problems with global planning based on a complete c-space representation is the required preprocessing before any planning can take place. If the construction itself is executed as a part of the planning process, and is based on a fixed resolution discretization of the c-space, the whole planning process is still *resolution complete*. And if the construction of the representation can furthermore be limited to a certain "region" that contains the solution, this appears to be a promising approach to motion planning.

Hierarchical Decomposition of the C-Space

One possibility to locally construct an inherently global representation is to use a hierarchical approach: If, at a certain resolution, the quality of the representation does not suffice to find a solution in a certain region, refine the resolution and focus the attention on the most promising partial region.

There are numerous approaches built upon this basic scheme of *divide and conquer*, see e.g. Bodduluri (1990), Duelen and Willnow (1991), Pobil et al. (1992), Pobil and Serna (1995). The best known planner among these is called SANDROS (Chen and Hwang (1992, 1996)) and is now explained in more detail. SANDROS employs a hierarchy of refinement of intersection regions. At the lowest level, an intersection region is unspecified in *any* joint, thus it spans the whole c-space. The level of refinement counts the number of joints that are specified upwards from the base, all remaining joints are unspecified at that level, and any configuration that matches the lower joints is called belonging to that region. Thus, a specific configuration is a fully refined intersection region.

Global planning starts off with a graph containing start and goal configurations only. Local planning does then attempt to connect the vertices of this graph. SANDROS' local planner tries to minimize the c-space distance to the goal and the cartesian distance to the obstacles, using heuristic simplifications to minimize the computational effort. If the local planner is successful, the task

is solved, but as this planner is a quite simple one, it only solves very simple problems in a purely local manner, and it often gets stuck in local minima.

If local planning fails, intersection regions are created with only the first joint fixed. They are spread heuristically along the first axis, their number is in the order of 10. Their positions are fixed using an implicit grid devised by a discretizing stepsize for each joint axis. They are stored in the graph together with start and goal. Edges are created, based on the adjacency in the c-space. The global component plans a connection in the graph via these intersection regions, and calls on the local planning to connect positions within the regions that were successfully reached from either start or goal. To clarify this: the local planning attempts to move the entire robot into an intersection region, and if it successfully reaches a position within this region, i.e. all currently specified joint positions are reached, local planning is successful.

Local planning deals with full configurations only, the global planner uses the intersection regions to control the attention of the local planner. If local planning fails, either new intersection regions are created at a higher refinement level, or other possible paths between the current regions are explored, or the whole process involves backtracking. The action taken by the global planner is based on a heuristic evaluation of the current search graph until possibly a complete solution can be found. In the worst case, SANDROS includes every possible configuration on the discretization grid as nodes in its search graph.

Summarizing, SANDROS is a complex, heuristic scheme, that is resolution complete, but nonetheless applicable to robots with more than just four or five joints. The published results are in the order of minutes or more for robots with nine joints. The created paths are unpredictable long, although the heuristics attempt to implicitly prefer short paths. They might keep a certain distance to the obstacles partially, as the local planner attempts to keep it. But this is only a result of the heuristics that minimize the probability of collision, not a controlled feature.

Implicit C-Space Grids

The hierarchical approaches are complete because they are based on an underlying discretizing c-space grid that is reached at the lowest level of decomposition. Alternatively, one can directly plan on the level of this grid. It can be directly related to the complete c-space maps of section 2.4, where graph searching was applied to the cells. In this case, graph searching is applied to the grid using full adjacency, not knowing the contents of the unexplored nodes. This is usually done using A*-related schemes, see e.g. Warren (1993), Wurll et al. (1997), Wörn et al. (1998).

Beginning at the start node, nodes are expanded that are closer to the goal, i.e. they are checked for collision and accepted if they are collsion free. If a collision occurs, alternative nodes in the neighbourhood are expanded. Planning

terminates successfully if the goal is reached. Possible local minima of direct search are subsequently flooded with expanded nodes and thus escaped.

The opportunistic global planning is resolution complete. But if the number of joints and the task difficulty increases, the possibly required memory explodes. Therefore, the "completeness" can be regarded as a purely theoretical attribute for more then six or seven joints. And this is the major drawback: the computational load, although it increases "proportional to task difficulty" (Chen and Hwang (1992)), may increase beyond acceptable bounds. The applicability to robots with a lot of joints is therefore doubtful, unless heuristics are involved that remove the completeness for the benefit of efficiency.

2.7 Opportunistic Random Global Planning

To avoid growing computational load, random techniques can also be used in an opportunistic fashion. This allows to limit the computational load, as the random components are controlled heuristically and scale the probability of success as well as they scale the invested effort. Two very different planning approaches based on these considerations are presented.

RPP - Randomized Path Planner

The Randomized Path Planner was introduced in 1990 and experiments were performed by several authors, see e.g. Barraquand and Latombe (1990), Latombe (1991), Barraquand et al. (1992), Challou et al. (1995), Chang (1996). RPP uses two different local planning components. The first local planner uses a potential field approach. It is based on a discretized map of the workspace, typically equal sized voxels in the order of 100^3. Within these voxels, the distances to the obstacles are propagated using a wavefront algorithm. This creates a spatial map which is capable to rate the distance of single points in the workspace to the obstacles. The robot itself has a selected set of control points heuristically assigned to its links. The number of the control points is in the order of the number of the links. They are chosen in a way that a pose is uniquely described by the position of the control points. The assignment of the control points and the creation of the discretized workspace are required preprocessing steps.

For a given task, numerical (i.e. discretized) potentials are independently created for each control point. A copy of the workspace voxel distance map is used to propagate the distance from the goal position of the respective control point, combining it with the obstacle distance information. This results in a set of minima-free numerical potentials for single moving points. Now the whole robot has to be moved. Beginning at the start configuration, a c-space potential is evaluated in a local heuristic vicinity by testing all the control points within their individual workspace potential and adding the resulting values. If any collision-free steps improve the c-space potential compared to the value at the current

configuration, the best one is chosen and the robot is moved. This is repeated until either the goal is reached or the planning gets stuck in a local minimum. The latter may happen easily, although the workspace potentials for the control points were created minima-free.

To escape a local minimum, the global planner invokes the second local planning component. Random "Brownian" motions are executed, parameterized by the length interval for random distance calculation and using arbitrary random directions in the c-space. At every configuration reached, the c-space potential value is calculated and compared to the local minimum. This kind of random planning stops either if a better potential value is found and the descent towards the goal can be continued, or if the global planner decides heuristically that this dead-end cannot be escaped. In the latter case, heuristic backtracking switches back to a selected configuration prior to the minimum and initiates a local search in another direction. If the goal cannot be reached in a certain time, the global planner has to decide heuristically to give up the planning and declare failure.

The resulting paths are very zig-zag-shaped, especially when their planning involved random motion. A local shortening of the path removes unnecessary nodes. The inner node out of a sequence of three nodes is taken away if the linear connection of the outer nodes does not yield collision. The distance to the obstacles is not explicitly considered for the whole robot. It is only implicitly and partially honoured through the workspace-potentials used for the control points.

Simple tasks are solved fast and easily, but if the minima of the potential are "deep", the random scheme may need arbitrarily long to escape it. A "deep" minimum is given if the control point could move directly to the goal – if it is not attached to the robot in a way that a long detour for the control point in the workspace is required. An example for such a situation is the stretching of an arm on either side of a column. The motion requires a complete contraction of the arm, but the control point potential does not help very much to find this solution, and the random motions are not promising either. In summary, RPP tries to use workspace information in its local planning. But the generation of this workspace information requires a *task-dependent* preprocessing, and it may contain local minima that are expensive to escape, especially for robots with a large number of joints in a cluttered environment.

Random Subgoals - The *ZZ*-Method

The ZZ-method was introduced in 1992 and is still further developed, see Glavina (1990, 1991), Kugelmann (1994), Braun and Corsépius (1996), Baginski (1996, 1998). It is somehow comparable to the subgoal graph methods (see 2.5 above), but with one major difference: The graph is not stored between tasks, but created anew for each task. That makes it extremely suitable for the desired applications of this work. It is impossible to place hundreds or thousands subgoals in the c-space to solve one task. To avoit this, the local planning of the ZZ-scheme is

quite powerful on its own. The ZZ-method works without any preprocessing.

The local planning attempts to move on the linear connection in c-space from start to goal. If a collision occurs, sideward steps orthogonal to the current direction are calculated and the first one that is accessible without collision is chosen to restart the linear motion towards the goal. This roughly approximates a "sliding" along the obstacle's surface, therefore these collision avoiding steps are called slidesteps. This planning method fails in a local minimum if no slidestep can be executed that yields a configuration closer to the goal.

The global planning creates random subgoals and attempts to connect start and goal via these subgoals using the local planner. Thus a small subgoal graph is constructed until start and goal are in the same component. Connections with few edges in the graph are preferred, i.e. all possible connections with one subgoal are tested first, then the possible connections using two subgoals and so on.

The number of random nodes and the number of edges within a possible solution path is limited. Global planning terminates without success if these limits are reached. In case of a successful planning, the resulting paths are zig-zag-shaped close to obstacles and may contain subgoal-detours. To reduce execution time and to avoid sharp turns, the paths are locally straightened and shortened by repeatedly cutting off triangular detours based on heuristic evaluations (Berchtold and Glavina (1994)). The distance to the obstacles is not considered.

The ZZ-method with slidesteps is very efficient for tasks that can be solved purely locally or with only one out of a few random subgoals. Planning times are in the order of seconds. If the number of joints gets larger and the local planning fails substantially due to dense passages that cannot be passed easily with slidesteps, the planning gets inefficient. Solutions that inherently require a suitable *sequence* of subgoals are rarely planned, as the probability to randomly find such a sequence in the n-dimensional c-space is very small. In cases like this, the effort increases to minutes or even far beyond, if a solution can be found at all.

Based on these results, the BB-method is developed. An incomplete local planner is extended by random global exploration based on subgoals. The use of random subgoals is mainly efficient for tasks that can be solved with only one of them. To be able to solve even very complex tasks the local planner of the BB-method is designed to be very powerful, being able to solve most of all tasks. Opposed to the ZZ-scheme, a subgoal graph is neglected, tasks that cannot be solved locally are attempted to be solved via one subgoal out of a limited number of random configurations.

2.8 Heuristic Planning

The following is a brief overview of other motion planning approaches that do not fit into the classification. They are all heuristic in nature, and therefore

incomplete. But this does not necessarily disqualify them: all the previously discussed approaches are practically incomplete, either due to limited resolution, due to limiting heuristics or due to practically limited computational resources. The approaches were selected due to their importance in literature or due to their relation to the BB-method.

Applying Forces in the Workspace

A very interesting approach in planning motions for robots with many joints is presented in McLean (1994), McLean and Cameron (1996). Within the workspace a so-called skeleton of continuous line segments is created in a preprocessing phase, using a propagation scheme quite similar to the one used within RPP. The skeleton approximates the voronoi manifolds of equal distance to all obstacles. For a given task, a *virtual span* is attached to the robot's tip point at the start configuration and connected to the closest point in the workspace skeleton. This span is now used to "pull" the robot to the goal configuration, propagating the applied forces into suitable joint movements. The reported results are in the low order of minutes (not considering the preprocessing) for robots with up to 15 joints moving through relatively small passages. According to the authors, the method is preferably suited for "manipulators whose links are thin compared to their length" (McLean and Cameron (1996)).

Related planning schemes are reported e.g. in Reznik and Lumelsky (1995) and Overgaard et al. (1994). In these works no supporting skeleton structure exists, the robot is just "pulled at its tip" towards the goal. This implies the risk of getting stuck in local minima, and the required search mechanisms are usually quite limited in their efficiency.

Joint-Sequential Planning

An incomplete planner that plans sequentially for the joints of a manipulator is presented in Gupta and Zhu (1994), Gupta and Guo (1995). For each link, beginning at the base, a path is planned by constructing a partial two-dimensional c-space map with one axis being the joint variable of the current joint, the second axis being the formal time implied by the motions of the previous joints. This reduces the complexity from n dimensions down to n two-dimensional planning problems, and thus this kind of planning is applicable for more than just a few joints. But if the planning fails for one link, backtracking is required and "virtual obstacles" are placed in the c-space maps of the previous joints. With this backtracking mechanism, the planning efficiency may turn very low, as complete two-dimensional "slices" have to be recalculated again and again. The reported computation times are in the order of minutes for 6-joint manipulators.

Heuristic Deformation of Paths

A planning scheme that deforms complete paths is presented in Dupont and Derby (1988). Based on a very coarse implicit c-space grid, an initial path is created as a sequence of grid configurations. If collisions are detected along this path, it is locally altered by replacing grid configurations. The direction of these modification steps is determined by preset strategy directions, i.e. rules like "if link 3 collides, turn joint 2 in positive direction" are employed. This limits the usability: strategies have to be developed for different robots and different regions of the c-space or the workspace. It may be almost impossible to define a promising strategy direction for robots with many joints. The idea, that is used within the BB-method as well, is to modify complete colliding paths to find a collision-free solution. The application presented by Dupont and Derby (1988) was limited to simple environments with up to 4-joint robots.

Using the Insertion Depth

The principle of complete path modification was used in combination with a certain "measure of collision depth" by other authors as well. In Buckley and Leifer (1985), Buckley (1989), the path of an object moving on a plane was modified according to the *minimum translational distance*. This distance is defined as the length of the shortest possible translational movement of an object in collision to get out of the collision. These and related measures are developed and examined e.g. in Gilbert and Ong (1994), Ong (1995), Cameron (1997), but their use for manipulator motion planning is almost unknown. Dai (1989) "pushes" single configurations out of collision by using a set of a few spheres as an approximative model, but only in the case of "slight" collisions, i.e. when the intersection depth is small.

In Ong and Gilbert (1994), the "penetration growth distance" is employed. It is defined for two intersecting convex objects as the largest scaling factor applied to both objects simultaneously, which results in a "touch" of the two objects. The planning scheme is based on a path modeled as a cubic B-spline with a fixed number of interpolation configurations. The entire path is evaluated by integrating the penetration growth distance of all links with respect to all colliding obstacles along the path, and modifications are applied to the interpolation configurations using methods of linear programming. Ong and Gilbert (1994) were able to plan for manipulators with up to 6 joints in very simplified environments in the low order of minutes. The application to robots with more joints in more cluttered environments appears doubtful and is not reported. The number of control parameters grows rapidly, and the penetration growth distance that is calculated for *all* robot links *simultaneously* does not result in an adequate measure of collision for the robot as a whole. Consider the same example as discussed in conjunction with the local planning scheme of the Randomized Path Planner: A stretched manipulator arm shall pass a column and has to retract itself to do so. The

penetration growth distance will not indicate any collision for the foremost links, if they can move to the goal position without collision. Thus, the mechanism inherently contains numerous local minima and is therefore unsuitable to modify paths successfully to avoid collision for the entire arm.

The BB-method is based upon a comparable scheme: modify paths that lead through collision. But the developed rating is suited for robots with an arbitrary number of joints, as not all the links individually, but the *entire robot* is considered for the rating.

2.9 Summary

Up to now, none of the developed planning schemes has made its way into a commercial product that is used in practical applications. They are all either too complex or too limited in their abilities. Planning schemes applied to real robots in experimental setups are also limited, either in the number of joints (e.g. only three joints were considered in Cheung and Lumelsky (1990)) or by decomposing the planning into low-dimensional planning for several groups of joints (e.g. three lower joints for long range motion and three separately planned joints for the wrist motion in Tarokh (1996), Tarokh and Hourtash (1997)).

Regarding objectives **O2** and **O3**– keeping safety distances and planning short paths – most of the planning schemes do not deliver satisfying results. The published works either completely neglect these aspects or simply pass over them. The BB-method attempts to integrate these additional aspects on the very low level of geometric path planning.

CHAPTER 3

Terms, Objectives, and Approach

*Within this chapter, the objectives **O1**, **O2** and **O3** are formally developed and the approach of the BB-method is presented.*

First, the basic objects of planning, the geometry models, are introduced. Then means are supplied to position them, the homogeneous transforms. These tools are in turn used to define joints, kinematic chains, and manipulator systems. For the latter, the respective work- and configuration spaces are defined. The notation used throughout this work is inspired by Latombe (1991) and Yoshikawa (1990).

The objectives are formally stated and discussed. The chosen approach is presented to constitute a structure for the following chapters. Three sample tasks for different robots are introduced that are used to illustrate the planning schemes.

3.1 Basic Terms

3.1.1 Geometry Models

Planning in general takes place *before* any action is executed in the *real world*. Thus, planning uses a *model* of the reality, a *virtual* approximation – usually a simplification – of what is the subject of planning. In the case of robot manipulators moving among obstacles, the basic entities are rigid bodies – they make up the environment, and they make up the robots.

For this reason, **geometry models** are required to represent the rigid bodies of the real world. The geometry model of an object o will be denoted as G_o. To gain a geometry model, a referencing *coordinate frame* has to be defined fixed to the object o. All objects are assumed to be objects of the three-dimensional euclidean space, denoted as $\mathbb{R}^3$. A cartesian coordinate frame in $\mathbb{R}^3$ consists of an origin and three orthogonal axes with unit length (units might be e.g. meters and are of course assumed to be used consistently throughout the whole modelling process). If this frame is set up for an object, each single point within o and on its surface has an exact, measurable position.

G_o is a description of o with respect to the local frame of o. There are several possibilities to make up feasable descriptions with any desired precision of approximation. Typical techniques are:

- **Boundary representation**
  G_o is made up of a description of the surface of o. The method generally

used is to describe a polyhedron, requiring vertices and structural information. This structural information describes which sequences of coplanar vertices span *facets*. There are advanced possibilities to model surfaces, e.g. spline patches, that are used in CAD-systems for example. Surfaces like this can easily be approximated by facets at any level of precision.

- **CSG - constructive solid geometry**
  In this case, G_o is made up by boolean operations on scaled primitive objects. Shapes like cubes, spheres and cylinders are intersected, united etc. This type of models explicitly includes the volume they describe. On the user-interface level, modern CAD-programs prefer this modelling to boundary representations, as construction changes can be easier propagated. For visualization and other purposes, these models are generally (and quite easily) transformed into boundary representations. The volume taken by o can be approximated

For path planning, the use of polyhedral models is sufficient. The only frame-relative position information within these models are point vectors $\mathbf{u} = (x, y, z)^T$ describing the positions of vertices. The structural information about the arrangement of edges and the orientation of facets is independent on position.

Bounding Models

A desirable property of a geometry model is to *bound* the real object o, e.g. all points of o are included in G_o. This is especially required for motion planning, as the absence of collision for a **bounding geometry model** assures the absence of collision for the real object. Such *slightly* bounding geometry models are assumed throughout this work. To put it formally: a geometry model G_o is called *bounding* (G_o bounds o) if

$$\{\boldsymbol{w} \mid \boldsymbol{w} \in o\} \subseteq G_o$$

where the inclusion operation is used for all points within the projection of the volume described by the model G_o and the "real" points within o.

Expanded Geometry Models

A bounding geometry model includes o completely, but will usually be as tight enclosing as possible. To assure a certain distance between models, **expanded geometry models** are introduced. With $\| x \|$ as the cartesian *length* (2-norm), a point $\mathbf{u}$ is defined expanded by d, denoted $\mathbf{u}^{+d}$, as the sphere

$$\boldsymbol{w}^{+d} = \{\mathbf{u} \mid \| \boldsymbol{w} - \mathbf{u} \| \leq d\}$$

Expanded geometry models G_o^{+d} are models that bound the object, if all points of the object are extended by d:

$$\bigcup_{\boldsymbol{w} \in o} \boldsymbol{w}^{+d} \subseteq G_o^{+d}$$

This isotropical expansion implies that the position of o can be varied in any direction by a maximum distance of d without leaving the spatial region modeled by G_o^{+d}.

3.1.2 Homogeneous Transforms

Geometry models are descriptions with respect to its local frame, a cartesian unit base fixed at an arbitrary position with arbitrary orientation. To set up several models in a spatial relation, a relation between the object frames has to be established. For this purpose, **homogeneous transforms** are employed. They are briefly introduced here, details can be found e.g. in Craig (1986). A coordinate frame $\mathbf{T}$ described within a given base consists of an origin vector $\mathbf{t} = (x_t, y_t, z_t)^T$ and a base $\mathbf{R}$, made up of three orthonormal and right-handed base vectors $(\mathbf{x}_\mathbf{R}, \mathbf{y}_\mathbf{R}, \mathbf{z}_\mathbf{R})$:

$$\mathbf{T} \;=\; \{\mathbf{R}, \mathbf{t}\}$$

A point $\mathbf{u} = (x, y, z)^T$ with its position described in coordinates relative to the frame $\mathbf{T}$ is transformed into coordinates of the base that was used to describe $\mathbf{T}$ by multiplying its coordinates with the base vectors of $\mathbf{R}$, summing up the results, and adding the origin $\mathbf{t}$:

$$\mathbf{u}^* \;=\; x * \mathbf{x}_\mathbf{R} + y * \mathbf{y}_\mathbf{R} + z * \mathbf{z}_\mathbf{R} + \mathbf{t}$$

If all components of $\mathbf{T}$ are put into a *homogeneous* 4×4 *matrix* this operation is formally simplified very much:

$$\mathbf{T} \;=\; \left(\begin{array}{ccc|c} \mathbf{x}_\mathbf{R} & \mathbf{y}_\mathbf{R} & \mathbf{z}_\mathbf{R} & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \;=\; \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

$\mathbf{u}^*$ can now be calculated with one matrix multiplication, if $\mathbf{u}$ is augmented by a 1 as fourth component, that is removed from $\mathbf{u}^*$ after the multiplication, shortly written:

$$\mathbf{u}^* \;=\; \mathbf{T}\,\mathbf{u}$$

(Note: In the general case of homogeneous matrix/vector multiplication, the resulting four-dimensional vector does not have a 1 as fourth component. A mapping into the 3-dimensional space is achieved by dividing the first three components by the fourth component.)

Inverse Transforms

Homogeneous transforms can be inverted. If the point $\mathbf{u}^*$ is described within the same coordinate frame as the transform $\mathbf{T}$, $\mathbf{u}$ can be calculated, with coordinates in the frame $\mathbf{T}$:

$$\mathbf{u} \;=\; \mathbf{T}^{-1}\,\mathbf{u}^*$$

The structure of a homogeneous transform $\mathbf{T}$ simplifies the calculation of an inverse very much, as the orthonormal component $\mathbf{R}$ is inverted by transposition, and the origin offset changes its description coordinates and its sign:

$$\mathbf{T}^{-1} = \left(\begin{array}{ccc|c} & \mathbf{R}^T & & -\mathbf{R}^T\mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

Placing Geometry Models

Homogeneous transforms use the relative coordinate frame itself as the operator. If the components of $\mathbf{T}$ are set up in a frame belonging to an object o and $\mathbf{T}$ is intended to describe the relative position of the object v, it will be denoted as ${}_o\mathbf{T}_v$. If applied to the geometry model G_v (i.e. if its local frame-relative positions are transformed by multiplication with ${}_o\mathbf{T}_v$), the object is placed at the desired position within the frame of o:

$$\text{Place } G_v \text{ at the relative position } {}_o\mathbf{T}_v\text{:} \quad {}_o\mathbf{T}_v\, G_v$$

Homogeneous transforms can be multiplied and the result is again a homogeneous transform. If the coordinates of object v shall be calculated within the frame of an object r, and the relative transform ${}_r\mathbf{T}_o$ is known, this is done by

$$\text{Place } G_v \text{ at the relative position } {}_o\mathbf{T}_v\text{, relative within } {}_r\mathbf{T}_o\text{:} \quad {}_r\mathbf{T}_o\, {}_o\mathbf{T}_v\, G_v$$

Thus, the complete transform ${}_r\mathbf{T}_v$ is the matrix product of ${}_r\mathbf{T}_o$ and ${}_o\mathbf{T}_v$:

$${}_r\mathbf{T}_v = {}_r\mathbf{T}_o\, {}_o\mathbf{T}_v$$

Scaling Transforms

A modified transform is used within the BB-method. Usually $\mathbf{R}$ is orthonormal, i.e. its rows and columns have unit length. If this matrix is multiplied with a scalar factor s, the resulting transform changes the *size* of the geometry model it is applied to (*without* shifting the origin, as $\mathbf{t}$ is untouched). If all components of $\mathbf{R}$ are multiplied with s – i.e. the length of the base vectors within $\mathbf{R}$ are set to s – a transform is denoted ${}^s\mathbf{T}$:

$${}^s\mathbf{T} = \left(\begin{array}{ccc|c} & s\,\mathbf{R} & & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

Elementary Parameterized Transforms

To be able to model joints, homogeneous transforms have to be parameter-dependent. A variable translation along an arbitrary axis $\mathfrak{a}$ is given by

$$\mathbf{T}_{\text{trans}}(q) = \{\mathrm{Id}_3, q\mathfrak{a}\}$$

A rotation around an arbitrary axis $\mathbf{a}$ through the local origin is given by

$$\mathbf{T}_{\text{rot}}(q) \;=\; \{\text{ROT}(\mathbf{a}, q), \mathbf{0}\}$$

Both transforms are dependent on one scalar parameter q and $\mathbf{T}_{\text{trans/rot}}(0)$ is the identity. The joint axis $\mathbf{a}$ is assumed to be of unit length.

3.2 Manipulator Systems

3.2.1 Modelling Joints Between Objects

Within a robot, several rigid objects are connected by movable joints. A **joint** between two objects o and v is modeled in general by three homogeneous transforms to form one parameter-dependent transform:

- a fixed transform to the joint origin: ${}_o\mathbf{T}$
- a parameterized transform to make the joint movable, either a rotation around or a translation along the **joint axis** $\mathbf{a}_v$: ${}_o\widehat{\mathbf{T}}_v(q_v)$
- a fixed transform to the object's origin: $\mathbf{T}_v$

The joint transformation is calculated by chain multiplication:

$${}_o\mathbf{T}_v(q_v) \;=\; {}_o\mathbf{T}\,{}_o\widehat{\mathbf{T}}_v(q_v)\,\mathbf{T}_v$$

The joint parameter q_v is limited in a closed interval, $q_v \in [q_{v_{\min}}, q_{v_{\max}}]$. These limits model the motion range of a joint and are practically set up by the technical constraints of the joint construction (see remarks in section 3.3.2). Physical constraints imply that the two objects are *in touch* for all possible values of q_v, thus requiring zero distance between the models:

$$q_v \in [q_{v_{\min}}, q_{v_{\max}}] \;\Rightarrow\; \min_{\substack{\mathbf{u}\in G_o \\ \boldsymbol{w}\in {}_o\mathbf{T}_v(q_v)G_v}} (\,\|\,\mathbf{u}-\boldsymbol{w}\,\|\,) \;=\; 0 \qquad (3.1)$$

Note that q_v either describes an angle (in the case of a rotational/revolute joint) or the units of spatial length (in the case of a translational/prismatic joint).

3.2.2 Modelling Manipulator Systems

Manipulators consist of a set of n links connected by n joints. Both links and joints are referred to by their number $i = 1, \ldots, n$. The links i are rigid objects that are modeled by the set of geometry models G_i. All environmental obstacles form one static "object" that is referred to as object 0 and modeled by the geometry model G_0 within the *workspace frame*.

Each link i is connected by joint i to the link v_i. To simplify the notation, the numbering of the links/joints is assumed to be monotone, i.e. a link is always connected to a link with a smaller number or the environment, $v_i \in \{0, \ldots, i-1\}$.

Each link i is connected to the environment by a sequence of joints $i, v_i, v_{v_i}, \ldots$. These numbers (they do not include the 0) are listed in the index set I_i, the cardinality n_i of I_i is the number of joints that influence the position of link i.

In the case of $v_i = i - 1$, $n_i = i$, a **serial manipulator** is modeled. A more general term for such a system is *open kinematic chain*. In the general case $v_i \in \{0, \ldots, i-1\}$, a tree-like *forked kinematic chain* is modeled. This can for example consist of several separate serial manipulators or one manipulator equipped with a gripper or a multi-fingered hand. The general term used for tree-like kinematic structures is **manipulator system**. Within this work, the term manipulator is used to refer to serial manipulators as well as manipulator systems. Closed kinematic loops are not considered. Two examples of manipulators are shown in fig. 3.1.

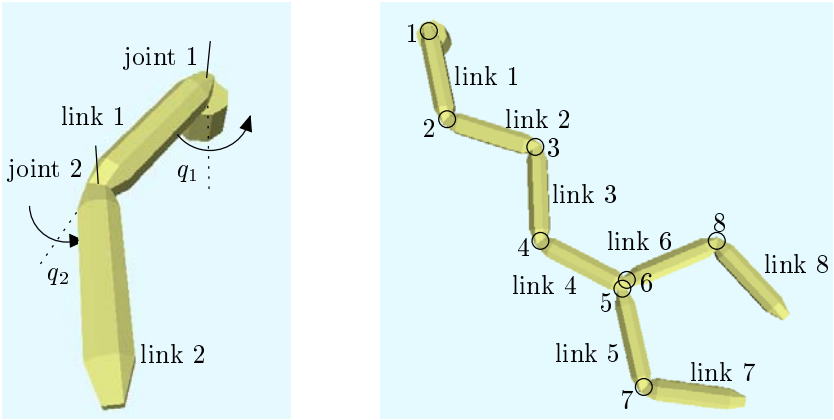

Figure 3.1: Two sample manipulators. The left example shows a 2-joint manipulator, the right example shows a forked 8-joint manipulator. The joints and links are numbered accordingly, i.e. each link i is connected with the joint i to the kinematic structure

The position of joint i is denoted as q_i, $q_i \in [q_{i_{\min}}, q_{i_{\max}}]$. The n-dimensional vector $\mathbf{q} = (q_1, \ldots, q_n)^T$ describes the state of the manipulator system, the **configuration**.

To simplify notations, all transforms used with links of manipulator systems are written with the full configuration $\mathbf{q}$ as parameter. The transform ${}_{v_i}\mathbf{T}_i(\mathbf{q})$ describes the joint relation between the frames of links i and v_i. Note that the value of this transform is actually dependent on the value q_i only.

The transform describing the position of link i within the workspace frame is calculated by chain multiplication of the joint transforms of all joints between link i and the environment 0:

$$ {}_0\mathbf{T}_i(\mathbf{q}) = \prod_{j \in I_i} {}_{v_j}\mathbf{T}_j(\mathbf{q}) $$

It should be noted that this transform (and thus the position of link i) is only dependent on $n_i \leq n$ components of $\mathbf{q}$.

The number of kinematic **degrees of freedom** (abbrev. **DOF**) of a manipulator system is the number of joints n within this system. The DOF of an individual link i is n_i. The number of geometric degrees of freedom of the last link of a manipulator, the *tool*, depends on the number and arrangement of joints, but is maximally 6 (three translational and three rotational geometric degrees of freedom in $\mathbb{R}^3$). So-called *universal manipulators* have at least 6 joints (in a proper arrangement) to enable 6 geometric degrees of freedom for the tool, i.e. the tool can be positioned at any point and with any orientation – limited by physical reachability only.

Some additional remarks on this type of modelling manipulator systems:

- The use of manipulator systems as defined above appeared necessary to model grippers and other handling devices or tools. A linear manipulator is unable to interact with its surroundings. If a manipulator grips an object in the environment (a part within G_0) to transport it, it becomes a part of the manipulator itself. This "switching of objects" happens *between* tasks and does not need to be considered for low-level motion planning. As explained in the first chapter, this falls into the responsibility of the higher level planning system that maintains the world model.
- Closed kinematic chains require a completely different mathematical treatment and are neglected. Typical closed chain devices are the stewart platform with three "legs" fixed to the ground and to the platform itself, and related mechanical setups. Closed chains do also occur if, for example, two manipulators both grip and handle the same object (see e.g. Koga and Latombe (1994) or Fischer (1996)). The problem of cooperative manipulation is not considered within this work.
- Open kinematic chains that are not connected to the environment at one end have to be separated into two groups. The first group, robots which crawl like snakes, are nowadays of little practical interest and therefore not further considered. The other group consists of manipulators which are attached to mobile platforms, with usually three degrees of freedom (if moving on a plane) between the environment and the "first" link that physically exists. These robots are of practical interest and considered as special cases within this work, although they are no real manipulator systems. They can be modeled with "empty" geometry models for links 1 and 2 (see section 8.2.7, p. 143, for an example).

3.2.3 Collision of a Manipulator System

A link i collides with the environment at a certain configuration $\mathbf{q}$, if the intersection of its model G_i at the configuration-dependent position with G_0 is

non-empty:

$$ {}_o\mathbf{T}_i(\mathbf{q})\, G_i \cap G_0 $$

A link i collides with link j, if their configuration-dependent models intersect in the worldspace. This can be simplified, if one of the models is transformed into the reference frame of the other model:

$$ {}_0\mathbf{T}_i(\mathbf{q})\, G_i \cap {}_0\mathbf{T}_j(\mathbf{q})\, G_j \;=\; {}_0\mathbf{T}_j(\mathbf{q})^{-1}\, {}_0\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j \;=\; {}_j\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j $$

If two links are connected by a joint, their models will typically intersect, unless the models are very precise. For this reason, neighbouring links are excluded in the definition of collision for a configuration. A certain configuration is called **colliding**, if any of the links collide with the environment or with any non-neighbouring link:

$$ \begin{aligned} \mathbf{q} \text{ collides} \quad :\Leftrightarrow \quad \exists\, i,j \;:\; & 2 \le i \le n \quad \wedge \\ & 0 \le j < i \quad \wedge \\ & j \neq v_i \quad \wedge \\ & {}_j\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j \neq \emptyset \end{aligned} \tag{3.2} $$

This compact definition is possible due to the numbering conventions for joints/links of manipulator systems set up above. As $v_j < j$ and $j < i$, $i \neq v_j$ always holds. A configuration is called **collision-free** if it does not collide. Note that link 1 (as all other links with $v_i = 0$) is not tested for collision with the environment, as it is "in touch" with it, see (3.1). Possible collisions with obstacles are avoided by proper selection of $q_{1_{\min}}$ and $q_{1_{\max}}$. (It should be noted that this somewhat restrictive modelling is used to simplify the formal description. Within an implementation, a dedicated object between the first movable links and the environment – the fixed robot base – would be treated separately, being checked for collision with non-neighbouring links in addition to the check with G_0.)

3.3 Workspace and Configuration Space

3.3.1 Workspace

The spatial region that can be accessed by a manipulator is limited. The area a manipulator reaches within $\mathbb{R}^3$ is called its **workspace** and denoted with W. The space that is free of obstacles is called the **free workspace** W_{free}.

$$ \mathsf{W} \;=\; \bigcup_{i=1}^{n} \left(\bigcup_{\substack{j \in \{1,\ldots,n\} \\ q_j \in [q_{j_{\min}}, q_{j_{\max}}]}} {}_0\mathbf{T}_i(\mathbf{q}) G_i \right) $$

$$ \mathsf{W}_{free} \;=\; \mathsf{W} \setminus G_0 $$

These definitions, as well as the following, are based on models instead of being based on reality. As the real space is not accessible for the planner, this makes sense. W_{free} is the *model of* the free space that is available for planning. Reality and model are treated synonymously in the following, what is actually referred to is a matter of context.

3.3.2 Configuration Space

All possible configurations $\mathbf{q}$ span up the **configuration space** C. This space is usually referred to as **c-space**. The dimension of C is n, the number of joints. As only bounded joint values are allowed, C is an interval of $\mathbb{R}^n$:

$$\mathsf{C} = [q_{1\min}, q_{1\max}] \times \ldots \times [q_{n\min}, q_{n\max}]$$

The subset of colliding configurations within C is denoted as C_{coll}, the complement C_{free} is the set of all collision-free configurations. The set of configurations within C_{coll} are also referred to as *c-obstacles*, as they – somehow – represent the workspace obstacles in the c-space (see remarks below). C_{free} and C_{coll} are illustrated for the 2-DOF manipulator in fig. 3.2.

$$\mathsf{C}_{coll} = \{\mathbf{q} \in \mathsf{C} \mid \mathbf{q} \text{ collides}\}$$

$$\mathsf{C}_{free} = \mathsf{C} \setminus \mathsf{C}_{coll}$$

Some important remarks:

- The c-space as defined above is a hypersquare in $\mathbb{R}^n$. It should be noted that the different components represent the positions of the different joints of the manipulator system. If a joint moves a sequence of links (e.g. the first joint of a linear manipulator), the respective axis of the c-space has a longer effective range in the workspace and implies "more movement" compared to joints closer to the tool of a manipulator.
- In literature, freely rotating joints are allowed for manipulators quite frequently (e.g. in Latombe (1991)). This results in some unbound dimensions for the c-space and a cyclic repetition (patches with the periodicity of 2π) of the same c-obstacles. Within the context of this work, these unbound joints are neglected as they are not used for real robots.
- The term c-obstacle is misleading. The c-obstacles represent the obstacles within the workspace *and* the shape and size of the links of the robot *and* the arrangement of joints. The topology of the c-obstacles is not very obvious.
- Note that a "map" of the c-space as shown in fig. 3.2 can only be drawn for two-dimensional manipulators. Motion planning is trivial for two joint robots, the examples are presented for illustration purposes only.

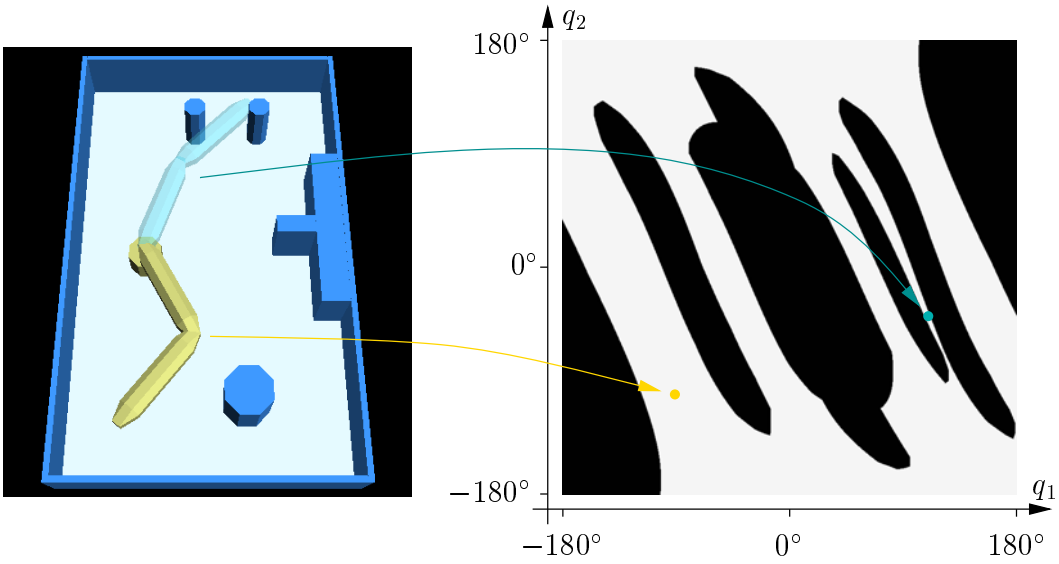

Figure 3.2: Workspace and configuration space for 2-DOF manipulator. The left picture shows the 2-DOF manipulator at two different configurations in an environment with obstacles. The right image maps these obstacles into the configuration space, black areas represent C_{coll}, bright areas C_{free}. The arrows connect the workspace postures with the respective configurations.

3.3.3 Paths

A **path** $\mathbf{P}(t)$ is a continuous map of a formal scalar parameter t into the c-space:

$$\mathbf{P} : [0,1] \to \mathsf{C}$$

If the parameter t is omitted, $\mathbf{P}$ denotes all possible configurations along the path, $\mathbf{P} = \cup_{t\in[0,1]}\mathbf{P}(t)$. Another term for path is *kinematic trajectory*. Throughout this work, the term path is used, to distinguish it from *(dynamic) trajectories* that are parameterized over time.

A path $\mathbf{P}$ is called colliding if any of its configurations collides, otherwise it is called **collision-free**.

$$\begin{aligned} \mathbf{P} \text{ collides} \quad &:\Leftrightarrow \quad \exists\, \mathbf{q} \;:\; \mathbf{q} \in \mathbf{P} \wedge \mathbf{q} \text{ collides} \\ \mathbf{P} \text{ collision-free} \quad &:\Leftrightarrow \quad \forall\, \mathbf{q} \in \mathbf{P} \;:\; \neg(\mathbf{q} \text{ collides}) \end{aligned}$$

3.4 Objectives

This section gives a formal statement of the objectives **O1** to **O3** that were set up in section 1.2.3, p. 8. Objective **O3** may contradict objective **O2**, as the assurance of safety distances may result in a detour that is necessary to allow for modelling and control uncertainties. Therefore, the objectives are inclusive: the safety distance has to be found for collision-free paths, and the short paths should keep the safety distance.

3.4.1 O1 – Plan Collision-Free Paths

The basic motion planning problem for manipulators is to find a collision-free path between a start configuration $\mathbf{q}_{start}$ and a goal configuration $\mathbf{q}_{goal}$. In literature, this problem is often called Generalized Mover's Problem (abbrev. **GMP**): Connect two points in a space of arbitrary dimensions with static obstacles. Using the terms defined in the previous section, objective **O1** can be expressed as follows:

Given two configurations $\mathbf{q}_{start} \in \mathsf{C}_{free}$ and $\mathbf{q}_{goal} \in \mathsf{C}_{free}$, find a path $\mathbf{P}$ with

$$\mathbf{P}(0) = \mathbf{q}_{start}\,, \quad \mathbf{P}(1) = \mathbf{q}_{goal}\,, \quad \mathbf{P} \text{ collision-free}$$

Some additional remarks:

- The usual definition of **GMP** in literature is not necessarily based on a c-space made up by the joint values of a manipulator, but on an arbitrary space with arbitrary obstacles. This explains the name: a point-object has to be moved through the obstructed space. And in C, the robot is just the point described by the components of a configuration $\mathbf{q}$.

- If any collision-free solution for a given task exists, it is obvious that the number of possible solutions is infinite. This allows to qualify possible solution paths, and preferably to plan "better" paths, as it is implied by the objectives **O2** and **O3**.

3.4.2 O2 – Keep Safety Distances

Safety distances along a path enable the motion planning system to take modelling uncertainties of the obstacles and control uncertainties of the robot itself into account.

The distance between a complete manipulator and the environmental obstacles is always zero, as the first link "touches" the fixed base. Thus, it is necessary to treat the individual distances d_i of all links i separately. For planning purposes, the real distance is of no interest if it is beyond a certain predefined threshold $d_{\max_i}$ that covers all modelling and control uncertainties. And this threshold can actually be expected to be relatively small in relation to the dimension of the robot, typically in the order of a few centimetres.

These thresholds form a n-dimensional vector $\mathbf{d}_{\max}$ that is given as an additional task parameter. As noted above, $d_{\max_1} = 0$ is a physical necessity. More general, for manipulator systems, $d_{\max_i} = 0$, if $v_i = 0$. It is also assumed that the safety distance does not imply collisions between non-neighbouring links in *any* configuration, i.e. the safety distance for each link i is small enough to not reach link v_{v_i} independent from the value q_{v_i} and $q_{v_{v_i}}$. Otherwise, it would not be possible to achieve a path keeping the safety distances even in the *absence* of obstacles.

The current safety distance for a certain link i is configuration-dependent, denoted as $d_i(\mathbf{q}) \in [0, d_{\max_i}]$. For a configuration $\mathbf{q} \in \mathsf{C}_{free}$ it is calculated as follows:

$$d_i(\mathbf{q}) \;=\; \min \left(d_{\max_i} \,,\; \left(\min_{\substack{0 \le j < i,\, j \neq v_i \\ \mathbf{u} \,\in\, {}_j\mathsf{T}_i(\mathbf{q}) G_i \\ \boldsymbol{w} \,\in\, G_j}} \left(\| \,\mathbf{u} - \boldsymbol{w}\, \| \right) \right) \right) \tag{3.3}$$

The values of $d_i(\mathbf{q})$, $i = 1, \ldots, n$ form a vector, the **configuration-dependent safety distance** $\mathbf{d}(\mathbf{q})$. It should be noted that this definition is somewhat arbitrary. The maximum distance calculated between the individual links is based on their *number*, for each pair of links $i > j$ the distance value $d_{\max_i}$ is used. There are other, more complex possibilities, e.g. a safety distance matrix that defines a maximum value $d_{\max_{i,j}}$ for each possible pair of links i, j.

Objective **O2** is to attempt to maximize the safety distance for a path solving **GMP**: Given two configurations $\mathbf{q}_{start}, \mathbf{q}_{goal} \in \mathsf{C}_{free}$ and a vector of desired safety

distances $\mathbf{d}_{\max}$, find a collision-free path $\mathbf{P}$ with $\mathbf{P}(0) = \mathbf{q}_{start}$, $\mathbf{P}(1) = \mathbf{q}_{goal}$ that *maximizes* $\mathbf{d}(\mathbf{q})$ for all $\mathbf{q} \in \mathbf{P}$.

Some additional remarks:

- Note that the maximum value $\mathbf{d}_{\max}$ cannot be achieved for all configurations of a path, especially if $\mathbf{d}(\mathbf{q}_{start})$ or $\mathbf{d}(\mathbf{q}_{goal})$ are not equal to $\mathbf{d}_{\max}$. There may also be tight passages along a path were it is impossible to achieve full safety distances.
- This work does not intend to find the *optimal* solution, but a *good* solution quickly (objective **O4**).
- To find a solution that maximizes the components of $\mathbf{d}(\mathbf{q})$ along a path is a multi-dimensional optimization problem. It is impossible to decide which links' safety distance is of higher importance and should be preferred. This might be improved by weightening the components of $\mathbf{d}(\mathbf{q})$, but that is beyond the scope of this work.
- An ideal solution that fulfils **O2** lifts the robot's links more or less orthogonally (in W) away from the obstacles into areas of C with $\mathbf{d}(\mathbf{q}) = \mathbf{d}_{\max}$, transfers close to $\mathbf{q}_{goal}$ and approaches the obstacles again orthogonally in W. The path shape in the regions of maximal safety distance is arbitrary.

3.4.3 O3 – Prefer Short Paths

The **path length** $l_{\mathbf{P}}$ is the length of the curve described by a path $\mathbf{P}$ in the configuration space C:

$$l_{\mathbf{P}} = \int_0^1 \| \mathbf{P}'(t) \| \, dt \tag{3.4}$$

Objective **O3** ist to attempt to minimize the length of a path that solves **GMP** and keeps safety distances wherever possible. Given two configurations $\mathbf{q}_{start}, \mathbf{q}_{goal} \in \mathsf{C}_{free}$ and a vector $\mathbf{d}_{\max}$ of desired safety distances, find a path $\mathbf{P} \in \mathsf{C}_{free}$ with $\mathbf{P}(0) = \mathbf{q}_{start}$, $\mathbf{P}(1) = \mathbf{q}_{goal}$ with maximum $\mathbf{d}(\mathbf{q})$, that *minimizes* $l_{\mathbf{P}}$.

Some additional remarks:

- As noted above in regard to the safety distances, it is not intended to find the overall shortest path, but a quite short one quickly (objective **O4**).
- There are other possibilities to define a path length, e.g. by measuring the distance travelled by the tool in W. As the robot is technically controlled in C, the above definition is appropriate. A possible extension to control the comparability of different solutions is to weighten the axes of C. See chapter 9.1, p. 153, for an extended discussion.

3.5 Approach – Path Rating and Modification

A closer look at the objectives **O2** and **O3** as defined above shows that these objectives implicitly describe a kind of rating of possible solutions to **GMP**. Collision free paths can be compared if there are several of them. And if they differ, the *best* one among them can be selected. This possibility is proposed as a general planning approach, applicable to collision-free path planning as well, if an appropriate rating function is used.

To be able to handle paths efficiently, the BB-method works with polygonal paths. They are modeled by a sequence of intersection configurations only, but can approximate any path with arbitrary precision. Within the framework of an intelligent robot system, they are also perfectly suited to be passed to the trajectory generation of a robot. A **path segment** $\bar{\mathbf{P}}$ is the C-linear interpolation between two configurations $\mathbf{q}_a, \mathbf{q}_b \in \mathsf{C}$:

$$\bar{\mathbf{P}} \;=\; \overline{\mathbf{q}_a \mathbf{q}_b} \;=\; \bigcup_{t\in[0,1]} t\mathbf{q}_a + (1-t)\mathbf{q}_b$$

The notation $\bar{\mathbf{P}}$ and $\overline{\mathbf{q}_a \mathbf{q}_b}$ is used synonymously. A **polygonal path** $\hat{\mathbf{P}} \in \mathsf{C}$ is a continuous sequence of l path segments $\bar{\mathbf{P}}_k$ between $l+1$ configurations, the **intersection configurations** $\mathbf{q}_k$, $l \in \mathbb{N}^+$:

$$\hat{\mathbf{P}} \;=\; \bigcup_{k=1}^{l} \overline{\mathbf{q}_{k-1}\mathbf{q}_k} \;=\; \bigcup_{k=1}^{l} \bar{\mathbf{P}}_k$$

With this kind of path modelling, the following principle tasks of a planning algorithm based on path modification can be identified:

- **Path rating**
  Appropriate rating functions have to be developed that allow the comparison of the "quality" of paths and path segments respectively.
- **Candidate creation**
  Create candidate configurations for appropriate intersection configurations within a path $\hat{\mathbf{P}}$ that are suitable to form possible alternative paths.
- **Replacement selection**
  Rate possible alternative paths where candidates replace the respective intersection configurations. From the candidates that yield improved rating, take the best ones and insert them to replace their respective intersection configurations.
- **Path refinement**
  If no candidate can be found to improve the overall rating, adapt the polygonal path by inserting new intersection configurations along the existing segments. Thus the respective segments are divided to allow further topological adaption of the path with regard to the rating function.

- **Termination**
  Candidate creation, replacement selection and path refinement are iterated repeatedly. If the rating achieves a certain value for the entire path, the planning terminates successfully. If no appropriate segment can be found to be modified and no segment is allowed to be divided, the planning algorithm fails and further improvement with respect to the objective is impossible. A typical reason for this termination is that the length of the segments to be divided fall below a certain threshold.

To enable efficient planning, the rating and modification within the BB-method takes place by considering attributes calculated for selected paths only. Neither the whole workspace nor the whole c-space is considered. The means to calculate candidates, to subdivide path segments and the respective parameters are founded on heuristics that are developed based on considering the motion of the manipulator among and *through* the obstacles.

3.6 Illustration Examples

Three example tasks were chosen to illustrate the different planning components in detail in the following chapters. More examples and experiments are presented in chapter 8, p. 127.

The 2-DOF Manipulator

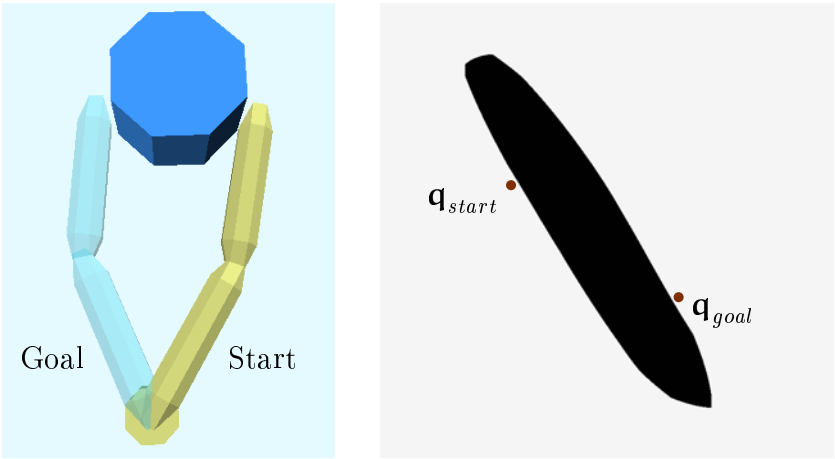

Figure 3.3: Workspace and configuration space for an illustrating sample task for the 2-DOF manipulator. The c-space map shows only a selected interval out of the full c-space of this manipulator.

To introduce the rating functions and the modification schemes employed for the BB-method, a simple 2 DOF example is chosen (see fig. 3.3). The robot has

to find a motion around an octogonal obstacle. The "elbow" has to pass through full stretch along this motion. The c-space map shows that there are two possible solutions, the stretching can take place either on the left-hand side or on the right-hand side of the obstacle. On either side of the obstacle, the robot is almost "in touch" with it.

The 6-DOF Manipulator

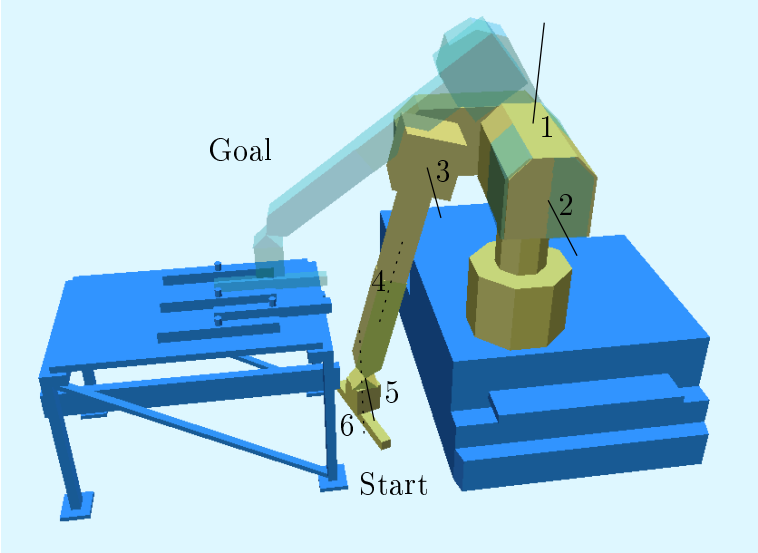

Figure 3.4: Workspace for an illustrating sample task for a 6-DOF manipulator. The six joint axis are marked.

Figure 3.4 shows a difficult example task out of the manufacturing domain. For clarity, all parts in the environment that are not of interest for the task itself were removed. The robot has to lift a lengthy load up from the floor onto the table. It can be assumed, that the part had fallen down, was localized with sensors and has to be stored again at the proper position.

The task is very difficult for several reasons. The load itself is quite long and has to be kept more or less orthogonal to its final position all the way up between the robot's base and the table. It has to be turned above the niveau of its final position. The overall motion requires the arm to move *upwards* and *backwards* simultaneously. To keep the orientation of the tool in a suitable range makes it necessary to consider all joints for this planning task. The load is "in touch" with the floor at the start position and in touch with the table at the goal position.

The kinematic structure equals a standard MANUTEC industrial manipulator. The first and second axes are orthogonal, the third axis is parallel to the second axis, and axes four, five and are orthogonal with respect to their prede-

cessor. The joint ranges are 215°, 280°, 280°, 380°, 180°, 360° (joints one through six).

The 16-DOF Manipulator

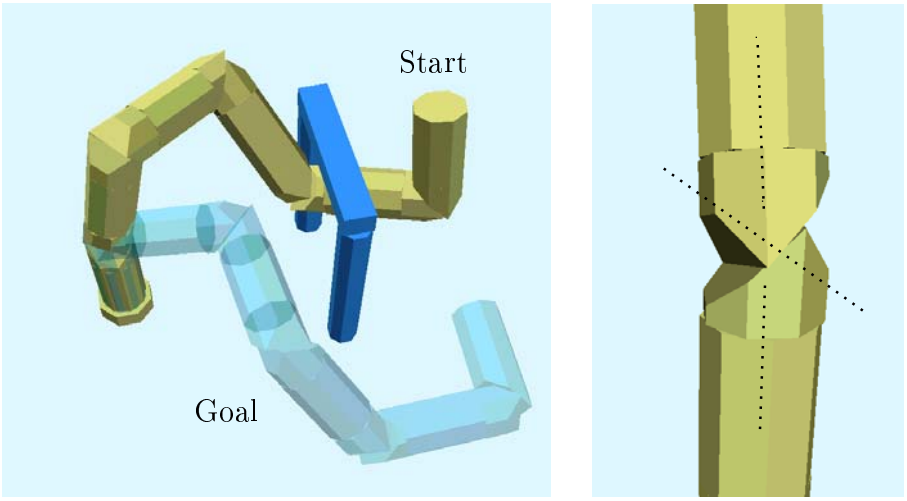

Figure 3.5: An illustrating sample task for a 16-DOF manipulator. Left: The task requires to move out of a gate. Right: The manipulator consists of five equaly modeled three-joint connections. The middle joint axis is turned by 45° with respect to the preceeding and the following axis.

This is an example for a possible future hyperredundant manipulator. A comparable structure was developed at the Jet Propulsion Laboratories for the maintainance of nuclear power plants and other applications in hazardous environments (see Paljug et al. (1995)).

The robot has 16 joints. The base joint allows it to rotate the arm completely. The remaining 15 joints are made up of a sequence of five equal three-joint connections (see fig. 3.5). All joints have a motion range of 360°.

The selected task is extremely difficult. The manipulator has to move its foremost four links through a very tight gate to be able to move to the desired goal. The passage in C that is suitable for this motion can be assumed to be very small. If it is attempted to move the robot out of the gate manually by carefully turning individual joints, it requires tens of minutes, if not hours, to find a solution.

Chapter 4

O1 – Plan a Collision-Free Motion

This chapter develops the rating function for colliding paths and the respective modification algorithms. The idea can be explained informally as follows: The robot's geometry model is moved along a certain initial path. If there are obstacles, its size is reduced until it can pass. To plan a collision-free motion, the robot is in turn "blown up" again and the path is "bent" until the original size is reached and the robot can move in its original size.

4.1 Rating Function Q_{scale} – Reducing the Robot's Size

4.1.1 Intention and Approach

In reality all colliding paths are just colliding and cannot be rated. But using the geometry models, it is possible to calculate *virtual* properties of colliding configurations and paths.

The rating function for collision-free motion has to be able to distinguish alternative paths that are colliding. Paths that are "less colliding" have to be perceived as "better" paths. To do so, a rating function has to measure the "degree of intersection" of the robot – paths are better if the robot intersects "less". The most important aspect of the rating function is the comparability of paths, the real intersection depth is of no specific interest.

Rating Colliding Configurations

The problem can be addressed by considering single configurations first, and extending this rating to entire path segments. There is no obvious physical differentiation possible for colliding configurations, in reality they are equally inaccessible for the robot.

Rating the collision state of a robot at a certain configuration requires a measure of "intersection depth". It has to be applicable to complete manipulators. Therefore, some desired attributes can be set up:

- All configurations $\mathbf{q} \in \mathsf{C}_{free}$ are rated equal with a maximum rating value.
- The rating should be defined single-valued and one-dimensional for all configurations $\mathbf{q} \in \mathsf{C}$, as only this allows comparison.

- For serial manipulators, the following aspect is obvious: if link i is the first colliding link for a configuration $\mathbf{q}_u$, and link j, $j < i$ is the first colliding link at a configuration $\mathbf{q}_v$, $\mathbf{q}_v$ should be rated worse than $\mathbf{q}_u$.

The chosen approach is not to measure the intersection, but the *size reduction* required for the robot's geometry model to avoid collisions. In the case of the 2 DOF example, link two is shown colliding at different configurations in fig. 4.1, and obviously, the intersection depth is very different. Different values of *scale*, intuitively proportional to the "degree of intersection", can be assigned to the configurations by maximizing the possible collision-free scaling of the geometry model:

$$scale \;=\; \max_{s\in[0,1]} \left({}^{s}_{0}\mathbf{T}_2(\mathbf{q})\, G_2 \cap G_0 \;=\; \emptyset\right)$$

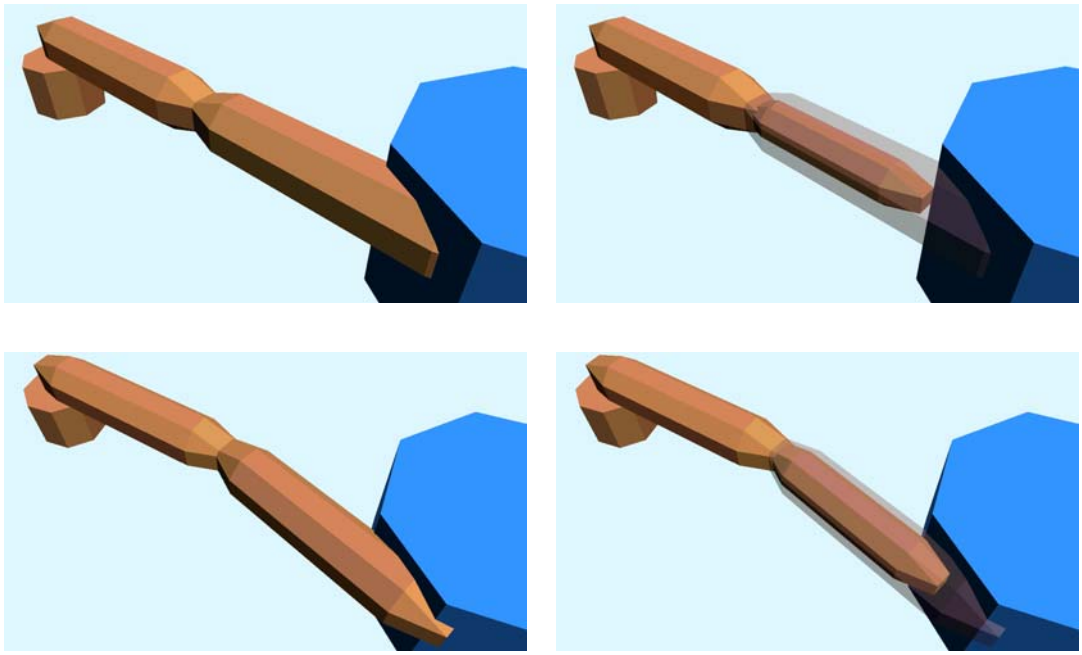

Figure 4.1: Left: Two different colliding locations for the 2-DOF manipulator. Right: Link 2 is scaled to its maximum collision-free size, original model shown transparent. The scaling captures the degree of intersection.

This way of rating is a kind of "reverse intersection depth", as it does not measure how far the link intersects, but how much it has to be "shrunk" so as not to intersect. The real size reduction implied by the scaling is not considered, as it is of no importance for the comparison of ratings, only the relative size is used. This scaling based rating of a single object in collision can be extended to complete manipulator systems and their motion along path segments.

4.1.2 Scaling of Links and Manipulator Systems

The scaling of an object moves all points within the geometry model towards its local origin. If the local origin itself gets *inside* a colliding object, the scaling is zero. This has to be avoided, as it would not capture the intersection depth and thus it cannot serve as rating value.

What is the consequence? The scaling has to take place in a way that the origin is **always** collision-free. For a manipulator system, the first link is always collision-free. Thus, if the second link is colliding, it can be scaled towards a point within link one. And such a point always exists as joints imply a physical connection (see (3.1)). If a link i_{coll} is the colliding link, $G_{i_{coll}}$ can always be scaled with respect to a point within $G_{v_{i_{coll}}}$. For this reason it is assumed that the modelling is done in a way that the local origin is within the preceding link (note the zero scaling applied to G_i, resulting in a single point):

$$ {}^{0}_{v_i}\mathbf{T}_i(\mathbf{q})G_i \;\in\; G_{v_i} \tag{4.1} $$

This has to be fulfilled for all links $i > 2$ and for all $q \in \mathsf{C}$. This is easily assured as the links are physically connected, and the local origin can be chosen at a proper position (i.e. the reference frame of the geometry model can be set up accordingly, see 3.2.1, p. 33).

To gain a single-valued rating of a configuration the scaling factor of one link is sufficient. The links are tested in the order they are labled, a first colliding link i_{coll} can be identified and scaled. The links *beyond* the first colliding link are neglected. They can be seen as scaled down to zero size. For a linear kinematic chain this treatment suggests itself. For forked manipulators, it is a convention to achieve one-dimensional rating. Illustrating examples are shown in fig. 4.2.

If the robot does not collide with an obstacle, but with itself, the respective link indices shall be i, j, $i > j$. The "first" colliding link cannot be the link with the smaller index j, as link i "disappears" if link j is scaled, according to the concept above. Link i can only be scaled with respect to link j. Therefore, the first colliding link in the case of self-collision is the higher indexed link $i = i_{coll}$ (see fig. 4.2, right).

Only the number of one link (the first colliding one i_{coll}) and its scaling factor are relevant for rating. These two numbers can be combined into a single value, and an intuitively understandable assignment to a real value between 1 and n is found by adding the number of collision-free links $i_{coll} - 1$ prior to the colliding link i_{coll} and the scaling factor found for link i_{coll}. A collision-free configuration is thus rated with n, indicating all links are collision-free. As the rating is used for comparison only, this artificial value is perfectly suited.

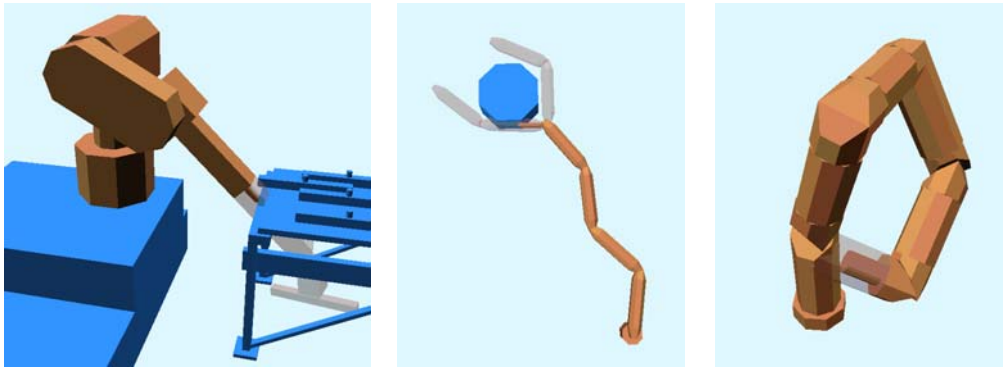

Figure 4.2: Left: The first colliding link for the 6-DOF manipulator is link 4. It is scaled, link 5 and 6 are not further considered. Middle: The forked 8-DOF manipulator collides with link 5, links 6 through 8 "disappear". Right: The 16-DOF manipulator collides with itself. Link 16 is scaled with respect to link 1 to rate the shown configuration.

4.1.3 Rating a Path Segment

On the one hand, the rating for a path segment should indicate the "deepest collision" that occurs for the motion along the segment, i.e the minimum scaling factor that occurs if all configurations along $\mathbf{q}$ are tested. On the other hand, it should be the scaling factor of the largest possible geometry model that is collision-free for the whole motion, i.e. the volume swept out by the moving robot along the path should just "touch" the obstacles at the configuration of minimal scaling. Only if this holds, the comparison between several alternative segments can be significant: one segment is better if an overall larger robot can pass. This condition can be formally expressed. For a link i moving relative to a link j, $j < i \wedge j \neq v_i$, the configuration $\mathbf{q}_{min}$ with the smallest scaling for a path segment $\bar{\mathbf{P}}$ in the case of a collision between i and j is given by:

$$\mathbf{q}_{min} = \min_{\mathbf{q} \in \bar{\mathbf{P}}} \left(\max_{s \in [0,1]} \left({}^{s}_{j}\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j = \emptyset \right) \right)$$

The scaling *scale* at this particular configuration is given by:

$$scale = \max_{s \in [0,1]} \left({}^{s}_{j}\mathbf{T}_i(\mathbf{q}_{min})\, G_i \cap G_j = \emptyset \right)$$

The condition that has to hold is the following:

$$\forall \mathbf{q} \in \bar{\mathbf{P}} : {}^{scale}_{j}\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j = \emptyset \tag{4.2}$$

If this does not hold, the rating becomes meaningless for comparison. This

may occur if the scaled model is not included in the original one. It is impossible if the model G_i is either convex or star-shaped with respect to its origin:

$$s_1 < s_2 \;\Leftrightarrow\; {}^{s_1}\mathbf{T}\, G_i \;\subset\; {}^{s_2}\mathbf{T}\, G_i \tag{4.3}$$

To allow a significant scaling of path segments this is set up as a necessary condition for the links' geometry models.

Eventhough this may appear as a limiting requirement, it is not. Usually robot links are convex. If they are not convex, planning can use the convex hull without any considerable loss of free space in most cases (see e.g. Barber et al. (1993)). But the convex hull may not be sufficient for any link and especially not for arbitrary loads. To handle non-convex objects, they can be decomposed into convex (or star-shaped) partial objects and modeled separately. The connections between them can be treated analogous to a joint *without motion range*. Thus it is possible to model arbitrary non-convex objects as a statically connected (possibly forked) sequence of convex objects that fulfil (4.1) and (4.3). This kind of extended modelling is not used within this work to keep the notation as simple as possible, but it is important to note that (4.3) does not limit the general applicability of the rating in particular and the BB-method in general.

The developed way of rating by scaling can be seen as a "virtual" retraction of the robot "into itself". All rating values are implied by a model of the robot that is completely contained in the original model. The following algorithm defines a rating function Q_{scale} that maps path segments onto real values. A possible practical realization for Q_{scale}, especially considering line $(*)$, is presented in chapter 7.

$$Q_{scale} : \mathsf{C} \times \mathsf{C} \to \mathbb{R}$$

Algorithm : Q_{scale}
Input : $\bar{\mathbf{P}}$
Output : $Q_{scale}(\bar{\mathbf{P}})$

$scale := 1$
for $i = 2 \ldots n$ *[All links]*
  for $j = 0 \ldots i-1, j \neq v_i$ *[Obstacles and previous links]*
$*$ $scale_{test} = \max_{s\in[0,1]} \left(\forall \mathbf{q} \in \bar{\mathbf{P}} : {}^{s}_{j}\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j = \emptyset\right)$
    if $(scale_{test} < scale)$ *[Select smallest scaling]*
      $scale := scale_{test}$
      $\mathbf{q}_{min}$:= (Location of minimal scaling)
    endif
  endfor *[Obstacles/previous links j]*
  if $(scale < 1)$ *[Link $i = i_{coll}$ collides and was scaled]*
    return $scale + i - 1$ *[Calculate rating value]*
  endif
endfor *[All links i]*
return n *[the segment is collision-free]*

Q_{scale} is based on condition (3.2), see p. 36. It implicitly realizes a collision detection. If $Q_{scale}(\bar{\mathbf{P}}) < n$, $\bar{\mathbf{P}}$ collides, otherwise $\bar{\mathbf{P}}$ is collision-free. The links i are tested with links $j < i$ only, as only these smaller labeled links can induce a scaling. From all collisions of link i_{coll} with smaller indexed links (including the obstacles with index 0), the smallest required scaling has to be selected to actually calculate the "size" of a robot geometry model that can move without collisions along the complete path segment. The location of minimal rating $\mathbf{q}_{min}$ is used in the following within the path modification, when path segments are divided.

4.2 Path Modification – Local Planning

The rating function assigns a value to each path segment in the (n^2)-dimensional space of segments. Within this abstract space it defines a potential and better path segments have a higher rating. The path modification uses this potential to replace path segments by higher rated alternative segments.

4.2.1 Intention and Approach

To start planning for a given task $\mathbf{q}_{start}$, $\mathbf{q}_{goal}$, an initial path has to be set up. According to objective **O3**, the best possible path is the C-straight connection between start and goal if there are no collisions and no obstacles closer than allowed by the desired safety distance. This path segment (without any intersection configurations) is therefore chosen as the initial path for planning.

If the initial path is colliding, it needs to be cut into segments and these segments have to be modified in a way that the rating is maximally inrceased. The path modification should attempt to minimize the overall planning effort by creating candidate points and alternative paths that can be expected to result in high rating increase.

It is intended to modify an entire segment within one step, i.e. replacing its first and last configuration simultaneously. To be able to do so, two intersection configurations are chosen along the initial path to form a linear path consisting of three segments.

Looking at the 2 DOF example in the workspace, the path of the tip should be "pushed towards" the base of the robot to gain a better rating and finally find a solution. In C-space, the suitable directions are diagonal and less obvious. As the rating is based on the path shape in the *workspace*, the possible replacement candidates are created on workspace based principles as well: they result of displacement approximately orthogonal to the cartesian motion of the robot's tip along a current path segment.

The modification function does a rating of possible alternative paths and selects the best one to replace the initial path. After a successful modification

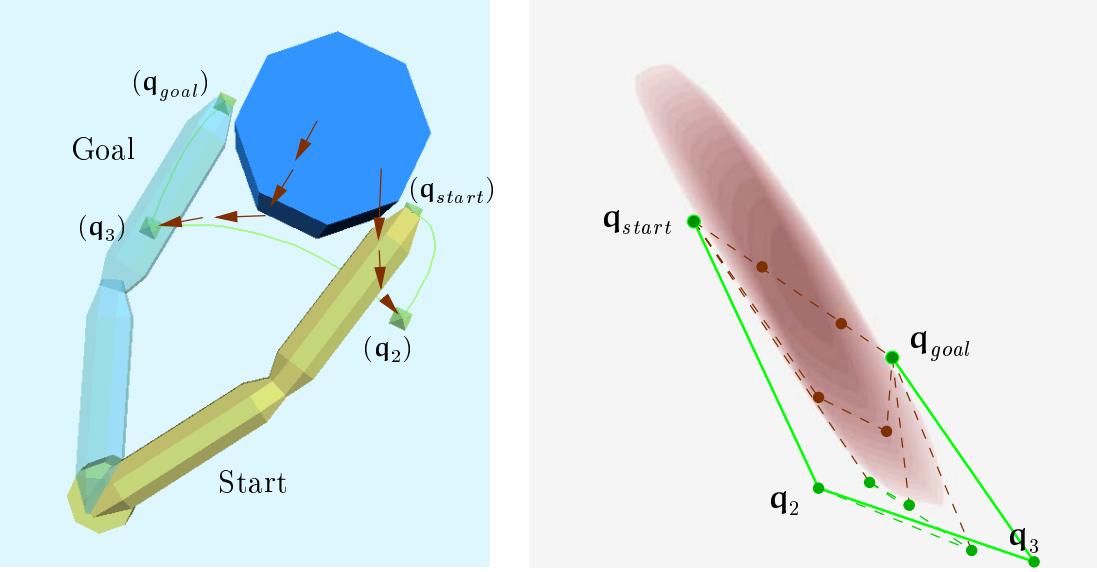

Figure 4.3: Planning for the 2-DOF manipulator visualized in the workspace and in the c-space. Left: Two intersection configurations along the initial path are iteratively shifted out of the obstacle, the displacement direction is chosen approximately orthogonal to the tip point motion along the currently considered segment. Right: The same process shown in the "rated c-space". The colour intensity indicates the required scaling for link two in the respective configuration. Darker regions require a smaller scaled modell to be collision-free. The whole path is modified according to the implied potential.

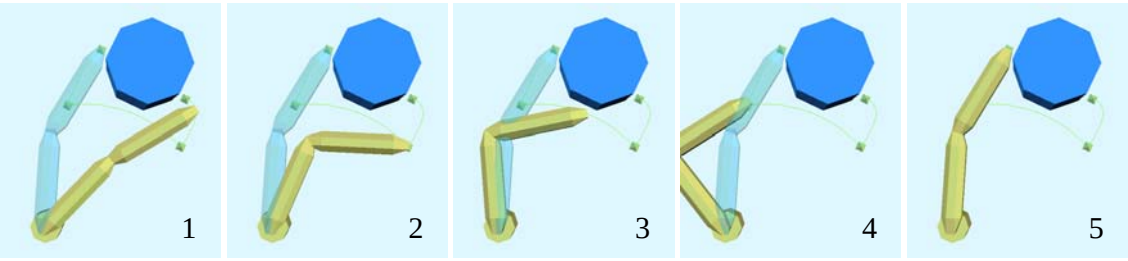

Figure 4.4: A sequence of configurations along the planned motion for the 2-DOF manipulator.

that has not led to a collision-free path, the modification is repeated iteratively, trying to move the segment with the worst rating and the neighbouring segments as well. In the example, the modification repeats four times until a collision-free path is found, i.e. all segments are rated with n. The complete planning process is illustrated in fig. 4.3.

In the example given, the combination of candidates that results in a "stretching" of the robot on the right-hand side of the obstacle is chosen as the planning task is not exactly symmetric. The resulting motion is shown in fig. 4.4.

4.2.2 Candidate Creation

Candidates are created to possibly replace existing intersection configurations along a path. For a single intersection configuration $\mathbf{q}_a$, a set of candidates $\mathbf{c}^a$ is constructed by displacing the intersection configuration in selected directions and with a selected step size. The configuration $\mathbf{q}_a$ is part of a path segment $\bar{\mathbf{P}} = \overline{\mathbf{q}_a \mathbf{q}_b}$, and the link that is currently scaled when $\bar{\mathbf{P}}$ is rated is labeled i_{coll}. The number of joints that are influencing the position of link i_{coll} is $n_{i_{coll}}$, the respective joints are listed in the set $I_{i_{coll}}$.

There are three main aspects to consider:

- **How many candidates are to be created?**
  The position of link i_{coll} has to be modified to allow an increased rating. The number of candidates has to be sufficiently large to allow the adaption of the path in all possible joints $I_{i_{coll}}$. The robot has to be able to use all its degrees of freedom for collision avoidance motion, otherwise possible solutions might not be found. But to modify any joint $j \notin I_{i_{coll}}$ cannot improve the state of collision for link i_{coll}. Therefore, only the $n_{i_{coll}}$-dimensional subspace of C has to be considered.
- **Which is a suitable displacement direction for the current intersection configuration?**
  It is not promising to displace the intersection configuration in the C-direction of the path segment itself, as this will hardly improve the rating. To get the path "away" from its current turn, it is possible to use orthogonal directions in C, as this would result in a C-parallel displacement for complete path segments.
  If seen from a workspace point of view, the colliding link moves on a curve through the obstacle. C-orthogonal directions might displace the path segment roughly along the current motion direction in the workspace and are thus not promising for improved rating. It is desired that the path segment is moved "out of collision", therefore the displacement direction should be orthogonal in W to allow a W-parallel displacement of path segments. Considering just one link moving through collision, these directions obviously result in the most promising candidates to improve the workspace-based rating, as they might shift the whole segment out of collision.

- **What is a suitable displacement step size?**
  Two aspects must be regarded. First, the relative step size for the candidates should be equal. As alternative paths are compared to select the best replacement, the comparison would not yield significant results if the stepsizes differ in magnitude. As the rating is workspace-based, the step sizes should be *equal* in W. Of course, they can only be approximately equal, since a measure of displacement for rigid objects is difficult to define in presence of rotations.
  The second aspect is the absolute step size. The planning process should be able to adapt the path to the obstacle topology. Using a fixed step size is not promising. If a path segment is very short, it has been created by several steps of path refinement out of longer path segments that could not be modified. To attempt to displace it as far as it was attempted to displace the previously longer segment is not suitable. Therefore, the absolute step size should be related to the length of the segment, longer segments are further displaced than shorter segments.

Based on the above considerations, a scheme to create candidate positions is developed. Using C-orthogonal directions, approximate W-orthogonal directions are calculated, and the displacement step size is found based on an approximate measure of the W-motion of the current link i_{coll}.

Calculating C-Orthogonal Directions

To initialize the candidate calculation with directions that use all possible joints, an orthonormal base is calculated in the $n_{i_{coll}}$-dimensional subspace of C, using the direction of $\bar{\mathbf{P}}$ as the first base vector. This can be done using standard mathematical methods and results in $n_{i_{coll}}$ orthonormal vectors $\mathbf{b}_k$, $k = 0, \ldots, n_{i_{coll}} - 1$. The direction of $\bar{\mathbf{P}}$ is $\mathbf{b}_0$. It should be noted that each orthonormal base vector implies *two* possible displacement directions, either in positive or in negative direction. Thus, in the final step, $2(n_{i_{coll}} - 1)$ possible candidates are created out of the initial set of $n_{i_{coll}} - 1$ direction vectors.

For segments of zero length no base can be constructed. If such a degenerated segment occurs within the planning process, no candidates are constructed and planning may possibly fail. But this case is almost impossible for numerical reasons, as it occurs only if an intersection configuration is displaced *exactly* onto another intersection configuration.

Calculating Approximate W-Orthogonal Directions

It is desired to move the intersection configuration "orthogonal in W". In W, the colliding link i_{coll} is a rigid body, and it is difficult to decide which direction is orthogonal to its current movement, especially as rotations are involved. Therefore,

one point on the link is selected, and the displacement direction is orthogonalized in W for this point only.

Looking at the 2 DOF example, it is an obvious choice to use the "tip point" of the moving link. This idea can be extended to any link i within the manipulator system: let $\mathbf{t}_i$ denote the point where the next link is connected, and the foremost point of the link, if no further link follows. As the links beyond i_{coll} are virtually "removed" within the rating, the point $\mathbf{t}_{i_{coll}}$ is the "virtual tip point" of the scaled robot.

Using this control point, the implied cartesian motion $\mathbf{m}_k$ is calculated for the base vectors $\mathbf{b}_k$ at the configuration $\mathbf{q}_a$:

$$\mathbf{m}_k \;=\; {}_0\mathsf{T}_{i_{coll}}(\mathbf{q}_a + \mathbf{b}_k)\mathbf{t}_{i_{coll}} \;-\; {}_0\mathsf{T}_{i_{coll}}(\mathbf{q}_a)\mathbf{t}_{i_{coll}}$$

These displacements are used to modify the base vectors $\mathbf{b}_l$, $l = 1, \ldots, n_{i_{coll}}-1$, in a way that they imply a W-orthogonal motion $\mathbf{o}_l$ of the control point $\mathbf{t}_{i_{coll}}$ with respect to the W-motion of $\mathbf{t}_{i_{coll}}$ along the base vector $\mathbf{b}_0$. This is done using the cartesian correlation of the displacement implied by the motion along the path segment itself ($\mathbf{m}_0$) and each other displacement, and according adaption of the base direction:

$$\mathbf{o}_l \;=\; \mathbf{b}_l \;-\; \frac{\mathbf{m}_0 \cdot \mathbf{m}_l}{\| \mathbf{m}_0 \|}\,\mathbf{b}_0$$

If the denominator falls below a certain numerical threshold, the directions are not orthogonalized in W, but the C-orthogonal based vectors are used in the following steps, i.e. $\mathbf{o}_l = \mathbf{b}_l$. In this case the path segment does not imply a notable motion to the control point of link i_{coll}, typically in the case of a rotational movement *around* the control point. The actual C-base vectors are a suitable alternative to use if no W-motion direction is given.

Calculate Candidate Configurations

The step size should reflect the length of the path segment. Especially when a path segment is "short", it has been created through repeated intersection of longer segments that could not have been modified, and the step size should honour this situation.

A workspace-oriented measure is preferred to a c-space measure to allow a reasonable comparison of alternative paths. To measure the motion of a single control point is not suitable, as it may not capture the motion of the complete link. A rough estimate of the cartesian motion of a link along a path segment can be found by the maximum distance of any point within the link if placed at the positions implied by the first and last configuration of the segment. This estimation does not reflect the real curved motion. For a link i, the **estimated cartesian motion** Δ_i along a path segment $\overline{\mathbf{q}_a\mathbf{q}_b}$ is defined as:

$$\Delta_i(\mathbf{q}_a, \mathbf{q}_b) \;=\; \max_{\mathbf{u}\in G_i} \| \, {}_0\mathsf{T}_i(\mathbf{q}_a)\mathbf{u} \;-\; {}_0\mathsf{T}_i(\mathbf{q}_b)\mathbf{u} \, \|$$

This measure delivers reasonable results even in the case of rotation. Note that within a realization this calculation will be replaced by the maximal distance between a few selected control points as it is too expansive to evaluate Δ_i exactly. The eight corners of a bounding box are suitable control points for a practical calculation of Δ_i.

The actual displacement d for a configuration $\mathbf{q}_a$ is chosen proportional to the length of the estimated cartesian motion along the segment $\overline{\mathbf{q}_a\mathbf{q}_b}$. The relation is expressed by a scalar factor f_{step}. This factor must be larger than zero, but should not be too large to assure a *local* modification of the path. A reasonable heuristic choice is $f_{step} = \frac{1}{2}$, although practical experiments show that other values are possible.

To keep the displacement in certain bounds, a maximum cartesian displacement $\delta_{\max}$ and a minimum displacement $\delta_{\min}$ are given as preset control parameters. If the estimated cartesian motion along the segment $\overline{\mathbf{q}_a\mathbf{q}_b}$ multiplied by f_{step} falls in the interval $[\delta_{\min}, \delta_{\max}]$, it is chosen as displacement stepwidth, otherwise the respective control parameter value is chosen:

$$d = \begin{cases} \delta_{\min} & : \quad f_{step}\,\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_b) < \delta_{\min} \\ \delta_{\max} & : \quad f_{step}\,\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_b) > \delta_{\max} \\ f_{step}\,\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_b) & : \quad else \end{cases}$$

The length d is used to calculate the candidate set $\mathbf{c}^a$ for the configuration $\mathbf{q}_a$ based on the directions $\mathbf{o}_l$. A total of $2(n_{i_{coll}} - 1)$ candidate configurations are found, as the directions $\mathbf{o}_l$ and their reverses are used. The $\mathbf{c}^a_k$, $k = 1, \ldots 2(n_{i_{coll}} - 1)$, are given with $l = 1, \ldots, n_{i_{coll}} - 1$:

$$\mathbf{c}^a_{2l-1} = \frac{d}{\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_a + \mathbf{o}_l)}\,\mathbf{o}_l + \mathbf{q}_a$$

$$\mathbf{c}^a_{2l} = -\frac{d}{\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_a + \mathbf{o}_l)}\,\mathbf{o}_l + \mathbf{q}_a$$

If the denominator falls below a numerical threshold, the respective pair of candidates is discarded, i.e. not used in the attempt of modification. This suppresses arbitrarily large C-steps. If any of the candidates result in a configuration $\mathbf{c}^a_k \notin \mathsf{C}$, the components i that are not within their allowed interval are set to $q_{i_{\min}}$ or $q_{i_{\max}}$ respectively to assure $\mathbf{c}^a_k \in \mathsf{C}$. The resulting candidate is usually neither approximately orthogonal displaced in W nor displaced by the distance d. As it is not desired to abort planning close to the border of the c-space, this joint-clipping is included in the candidate calculation.

It should be noted that this candidate calculation is dependent on the intersection configuration $\mathbf{q}_a$ and the path segment $\overline{\mathbf{q}_a\mathbf{q}_b}$. The candidates calculated for the segment adjacent in $\mathbf{q}_a$ are different if the adjacent path segment has a dif-

ferent direction (this explains the directional change in the series of displacements visualized in fig. 4.3, p. 53).

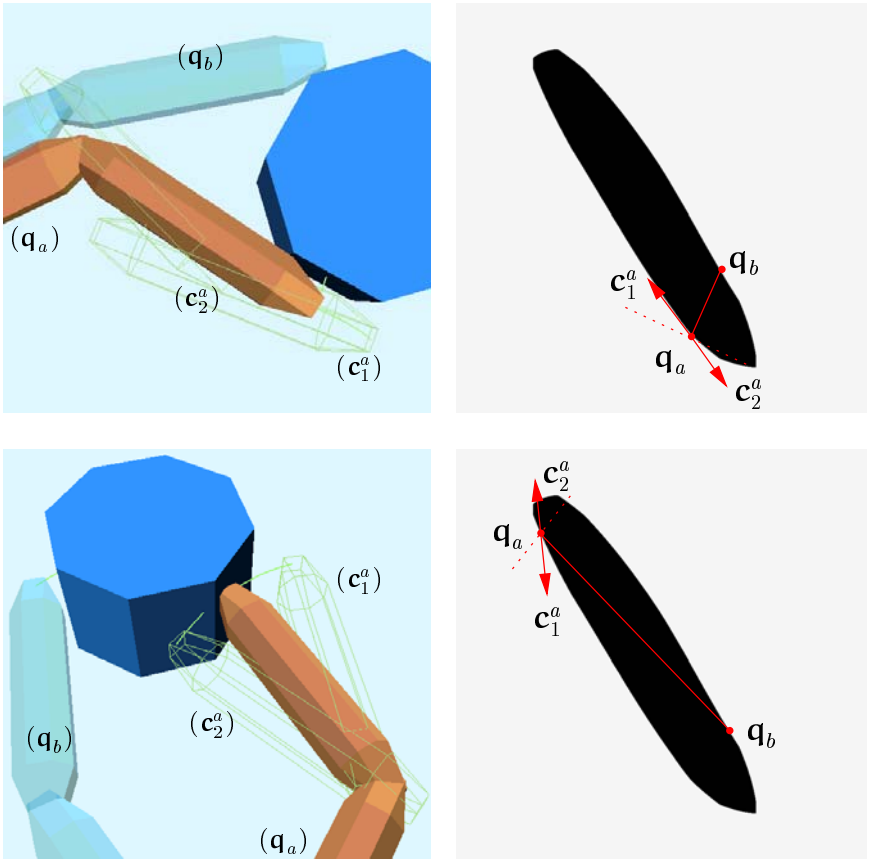

Figure 4.5: Two examples for candidate calculation in the 2-DOF example scenery. The c-space mappings show the path segments and the respective configuration $\mathbf{q}_a$. The C-orthogonal base directions are adapted to yield a workspace-orthogonal displacement of the tip point. Link two is shown at the candidate configurations on the two images on the left-hand side.

It is important to note that the W-orthogonalization is only approximate. The cartesian directions implied by the C-directions $\mathbf{o}_l$ are only *locally* orthogonal for the control point. The candidate calculation is illustrated in fig. 4.5. The "stretching" of the displacement to yield a cartesian steplength d does not explicitly consider the kinematics of the manipulator. The calculated displacement steps, i.e. the W-motions for the control point between $\mathbf{q}_a$ and $\mathbf{c}_k^a$, are usually curved in W. The passage of singularities, any difference between rotational and translational movement and other possible aspects are not considered. But for candidate calculation, this approximation has proven to be highly efficient, and as it is independent from the kinematic model of the manipulator, it works for

any link within any kind of kinematic structure. The employed heuristics are workspace-oriented. To achieve improved rating and to enable comparison of alternatives, they are highly superior to candidate creation schemes based on purely c-space concepts.

4.2.3 Replacement Selection

The modification is based on path segments, not on single intersection configurations, as it attempts to displace complete segments. If one path segment $\bar{\mathbf{P}} = \overline{\mathbf{q}_a \mathbf{q}_b}$ is to be modified, the previous segment $\bar{\mathbf{P}}_{prev} = \overline{\mathbf{q}_{prev} \mathbf{q}_a}$ and the following segment $\bar{\mathbf{P}}_{next} = \overline{\mathbf{q}_b \mathbf{q}_{next}}$ have to be regarded as well.

Candidate sets $\mathbf{c}^a$ and $\mathbf{c}^b$ are created for $\mathbf{q}_a$ and $\mathbf{q}_b$. Alternative paths can be constructed either by replacing only $\mathbf{q}_a$ or by replacing only $\mathbf{q}_b$ or by replacing both configurations. The latter case is limited, as the full combination of alternative paths would require the rating of $(2(n_{i_{coll}} - 1))^2$ combinations. Only those $2(n_{i_{coll}} - 1)$ combinations are considered where the respective candidates were calculated based on the same C-orthogonal base vectors. In the following, the candidate sets are assumed to be ordered accordingly, i.e. equally indexed candidates $\mathbf{c}_k^a$ and $\mathbf{c}_k^b$ imply a displacement into approximately the same workspace-direction, if the segment itself implies a relatively straight W-motion for link i_{coll} (see fig. 4.6 for illustration).

Alternative paths based on this pairs of candidates can shift the entire segment into one direction in the workspace, thus they are an approximation of the intended "workspace-parallel" displacement. Among all possible combinations of candidates, these pairs are chosen, as they are the most promising ones that can be identified without further tests.

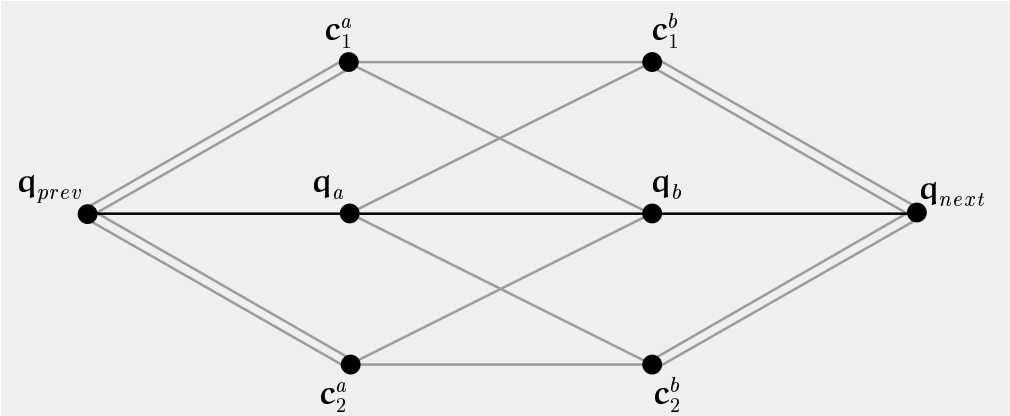

Figure 4.6: Principle illustration of the alternative paths that are considered. Each single candidate is used to form a possible alternative. To replace the whole segment, only pairs of candidates that possibly shift the whole segment in the same W-direction are considered. These pairs are equally indexed.

A best alternative path has to be chosen. Considering three segments simultaneously results in a three-dimensional decision problem. This is simplified by using only the rating of the considered inner segment $\overline{\mathbf{q}_a\mathbf{q}_b}$. The alternative path that results in the *highest increase* of the rating for $\overline{\mathbf{q}_a\mathbf{q}_b}$ and does *not decrease* the rating of the previous and the next segment is chosen to replace the original path.

The following algorithm realizes this scheme. There are three special cases that have to be handled, as both the start configuration $\mathbf{q}_{start}$ and the goal configuration $\mathbf{q}_{goal}$ cannot be modified. If the segment connects start and goal only, no modification can be applied at all, and if either start or goal are elements of the current segment, only alternatives displacing the single intersection position adjacent to start/goal are considered.

Algorithm : **ModifySegmentScale**
Input : $\overline{\mathbf{q}_a\mathbf{q}_b}$
Output : FAILURE or SUCCESS

$bestrating := Q_{scale}(\overline{\mathbf{q}_a\mathbf{q}_b})$ *[A better rating is searched]*

if $(\mathbf{q}_a = \mathbf{q}_{start} \wedge \mathbf{q}_b = \mathbf{q}_{goal})$ **return** FAILURE

if $(\mathbf{q}_a \neq \mathbf{q}_{start})$ *[$\mathbf{q}_b$ may be $\mathbf{q}_{goal}$]*
  Create Candidates $\mathbf{c}^a$
  for $k = 0 \ldots 2(n_{i_{coll}} - 1)$ *[alternatives replacing $\mathbf{q}_a$ only]*
    if ($Q_{scale}(\overline{\mathbf{q}_{prev}\mathbf{c}^a_k}) \geq Q_{scale}(\overline{\mathbf{q}_{prev}\mathbf{q}_a}) \wedge$
        $Q_{scale}(\overline{\mathbf{c}^a_k\mathbf{q}_b}) > bestrating$)
      $bestrating := Q_{scale}(\overline{\mathbf{c}^a_k\mathbf{q}_b})$; $\mathbf{q}^*_a := \mathbf{c}^a_k$; $\mathbf{q}^*_b := \mathbf{q}_b$
    endif
  endfor
endif

if $(\mathbf{q}_b \neq \mathbf{q}_{goal})$ *[$\mathbf{q}_a$ may be $\mathbf{q}_{start}$]*
  Create Candidates $\mathbf{c}^b$
  for $k = 0 \ldots 2(n_{i_{coll}} - 1)$ *[alternatives replacing $\mathbf{q}_b$ only]*
    if ($Q_{scale}(\overline{\mathbf{q}_a\mathbf{c}^b_k}) > bestrating \wedge$
        $Q_{scale}(\overline{\mathbf{c}^b_k\mathbf{q}_{next}}) \geq Q_{scale}(\overline{\mathbf{q}_b\mathbf{q}_{next}})$)
      $bestrating := Q_{scale}(\overline{\mathbf{q}_a\mathbf{c}^b_k})$; $\mathbf{q}^*_a := \mathbf{q}_a$; $\mathbf{q}^*_b := \mathbf{c}^b_k$
    endif
  endfor
endif

if $(\mathbf{q}_a \neq \mathbf{q}_{start} \ \wedge \ \mathbf{q}_b \neq \mathbf{q}_{goal})$ *[general case]*
  for $k = 0 \ldots 2(n_{i_{coll}} - 1)$ *[alternatives replacing $\mathbf{q}_a$, $\mathbf{q}_b$]*
    if ($Q_{scale}(\overline{\mathbf{q}_{prev}\mathbf{c}_k^a}) \geq Q_{scale}(\overline{\mathbf{q}_{prev}\mathbf{q}_a}) \wedge$
      $Q_{scale}(\overline{\mathbf{c}_k^a\mathbf{c}_k^b}) > bestrating \wedge$
      $Q_{scale}(\overline{\mathbf{c}_k^b\mathbf{q}_{next}}) \geq Q_{scale}(\overline{\mathbf{q}_b\mathbf{q}_{next}})$)
      $bestrating := Q_{scale}(\overline{\mathbf{c}_k^a\mathbf{c}_k^b}); \quad \mathbf{q}_a^* := \mathbf{c}_k^a; \quad \mathbf{q}_b^* := \mathbf{c}_k^b$
    endif
  endfor
endif

if $(bestrating > Q_{scale}(\overline{\mathbf{q}_a\mathbf{q}_b}))$
  Replace $\mathbf{q}_a$by $\mathbf{q}_a^*$, $\mathbf{q}_b$by $\mathbf{q}_b^*$
  return SUCCESS *[Path improved]*
endif

return FAILURE *[No better alternative found]*

4.2.4 Path Refinement

If no alternative path can be found for the current sequence of path segments, new intersection configurations have to be created, dividing segments into several parts. This subdivision of a path segment $\overline{\mathbf{q}_a\mathbf{q}_b}$ is based on the location of minimal rating $\mathbf{q}_{min}$ that is found when rating the segment. Using this additional information increases the spatial adaptability of the path with respect to the real obstacle topology. But it is not a good choice to cut exactly at $\mathbf{q}_{min}$. This would result in two equally bad rated segments, and if none of them can be modified, no further subdivision is possible, as their location of minimal rating is now either the first or last configuration of the path segment.

Generally, the subdivision should not create segments that are too short, especially if segments are repeatedly divided. Therefore, the longer part of the segment with respect to $\mathbf{q}_{min}$ is divided, i.e. a new intersection position is entered either in between $\mathbf{q}_a$ and $\mathbf{q}_{min}$ or in between $\mathbf{q}_{min}$ and $\mathbf{q}_b$. A constant factor $f_{divide} \in [0, 1]$ is chosen for this subdivision. There is no further knowledge available beside the configuration $\mathbf{q}_{min}$ to guide this subdivision, and assuming that the segments should be of comparable length, $f_{divide} = \frac{2}{3}$ is a reasonable heuristic choice. If segments cannot be modified and the subdivision is repeated, this mechanism creates a stepwise shortened "inner" segment that containes $\mathbf{q}_{min}$. The principle is illustrated in fig. 4.7.

There has to be a lower limit for the length of path segments, otherwise the subdivision might continue *ad infinitum*. To keep the number of control parameters low, the smallest cartesian steplength $\delta_{\min}$ – used within the candidate creation above – is also used. If the estimated cartesian motion falls below this

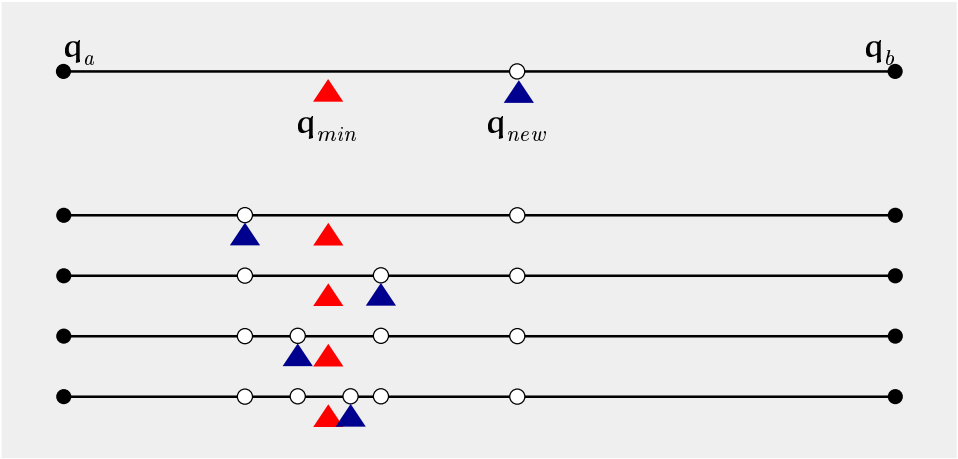

Figure 4.7: Principle path refinement. Top: The longer part with respect to the configuration of minimal rating is divided. Bottom: In case of repeated intersection, the configuration of minimal rating is part of a segment of decreasing length.

value, $\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_b) < \delta_{\min}$, no intersection takes place. This is a reasonable choice, as it keeps the segment "length" and the candidate displacement in the same order, thus $\delta_{\min}$ controls the "fine resultion" of the planning process.

Algorithm : **DivideSegmentScale**
Input : $\overline{\mathbf{q}_a \mathbf{q}_b}$
Output : FAILURE or SUCCESS

if $(\Delta_{i_{coll}}(\mathbf{q}_a, \mathbf{q}_b) < \delta_{\min})$ *[Segment is too short]*
  return FAILURE
endif
$\mathbf{q}_{min}$:= (Location of minimal rating)
if $(\| \mathbf{q}_{min} - \mathbf{q}_a \| > \| \mathbf{q}_b - \mathbf{q}_{min} \|) \vee (\mathbf{q}_a = \mathbf{q}_{start} \wedge \mathbf{q}_b = \mathbf{q}_{goal})$
  $\mathbf{q}_{new} := \mathbf{q}_a + f_{divide}(\mathbf{q}_{min} - \mathbf{q}_a)$
  Insert $\mathbf{q}_{new}$ into path, creating 2 colinear segments
  $segnum := segnum + 1$
endif
if $(\| \mathbf{q}_{min} - \mathbf{q}_a \| \leq \| \mathbf{q}_b - \mathbf{q}_{min} \|) \vee (\mathbf{q}_a = \mathbf{q}_{start} \wedge \mathbf{q}_b = \mathbf{q}_{goal})$
  $\mathbf{q}_{new} := \mathbf{q}_{min} + (1 - f_{divide})(\mathbf{q}_b - \mathbf{q}_{min})$
  Insert $\mathbf{q}_{new}$ into path, creating 2 colinear segments
  $segnum := segnum + 1$
endif
return SUCCESS

One special case is considered within the subdivision algorithm. If the current segment is the initial path between start and goal, both parts are divided.

This results in three segments with the inner one being the one with the worst rating, which allows to possibly displace a complete segment and not only single intersection configurations right at the beginning of the planning process.

4.2.5 Planning Algorithm

The developed components are integrated into one planning algorithm that attempts to plan a complete collision-free path, i.e. a path where any segment is rated with n. The algorithm should try to limit the overall effort. It is especially futile to work extensively on one part of the path if there is another part that cannot be modified successfully and the whole planning process has to be terminated with a failure. Therefore, the planning algorithm focusses its attention on the segment that is rated worst.

Algorithm : **PlanScale**
Input : $\mathbf{q}_{start}$, $\mathbf{q}_{goal}$
Output : FAILURE or SUCCESS

$\bar{\mathbf{P}}_1 := \overline{\mathbf{q}_{start}\mathbf{q}_{goal}}$
$segnum := 1$
while (Any segment is rated $< n$)
  $worst =$ (Number of worst segment)
  if (**ModifySegmentScale**($\bar{\mathbf{P}}_{worst}$) = SUCCESS)
    $backward := worst - 1$
    while $(backward > 0) \wedge (Q_{scale}(\bar{\mathbf{P}}_{backward}) \neq n)$
      ModifySegmentScale($\bar{\mathbf{P}}_{backward}$)
      $backward := backward - 1$
    endwhile *["Planning wave" backward]*
    $forward := worst + 1$
    while $(forward \leq segnum) \wedge (Q_{scale}(\bar{\mathbf{P}}_{forward}) \neq n)$
      ModifySegmentScale($\bar{\mathbf{P}}_{forward}$)
      $forward := forward + 1$
    endwhile *["Planning wave" forward]*
  else *[Worst segment not modified]*
    if (**DivideSegmentScale**($\bar{\mathbf{P}}_{worst}$) = FAILURE)
      return FAILURE *[No further activity possible]*
    endif
  endif
endwhile *[Any colliding segment exists]*
return SUCCESS

The planning scheme works in a loop that is repeated until no more modification is possible – either because a solution has been found or because the worst segment cannot be divided, i.e. has fallen below the length threshold $\delta_{\min}$. Within

each iteration it is attempted to modify the segment with minimal rating. If this succeeds, the planner "walks" backward and forward along the path and tries to modify all neighbouring segments. These "planning waves" are terminated either by reaching the start/goal configuration or a segment that is already collision-free. This mechanism is included to allow the polygonal path to adapt its shape in the region of collision, i.e. it allows the neighbouring segments to "follow" the worst segment on its "way" out of the obstacles. After that, the main loop starts all over again, locating the worst segment and so on.

If the worst segment cannot be modified, no "planning wave" is executed, but the worst segment is divided and the loop is repeated if the subdivision is possible, otherwise planning fails.

4.2.6 Termination

The planning scheme has a monotone behaviour and is therefore *practically* guaranteed to terminate. Within each iteration of the planning loop it is assured that no segments' rating decreases. If the worst segment is cut, no segment can get any worse, and if the segment is shifted, it gets a better rating. The subdivision of the worst segment is limited by a length threshold. This implies that one of the termination conditions – SUCCESS or FAILURE – is reached.

But this does not formally prove termination. It might happen that repeated intersection and modification does not improve the worst rating of the path, if there are two adjacent segments with the same worst rating. It might as well happen that the improvement within each step gets infinitesimal small and the main loop behaves asymptotically. As cases like this were not experienced in the practical tests, no additional termination mechanisms appear necessary. To enforce termination in all cases, the number of main planning loop iterations is limited to a value practically never reached (order of 1000). Other ways to ensure termination are possible, e.g. by limiting the number of path segments or by limiting the running time of the planner.

4.3 Planning Examples

To illustrate the properties and behaviour of the developed scheme in realistically complex sceneries, the planning for the two sample tasks introduced in 3.6, p. 43, is presented in detail.

The 6-DOF Manipulator

The 6-DOF manipulator has to lift a lengthy part from the floor back onto a table (see fig. 3.4, p.44). The size of the robot, stretched to full length, is approximately $1\,m$, the load has a length of $25\,cm$. The parameters chosen for the example are $\delta_{\max} = 10\,cm$ and $\delta_{\min} = 5\,mm$. A displacement of a maximum of $10\,cm$ is a reasonable step in the workspace of the manipulator. A stepsize below $5\,mm$

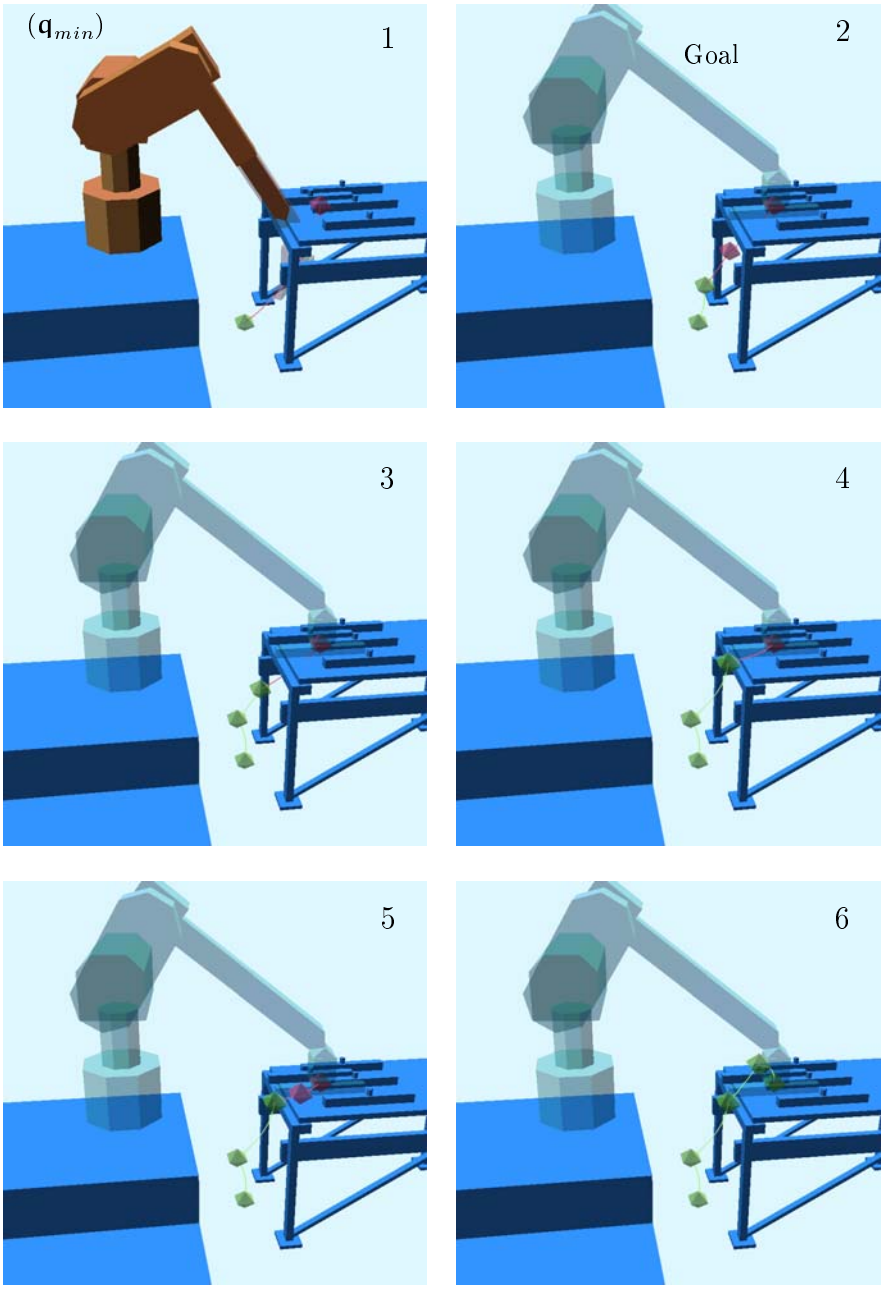

Figure 4.8: Planning for the 6-DOF manipulator. 1: The configuration $\mathbf{q}_{min}$ within the initial straight path and the respective scaling of link four. 2–5: Intermediate stages of the planning process. For clarity, only the tip trace and the goal posture it shown. 6: The collision-free path found as solution.

does not appear to be useful, as it would require a lot of steps to get out of collision, thus it is a reasonable termination threshold.

Planning starts with the rating of the straight line connection. The configuration of worst rating $\mathbf{q}_{min}$ for the initial path is shown in fig. 4.8.1. Link four is scaled, the wrist and the load are not considered. The rating of the initial path is approximately 3.6. The initial path is cut into three colinear segments, and as $\mathbf{q}_{min}$ is situated approximately in the center between start and goal, the segments are roughly of equal length. The segment adjacent to start is collision-free.

Two iterations of the mainloop (fig. 4.8.2 and 4.8.3) move the middle segment out of collision. The "planning wave" towards the goal is not successful in these two steps. The goal configuration is very close to collision, and no modification attempt results in a "lift" up from the surface of the table.

Now the last segment is the only one available for modification, and two successful steps can be applied (fig. 4.8.4). The improvement is not very obvious in the picture, but it is important as the load is now aligned with the edge of the table until it has reached the niveau of the goal position. No candidate can be found to achieve the required 90 degree turn of the load *and* the slight lift above the other parts that lie on the table. Further adaption is necessary, the segment is divided (fig. 4.8.5).

Two further small modification steps (the segment length is used to calculate the displacement step size in this case as it is shorter than $2\delta_{\max} = 20\,cm$) are sufficient to finally find a collision-free path. **ModifySegmentScale** was called just 8 times and a total of 94 candidates were calculated to find this solution.

The resulting motion is visualized in fig. 4.9. Just four linear segments in the six-dimensional space were found to realize this difficult motion. The part is lifted up, turned above the edge of the table and placed there, almost touching, with its back end first and then tilted down.

This demonstrates a typical feature of the BB-method: to "adjust" the path among the obstacles without requiring many intersection configurations. High dimensional spaces, although cluttered with obstacles or containing tight passages, are usually passed with a small number of well placed configurations. This becomes even more obvious for the 16-DOF manipulator.

The 16-DOF Manipulator

The 16-DOF manipulator has to move out of a rather tight gate. The full extent of the snake-like arm is about $2\,m$, the diameter of its links is $12\,cm$. The gate has an inner width of $32\,cm$ and a height of $50\,cm$. The parameters chosen for this example are $\delta_{\max} = 20\,cm$ and $\delta_{\min} = 2\,cm$. This is motivated by the size of the robot and the obstacles: larger displacements do not appear promising, as they might "push" the entire robot through both jambs of the gate. And another aspect is important: the displacement is calculated for the link in collision, joints beyond i_{coll} are not modified. This could result in very large displacements for

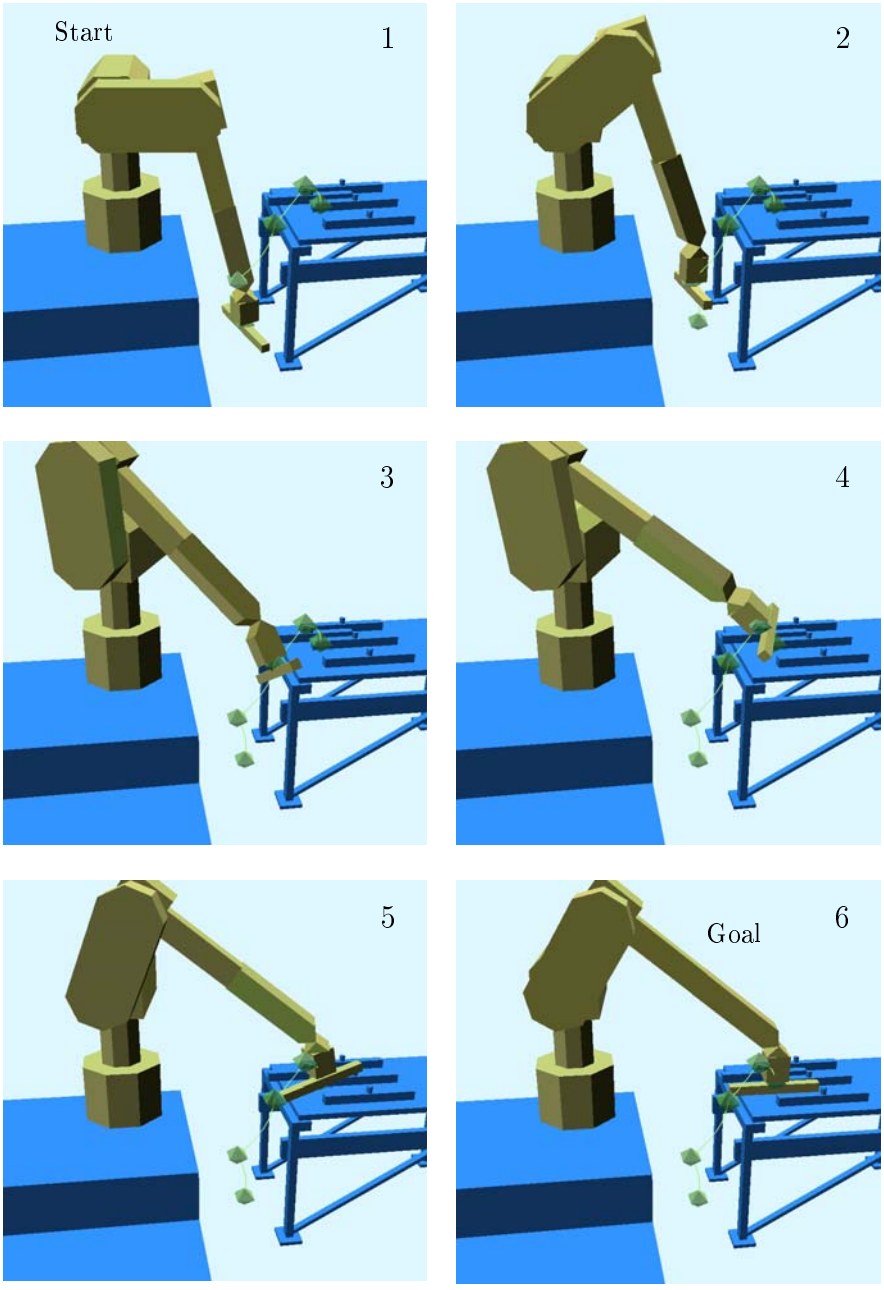

Figure 4.9: A sequence of configurations along the planned path. 1: Start. 2–5: Intermediate postures, the load is lifted up and turned. 6: Goal.

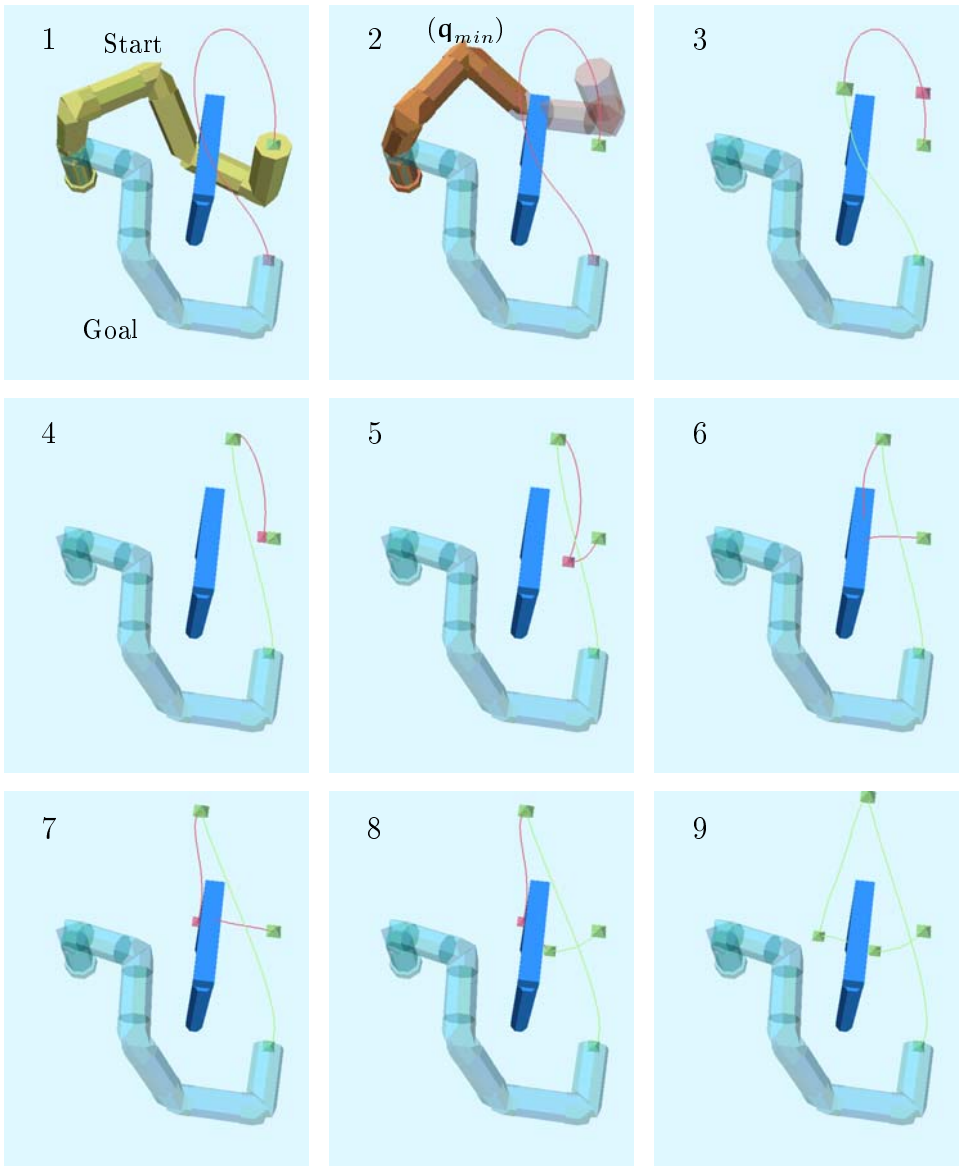

Figure 4.10: Planning for the 16-DOF manipulator. 1: Start and goal and the tip trace of the colliding initial path, the C-straight connection in the 16-dimensional space. 2: The configuration $\mathbf{q}_{min}$ within the initial path and the respective scaling of link eleven. 3–8: Intermediate stages of the planning. For clarity, only the tip trace and the goal posture is shown. 9: The collision-free path found as solution, made up of just four linear segments.

outer links if the effective radius of an inner link is high and a large value is chosen for the maximum displacement.

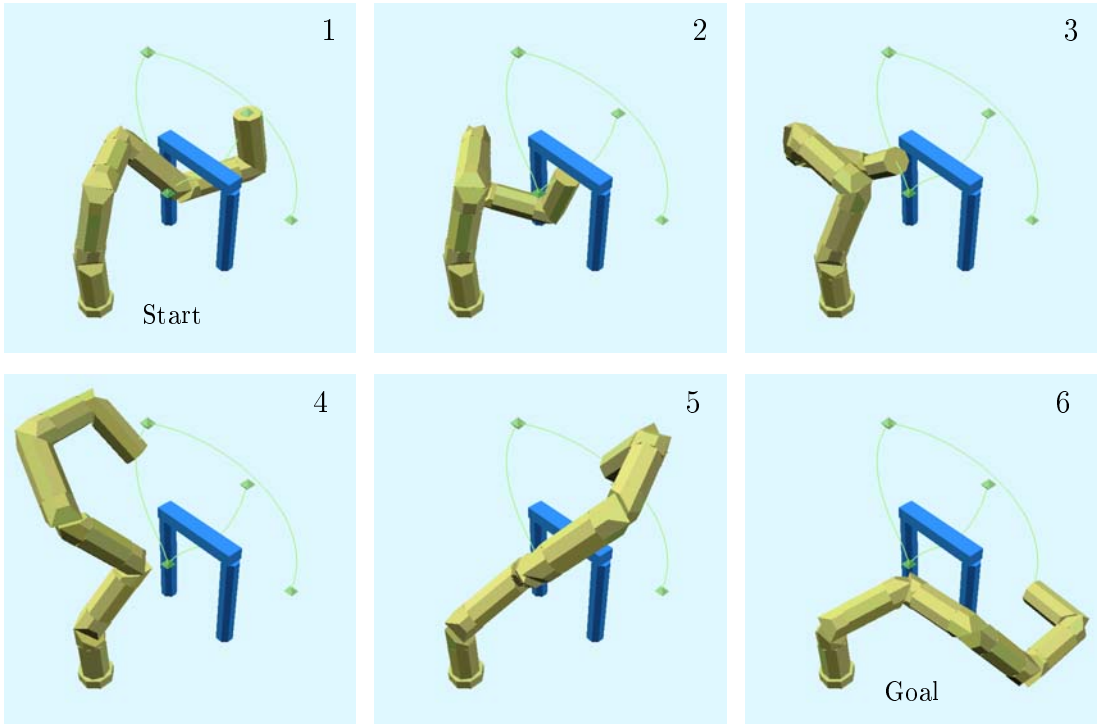

Figure 4.11: A sequence of configurations along the planned path. 1: Start. 2–5: Intermediate postures. The manipulator retracts through the gate and turns to the goal. 6: Goal configuration is reached.

The initial straight path results in a very curved tip trace (see fig. 4.10.1). It is rated, $\mathbf{q}_{min}$ is found relatively close to start, and the colliding scaled link is link number eleven (4.10.1). Three new segments are created by subdivision, and the last of these segments is found to be collision-free (4.10.3). A sequence of five modification steps is successfully applied to the inner path segment. These modifications are the result of attempts to modify the worst rated segment and of the "planning wave" that is executed to modify the segment adjacent to start (4.10.4 through 4.10.7).

In this stage, the first segment has become the worst rated segment, as link 16 is in "deep intersection" with the upper bar of the gate. No successful modification step can be found, as either the segment itself or the middle segment decreases in rating for all alternative paths created W-orthogonal for the intersection configuration adjacent to start. The segment is therefore subdivided, and one modification step is sufficient to get both new segments out of collision

(4.10.8). A final modification of the last colliding segment yields a collision-free path (4.10.9). **ModifySegmentScale** was called 7 times and a total of 252 candidates were created to find a collision-free path in the sixteen-dimensional space. It is made up of just four linear segments. Although fewer modifications were applied, the number of candidates is much larger than in the 6-DOF example due to the larger number of joints.

The motion is sketched in fig. 4.11. The manipulator collapses spiral-shaped to pass through the gate, and reaches the goal in a wide turn. The latter part is quite similar to the initial straight motion, planning mainly modified the first part where the obstacle had to be avoided. As in the 6-DOF example, the number of intersection configurations is very low, just three between start and goal.

4.4 Failure of Local Planning

First of all, it should be noted that the local planning is practically very robust and failure is rare – the difficult examples above were solved easily. The whole planning scheme is built upon heuristics that were developed to minimize the possible failures, but nonetheless every local, incomplete planning scheme will fail in certain situations. The local planning of the BB-method fails, if the worst rated path segment can neither be subdivided nor improved. Three main situations can be identified to result in such a failure.

4.4.1 "Local Maximum" of the Rating Function

A local maximum of the rating function is given if no slight change in the first or last configuration of the rated path segment is able to increase the rating. Such a situation is not immediately recognized by the planning scheme, but results in failure when the segment is subdivided repeatedly.

Figure 4.12 illustrates a local maximum for a 2 DOF example. The manipulator has to extend itself to its full length along the motion from the start to the goal configuration. This can only be achieved if it leaves the L-shaped obstacle, but local path modification is impossible as the segments would have to be shifted through a region of worse rating. This kind of failure is no consequence of the heuristic construction of the planning scheme, but an inherent rating maximum that cannot be locally escaped.

In general, some kind of "local maximum" of the rating function prevents successful path planning if the current path is rated higher than the intermediate stages of planning of a suitable "detour". This rarely occurs for robots with several degrees of freedom, and it is worth considering why:

The rating function virtually retracts the robot from the obstacles. Modifiying the segments to improve the rating replaces this *virtual* retraction by a *real* one, i.e. the joints are bent in such a way that the robot is positioned out of the obstacles in all possible configurations along the path. If there are enough links

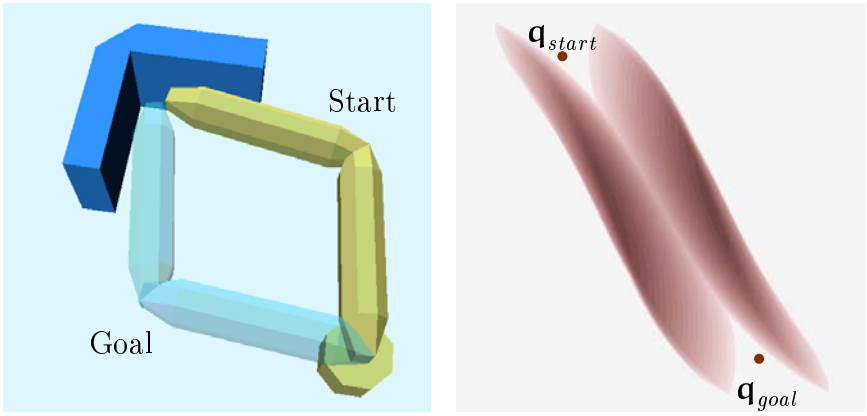

Figure 4.12: A possible local maximum for the 2-DOF manipulator. Left: The L-shaped obstacle requires a "detour" to enable the stretching of the arm. Right: The visualized rating shows that local modification cant yield a "better" path.

prior to the currently colliding link, the planning is able to retract the physical robot far enough – unless failure situation 2 occurs:

4.4.2 "Loss" of Degrees of Freedom

The retraction of the robot's path out of the obstacles can fail if the combination of joints, links and obstacles reduces the adaption ability in the joints in a way that not all the joints $I_{i_{coll}}$ can be used successfully. Two reasons can be identified for such a situation:

- **Joint limits reached**
  The retraction of the path segments out of the obstacles may be blocked if a lower joint reaches its limit and cannot be twisted further to allow an increasing adaption. The remaining joints may not be sufficient for further expansion, the planning process fails in a kind of "local maximum" at the border of C. Such a failure is a result of the physical limitations of the robot that can hardly be considered by the local path modification. The improving steps that lead to this situation may have started "far away" from the joint limits, ever increasing until the dead end is reached. The situation is illustrated in fig. 4.13.
- **Lower links are "blocked"**
  It may occur that the number of joints practically available for the required retraction is reduced, and the planning does not take that into account. This happens if a link j between the current link i_{coll} and the base is allmost in touch with an obstacle along a path segment. All alternatives that possibly allow a larger model of the link i_{coll} to move may push link j into collision. For a proper retraction of link i_{coll}, only the links between j and i_{coll} are

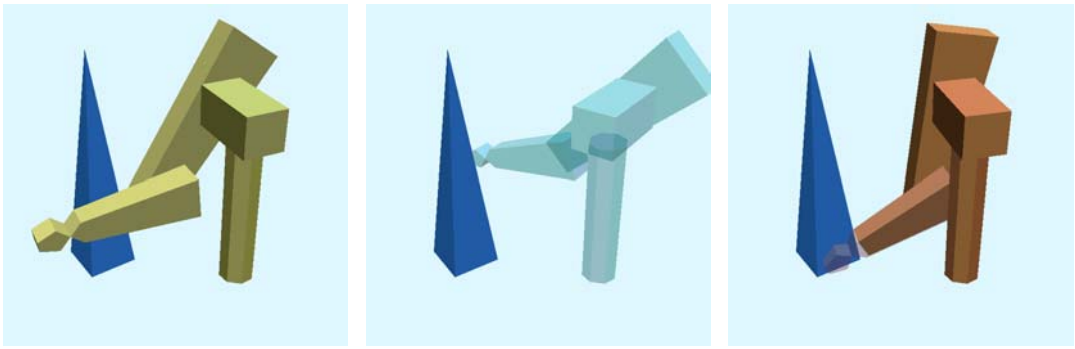

Figure 4.13: Failure of local planning for a 6-DOF manipulator. The task is to move from one side of an obstacle to the other side (left and center). The highest rating increase is achieved by moving downwards, i.e. turning link 2. This reaches its lower limit, and link 3 cannot be turned up again, as this would decrease the rating (right).

actually available. This is not regarded in the candidate calculation and may result in failure, see fig. 4.14 for an illustrating example. Such a "dead end" of local planning is a result of the heuristic mechanisms employed and not inherent in the rating and modification approach. The candidate calculation only creates a small limited number of representatives out of the continuous space of possible alternatives.

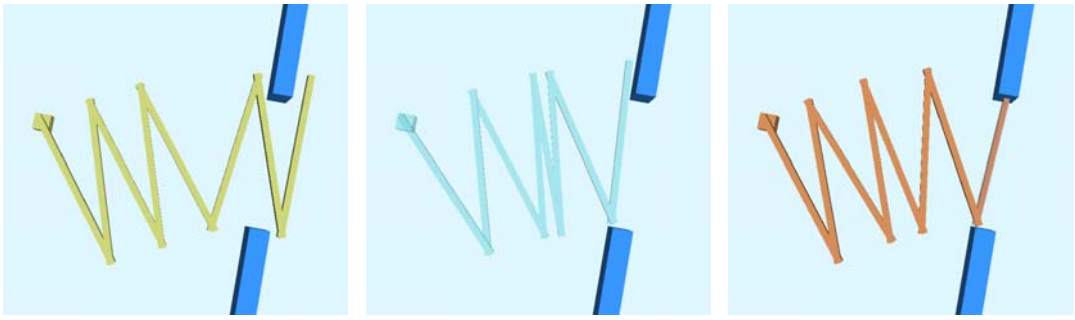

Figure 4.14: Failure of local planning for an 8-DOF planar manipulator. If it is attempted to move link 8 approximately parallel through an opening (left and center), every alternative path will either push link 8 further in collision or link 7 collides with the obstacle, thus no solution can be found. The configuration of worst rating is shown in the right image.

In the latter situation a possible solution may be found if the chosen displacement applied to link i_{coll} is very small in each step. This leads to the third possible reason of failure:

4.4.3 Parameter Selection

The planning process is dependent on the two control parameters $\delta_{\min}$ and $\delta_{\max}$. They guide the candidate creation and the segment length, and an improper choice may result in failure situations.

- **Lower displacement limit** $\delta_{\min}$
  If this parameter is chosen too large and the obstacle topology requires a fine adaption of the path, it may be impossible to find a solution that can possibly be planned with a smaller value for $\delta_{\min}$.
- **Upper displacement limit** $\delta_{\max}$
  If this parameter is chosen too large, the segments may really get "thrown around" in the c-space instead of being displaced to locally increase the rating. The planning may thus get trapped in some "dead end" that would not have occured if smaller steps had been used.
  Another possible effect of $\delta_{\max}$ is implied by its influence on the "direction" of the planning. In a certain situation, completely different segment displacements may be the best possible modification for different values of $\delta_{\max}$. Once just one modification step has altered the path in a different fashion, the whole planning process will run completely different and may either run into failure or success. This kind of unsteadiness can also be observed for $\delta_{\min}$.

All other parameters that are embedded in the heuristic scheme – f_{divide} used in the subdivision, f_{step} used for the relative step size, the candidate calculation in the approximate W-orthogonal plane – may also contribute to failure in certain situations.

4.5 Global Planning with Random Subgoals

To overcome the failure of local planning, a random exploration scheme is employed. The experimental results with the ZZ-method (Baginski (1998), see also 2.7, p. 24) have shown that – even with a much less sophisticated local planning scheme – most tasks that cannot be solved locally are solved with just one out of a few subgoals. If a solution cannot be found in this "easy" way, a lot of subgoals are created and the local planner is started again and again, but planning success is doubtful. These results influenced the developement of the global planning component of the BB-method.

The creation of a subgoal graph with its number of edges quadratic in the number of the subgoals does not appeer promising, especially with such a powerful local planner. The evaluation of many edges increases the effort a lot, without proportionally increasing the success rate. A better choice is to completely give up the planning if no solution can be found with just one subgoal out of a limited number.

Based on this consideration, the following very simple global planner is developed. If start and goal cannot be connected with local planning, a random subgoal $\mathbf{q}_{random} \in \mathsf{C}_{free}$ is created and it is attempted to plan a path from $\mathbf{q}_{start}$ to $\mathbf{q}_{random}$. If this succeeds, a second path is planned between $\mathbf{q}_{random}$ and $\mathbf{q}_{goal}$. If this also succeeds, a solution is found, otherwise a new subgoal is created. The maximum number of subgoals tested is a predefined parameter z_{random}. This planning algorithm minimizes the number of calls of the local planning.

Algorithm : **PlanGlobal**
Input : $\mathbf{q}_{start}$, $\mathbf{q}_{goal}$
Output : FAILURE or SUCCESS

```
if (PlanScale(q_start, q_goal) = SUCCESS)
  return SUCCESS                          [ Local Solution found ]
endif
for 1 ... z_random
  Create randon subgoal q_random ∈ C_free
  if (PlanScale(q_start, q_random) = SUCCESS)
    if (PlanScale(q_random, q_goal) = SUCCESS)
      return SUCCESS                      [ Solution with subgoal found ]
    endif
  endif
endfor
return FAILURE                            [ Planning for task failed ]
```

If this global exploration scheme fails, the complete planning task is terminated with failure. This planning is incomplete as the local planning: if no solution is found for the given task, it cannot be decided whether a solution exists. But the probability to find a solution is increased very much. In practical tests, z_{random} in the order of 10 is completely sufficient to find a solution to a task that failed locally, and in most cases, the first or second subgoal is successful.

4.6 Complexity and Computational Effort

A formal analysis of heuristic or randomized planning algorithms is inherently extremely difficult. Attempts to analyse the random subgoal graph (see Kavraki et al. (1996)) or the RPP-planner (see Lamiraux and Laumond (1996)) have shown that reasonable statements are only possible with non-realistic restrictions and assumptions. But nonetheless, some basic facts can be deduced to qualify the planning algorithm.

Complexity of Local Planning

The input of path planning is made up of two configurations, the input size is measured in n, the number of joints respectively the dimensionality of the c-space. The calculation of the orthogonal bases and the handling of up to n direction vectors with n components is of quadratic complexity in n, i.e. $O(n^2)$. But the constant factor that applies to these calculation steps is so small that even for n in the order of 30 or 40, a reasonable upper limit for real robots, the expected computation time is in the order of milliseconds. This is dominated by far by the effort that is required to rate the alternative paths. The number of alternative paths rated within each step is $6(n-1)$ at maximum (within **ModifySegmentScale**), therefore the effective complexity of one path modification step is approximately linear in n.

The number of modification steps is independent from n by construction, and no other component of the planning scheme but the modification function is in any way related to the number of joints. Thus, the overall effective complexity of the local planning is $O(n)$. This result is to be seen in comparison to the exponential complexity of **GMP**. And even if the quadratic term of the candidate calculation and handling gets dominant for imaginable robots with a number of joints in the order of hundreds, this planning scheme remains applicable.

The memory requirements for local planning are very small: for each path segment only the configurations, the rating, the position of minimal scaling $\mathbf{q}_{min}$ and few additional organizational data has to be stored. No previous stages of the planning are stored, just the current path and the considered alternatives for one segment.

Complexity of Global Planning

The complexity of the global planning is solely dependent on z_{random}, the maximum number of subgoals, and the complexity of the local planning. As z_{random} is predefined constant, collision-free planning as a whole is linear in the number of joints.

Global planning does not require additional memory, as the data stored for subtasks that resulted in failure can be deleted immediately.

Computational Effort

The scheme was created to plan paths with high efficiency, i.e. with low computational effort. The most expensive part in terms of computing time is the evaluation of the rating function, as it involves geometry model intersection and maximization. The complexity and the computational effort of the rating function are considered in conjunction with the suggested realization, see section 7.5, p. 122.

To be efficient, local planning attempts to minimize the rating effort. This is successfully achieved by:

- **Keeping the number of modification steps small**
  No fixed intersection scheme is used to plan with a constant number of path segments or with path segments of a fixed length. Instead, it is attempted to solve the task with as few (long) segments as possible. Of course, longer segments are more expensive to rate, but it is overall much more efficient to modify a few long segments. The actually required number of modification steps cannot be predicted. It depends on the robot, the task and the obstacle configuration. More modification steps are required if the robot has to be moved out of very "deep" collisions, and more modification steps are necessary if the path has to be divided again and again to adapt to the obstacles. Thus it is reasonable to say that the planning effort is dependent on the "task difficulty". This is a valuable property as many realistic tasks are "easy" and therefore quickly solved.
- **Maximizing the possible rating increase within each modification step**
  The approximate cartesian displacement has proven to yield well suited alternative paths. This reduces the overall effort, as the paths are "pushed out" of the obstacles very directly, requiring only a small number of modification steps. If more joints precede the currently colliding joint i_{coll}, the possibility to gain better ratings for alternative paths is increased. This implies that a task might require *less* modification steps if the robot has *more* joints. Although the number of candidates created within each step is increased, the overall effort may even be *reduced* or remain constant due to the reduced number of steps, and does not necessarily grow with the number of joints. This is a consequence of the approach: if the robot has more joints, it is easier to retract out of collision.
- **Fast failure, if a task cannot be solved**
  The worst segment is focussed by the planner, and if it cannot by modified, the location of worst rating is captured by smaller and smaller segments. All effort is concentrated on this location, and unavoidable failure is quickly detected.

The computational effort is also strongly influenced by the control parameters $\delta_{\min}$ and $\delta_{\max}$. Apart from their possible effect on the planning success, as discussed above, different values also result in different planning times. But this does not happen in a linear fashion: A small value for $\delta_{\max}$ might result in faster planning if the path has to be modified only slightly. On the other hand, repeated intersection is required to yield sufficiently small displacement steps when $\delta_{\max}$ is large. There exists no perfect choice for the parameters, they have to be chosen heuristically. But the empirical results show that planning is successful and efficient for a wide range of possible values, and that within a scenery one set of parameters is sufficient to solve arbitrary tasks.

Chapter 5

O2 – Planning Safety Distances

Safety distances are important for collision-free paths only. They appear necessary to allow for modelling and control uncertainties. Within the BB-method, they are planned as a kind of continuation of the collision-free path planning, using quite similar components. As opposed to collision-free path planning, where typically the path is modified to a large extent, the desired safety distance is quite small in relation to the size of the robot, and the path is adapted with smaller steps and only slightly modified.

Informally speaking, the robot is "blown up" beyond its original size, and the path is "bent" to let the extended robot model pass.

5.1 Rating Function Q_{dist} – Expanding the Robot

5.1.1 Intention and Approach

The required safety distances for the individual links are a task parameter. They form an n-dimensional vector $\mathbf{d}_{\max}$. In the 2 DOF example, safety distances have to be planned for link 2 only. To rate the safety distance, isotropically expanded geometry models G^{+d} are used instead of scaled geometry models.

The case of rating a single configuration is considered first. Different values *dist* can be calculated for collision-free configurations by maximizing the possible collision-free expansion of link 2. An upper threshold for the rating is given by the desired safety distance $d_{\max_2}$. Any configuration that is further away than this desired safety distance is rated with $d_{\max_2}$:

$$dist \;=\; \max_{d\in[0,d_{\max_2}]} \Big({}_0\mathbf{T}_2(\mathbf{q})\, G_2^{+d} \cap G_0 \;=\; \emptyset \Big)$$

The expansion gives an absolute rating of the distance between the collision-free link and the closest obstacle, see fig. 5.1 for illustration. This can be extended to complete manipulator systems moving along path segments.

To use the planning scheme of rated path modification, a one-dimensional rating function has to be defined that captures the safety distance along a path segment. But possibly there are different links in different distances, and a com-

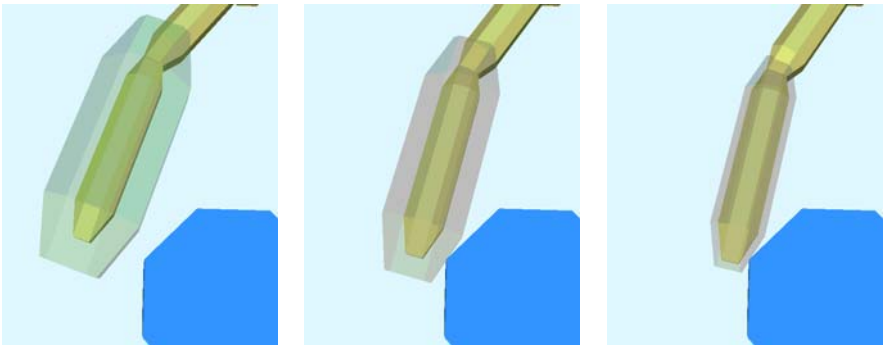

Figure 5.1: Rating by expansion. Link 2 is expanded. If it does not collide for the upper threshold $d_{\max_2}$, this is used as rating value (left). Different values are found by expanding up to the largest possible collision-free size (center and right).

parison could turn into a multi-dimensional decision problem. Therefore, only one dedicated link i_{act} is considered for the rating at a time.

5.1.2 Rating a Path Segment

The intention of the rating is to enable the comparison between alternative path segments $\bar{\mathbf{P}}_{alt}$ and segments $\bar{\mathbf{P}}$ within a path. To reduce this rating to the comparison of the possible expansion of one link i_{act}, it is necessary to supply the rating function with the safety distances of all other links for path segment $\bar{\mathbf{P}}$ as the alternative segment $\bar{\mathbf{P}}_{alt}$ is rated. The alternative path segment has to keep the same distances for all links but i_{act} and should allow for a larger safety distance for link i_{act}. In this case, it is a "better" alternative, and worth to be considered to replace $\bar{\mathbf{P}}$.

This information is kept in the vector $\mathbf{d}_{\bar{\mathbf{P}}}$. Each component $d_{\bar{\mathbf{P}}i}$ stores the currently achieved safety distance for the respective links. If the alternative path segment collides in any link i with the applied expansion $d_{\bar{\mathbf{P}}i}$, the rating function returns the symbolic value COLLIDING to indicate that the alternative segment is worse than the current segment. If it does not collide, the link i_{act} is expanded to the maximum possible size in the interval $[d_{\bar{\mathbf{P}}i_{act}}, d_{\max_{i_{act}}}]$ that can be applied to the complete segment, i.e. it is the largest expanded model that is suitable to move without collision. This is returned as the comparative rating for the path segment $\bar{\mathbf{P}}_{alt}$.

Note that the problems that might occur when *scaling* non-convex objects do not arise when models are expanded (see 4.1.3, p. 50). Isotropical expanded models are always self-contained, i.e.

$$d_1 < d_2 \;\Leftrightarrow\; G_i^{+d1} \;\subset\; G_i^{+d2}$$

always holds. Thus the minimum expansion found for any configuration of a path segment is implicitly the largest possible model that can move along that segment.

The following algorithm defines the rating function Q_{dist} that maps path segments dependent on a given vector of safety distances and a particular link number onto a real value:

$$Q_{dist} : \mathsf{C} \times \mathsf{C} \times \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$$

Algorithm : Q_{dist}
Input : $\bar{\mathbf{P}}_{alt}$, $\mathbf{d}_{\bar{\mathbf{P}}}$, i_{act}
Output : $Q_{dist}(\bar{\mathbf{P}}_{alt}, \mathbf{d}_{\bar{\mathbf{P}}}, i_{act})$

```
      dist := 0
      for i = 2...n                                          [ All links ]
        for j = 0...i-1, j ≠ v_i              [ Obstacles and previous links ]
*         if (¬∀q ∈ P̄_alt : _jT_i(q) G_i^{+d_P̄i} ∩ G_j = ∅)
            return COLLIDING          [ Collision occured for expanded link ]
          endif
          if (i = i_act)              [ Maximize safety distance for current link ]
**          dist_test := max_{d∈[d_P̄i, d_max_i]} (∀q ∈ P̄_alt : _jT_i(q) G_i^{+d} ∩ G_j = ∅)
            if (dist_test < dist ∨ dist = 0)
              dist := dist_test
              q_min := (Location of minimal expansion)
            endif
          endif
        endfor                                    [ Obstacles/previous links j ]
      endfor                                                   [ All links i ]
      return dist
```

The safety distance of link i_{act} is found by selecting the smallest value found in relation to all links preceeding i_{act} except the neighbouring link v_i. All links are tested to be collision-free with safety distance $d_{\bar{\mathbf{P}}i}$, otherwise COLLIDING is returned to indicate that the alternative segment is not appropriate.

Note that this rating function is suited not only for comparative rating. It implicitly defines a collision detection algorithm: if called with a non-valid link number, i.e. $i_{act} = 0$, the calculated value is either 0, indicating no collision, or COLLIDING, indicating a collision for a given vector of distances $\mathbf{d}_{\bar{\mathbf{P}}}$. In addition, Q_{dist} is also suited to calculate the current safety distance of a link along a path segment, that is unknown for the collision-free path segments prior to safety distance planning. To do so, Q_{dist} is evaluated with the path segment itself and a zero value in the respective component $d_{\bar{\mathbf{P}}i_{act}}$. The return value is the highest possible safety distance along the path segment for link i_{act}.

A possible realization of Q_{dist}, especially considering the lines $(*)$ and $(**)$, is presented in chapter 7.

5.2 Path Modification

The initial path supplied to the planning of safety distances is a collision-free path $\hat{\mathbf{P}}$, consisting of *segnum* segments $\bar{\mathbf{P}}_k$, $k = 1, \ldots, segnum$. The respective distance vectors $\mathbf{d}_{\bar{\mathbf{P}}_k}$ are initialized with all components set to zero. Usually a certain safety distance is possible for many links within the segments. Therefore, the maximum safety distance is evaluated prior to any planning. For each path segment $k = 1, \ldots, segnum$ and each link $i = 2, \ldots, n$ the following term is evaluated:

$$d_{\bar{\mathbf{P}}_k i} = Q_{dist}(\bar{\mathbf{P}}_k, \mathbf{d}_{\bar{\mathbf{P}}_k}, i)$$

The rating function is not used for comparison, but to calculate the maximum expansion. As most of the links can be expected to move in relatively free space along a collision-free path, many $d_{\bar{\mathbf{P}}i}$ will achieve the maximum value $d_{\max_i}$ right at the initialization.

5.2.1 Intention and Approach

In general it is attempted to increase the segment rating using the same basic mechanisms that are used for collision-free path planning. But in difference to this, it usually cannot be expected that all segments can be improved to having the maximum safety distance $\mathbf{d}_{\max}$. For transfer tasks the start and goal configurations are very close to collision-for the last link (respectively the load that is modeled together with the last link).

Such a situation occurs in the 2 DOF example, see fig. 5.2. The middle segment is modified successfully to achieve the safety distance. The first (last) segment cannot be altered successfully as the minimum rating occurs for $\mathbf{q}_{start}$ ($\mathbf{q}_{goal}$), thus it is divided. The new inner segments allow further modification, and this will implicitly modify the first and last segment as the adjacent intersection configurations are moved. But as in the first attempt, the first and last segment itself cannot be modified. To prevent repeated intersection, a dedicated mechanism is developed. Repeated intersection and modification could possibly result in non-terminating planning.

Considering the 2 DOF example, the planning of safety distances is terminated after two loops of the modification function, resulting in a path of 5 segments. The planning is illustrated in fig. 5.2, the resulting motion is shown in fig. 5.3 (see also p. 53).

As opposed to collision-free path planning, there is no colliding link i_{coll}. The safety distances are planned sequentially for the links 2 up to n, and the link that is currently processed is labeled i_{act}.

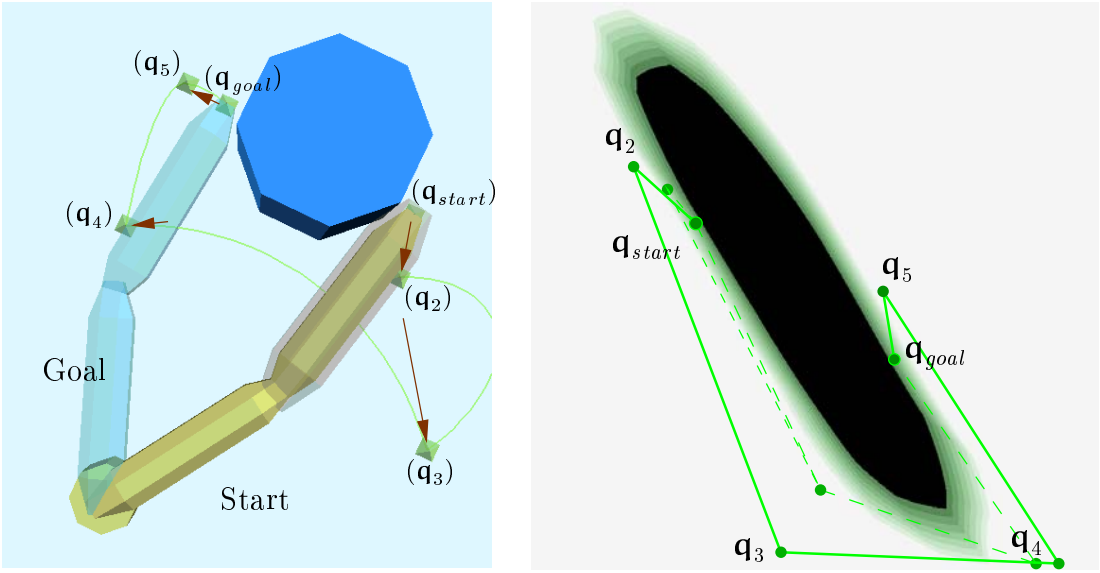

Figure 5.2: Planning for the 2-DOF manipulator visualized in the workspace and in the c-space. Left: The two intersection configurations found in collision-free path planning are pushed further away from the obstacles. New intersection configurations are inserted close to start and goal and shifted away to result in "departure" and "approach" of the obstacle. Right: The same process shown in the "rated c-space". The colour intensity indicates the possible safety distance for link two in the respective configuration, larger models are possible in brighter regions.

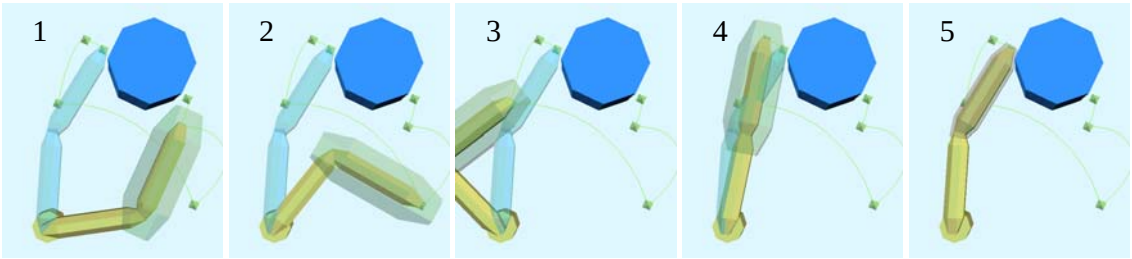

Figure 5.3: A sequence of configurations along the planned motion for the 2-DOF manipulator. The departure (approach) increases (decreases) the possible safety distance to (from) its highest possible value in a very suitable short motion. The stretching of the manipulator occurs after the "departure" in the free space on the right-hand side of the obstacle.

5.2.2 Candidate Creation

The creation of candidates employed for collision-free path planning is also used for the planning of safety distances, but here two differences apply. One is intended to minimize the computational effort, the other difference allows for the smaller magnitude of modification that is required for safety distances.

Consider the following situation: If the safety distance for link i_{act} is to be increased, no modification of the joints *beyond* this link can contribute to a possible increase. This is equivalent to collision-free path planning. But if there is a link $j \in I_{i_{act}}$, $j \neq v_{i_{act}}$ prior to the currently considered link that was unable has not been planned up to full safety distance, its currently expanded geometry model is "in touch" with another object somewhere along the segment. To apply any modification to the joints I_j is not very promising, as this can be expected to result in a collision for link j. Therefore, the subspace of C that is used to calculate the initial C-orthogonal directions is limited to the dimensions indexed by $I_{i_{act}} \setminus I_j$ for the largest link number j with $d_{\bar{\mathbf{P}}j} < d_{\max_j}$, $j < i_{act}$ and $j \neq v_{i_{act}}$ for a given path segment $\bar{\mathbf{P}}$. This results in reduced effort as the number of candidates is reduced to more promising ones for rating improvement. Note that the link $v_{i_{act}}$ that directly preceeds link j is not included in this consideration. If it was, the limitation of the number of considered joints might result in an only onedimensional subspace of C, where no candidates can be created.

The second difference to the candidate calculation used for collision-free path planning is the upper limit of the cartesian displacement. It is reasonable to avoid too large displacements that might push the path segment, especially with the links beyond i_{act}, into collision. The path as a whole is already collision-free, and the safety distance planning ought to alter it just slightly. Therefore, the upper displacement limit is set to twice the desired safety distance $d_{\max_{i_{act}}}$, a value that is typically much smaller than $\delta_{\max}$. With twice the desired safety distance a "good" alternative path can achieve full safety distance within one step. For a segment made up of $\mathbf{q}_a$ and $\mathbf{q}_b$, the actual displacement d results to:

$$d = \begin{cases} \delta_{\min} & : \quad f_{step}\,\Delta_{i_{act}}(\mathbf{q}_a, \mathbf{q}_b) < \delta_{\min} \\ 2\,d_{\max_{i_{act}}} & : \quad f_{step}\,\Delta_{i_{act}}(\mathbf{q}_a, \mathbf{q}_b) > 2\,d_{\max_{i_{act}}} \\ f_{step}\,\Delta_{i_{act}}(\mathbf{q}_a, \mathbf{q}_b) & : \quad else \end{cases}$$

5.2.3 Replacement Selection

The replacement selection scheme used for the planning of safety distances works analogous to the scheme employed for collision-free planning. It is attempted to modify one segment $\bar{\mathbf{P}}_{act} = \overline{\mathbf{q}_a \mathbf{q}_b}$ in such a way that its rating is improved. The previous segment $\bar{\mathbf{P}}_{prev} = \overline{\mathbf{q}_{prev} \mathbf{q}_a}$ and the following segment $\bar{\mathbf{P}}_{next} = \overline{\mathbf{q}_b \mathbf{q}_{next}}$ are checked to assure that their rating stays at least equal if the intersection configurations are altered.

The respective algorithm is referred to as **ModifySegmentDist** in the following, and it works like the algorithm **ModifySegmentScale** (see p. 61). The initial value for *bestrating* is the currently achieved safety distance for the segment $\bar{\mathbf{P}}_{act}$, $d_{\bar{\mathbf{P}}_{act} i_{act}}$. Only if an alternative path with a better rating can be found, a replacement takes place. In the case that neither $\mathbf{q}_a = \mathbf{q}_{start}$ nor $\mathbf{q}_b = \mathbf{q}_{goal}$ a best possible alternative path is selected by evaluating the elements of the candidate sets $\mathbf{c}^a$ and $\mathbf{c}^b$ as follows:

if ($Q_{dist}(\overline{\mathbf{q}_{prev}\mathbf{c}_k^a}, \mathbf{d}_{\bar{\mathbf{P}}_{prev}}, i_{act}) \geq d_{\bar{\mathbf{P}}_{prev} i_{act}} \wedge$
$Q_{dist}(\overline{\mathbf{c}_k^a \mathbf{c}_k^b}, \mathbf{d}_{\bar{\mathbf{P}}_{act}}, i_{act}) > bestrating \wedge$
$Q_{dist}(\overline{\mathbf{c}_k^b \mathbf{q}_{next}}, \mathbf{d}_{\bar{\mathbf{P}}_{next}}, i_{act}) \geq d_{\bar{\mathbf{P}}_{next} i_{act}}$)
$bestrating := Q_{dist}(\overline{\mathbf{c}_k^a \mathbf{c}_k^b}, \mathbf{d}_{\bar{\mathbf{P}}_{act}}, i_{act})$
$\mathbf{q}_a^* := \mathbf{c}_k^a$
$\mathbf{q}_b^* := \mathbf{c}_k^b$
endif

The two other cases where there is just one intersection configuration modified to form an alternative path are treated analogously. If a combination of intersection configurations $\mathbf{q}_a^*$, $\mathbf{q}_b^*$ can be found that improves the rating of $\bar{\mathbf{P}}_{act}$, the distance information of the path segments has to be updated for all modified segments to assure that the highest possible safety distance is stored in all $\mathbf{d}_{\bar{\mathbf{P}}}$:

Replace $\mathbf{q}_a$ by $\mathbf{q}_a^*$, $\mathbf{q}_b$ by $\mathbf{q}_b^*$
$d_{\bar{\mathbf{P}}_{prev} i_{act}} := Q_{dist}(\overline{\mathbf{q}_{prev}\mathbf{q}_a^*}, \mathbf{d}_{\bar{\mathbf{P}}_{prev}}, i_{act})$
$d_{\bar{\mathbf{P}}_{akt} i_{act}} := bestrating$
$d_{\bar{\mathbf{P}}_{next} i_{act}} := Q_{dist}(\overline{\mathbf{q}_b^* \mathbf{q}_{next}}, \mathbf{d}_{\bar{\mathbf{P}}_{next}}, i_{act})$

5.2.4 Path Refinement

The following situation shall be considered: $\mathbf{q}_{start}$ is too close to collision to allow the maximum desired safety distance $d_{\max_i}$ for a link i. $\mathbf{q}_b$ shall be the first intersection configuration, thus $\overline{\mathbf{q}_{start}\mathbf{q}_b}$ is the first segment of the path. $\mathbf{q}_{start}$ itself cannot be modified, and whatever modification is applied to $\mathbf{q}_b$, the rating of the segment is either the safety distance possible at configuration $\mathbf{q}_{start}$ or below.

Therefore, the segment is subject to subdivision. A new intersection configuration $\mathbf{q}_a$ is created in between $\mathbf{q}_{start}$ and $\mathbf{q}_b$ and is subject to further modification. The new segment $\overline{\mathbf{q}_{start}\mathbf{q}_a}$ may suffer the same problem as the original segment $\overline{\mathbf{q}_{start}\mathbf{q}_b}$. To avoid this, the segment has to be excluded from further subdivision and modification attempts. But this is not sufficient to avoid repeated subdivision in general. If $\mathbf{q}_a$ cannot be shifted far enough to assure maximum safety distance for the link, the segment $\overline{\mathbf{q}_a\mathbf{q}_b}$ in turn is in danger to be repeatedly subdivided.

To take this into account the subdivision mechanism used for collision-free path planning is extended. If the segment that is to be divided is the first (or last) of the path, an intersection configuration very close to start (goal) is calculated and the resulting short segment is marked $[DoNotModify]$ to be excluded from further modification attempts, i.e. it is in turn neither attempted to displace it, nor is it further subdivided. The position is chosen close to the fixed configuration to assure the topologic adaptability of the entire path. It is not useful to prohibit subdivision for long path segments, thus it is not divided using the mechanisms of "regular" subdivision that is intended to keep the segments long. Note that the segment may be prolonged in the further planning due to modifications of the adjacent segment, as in the 2 DOF example above. In such a case the prolonged segment will result in a more or less direct motion into a region with higher safety distances, a very desirable behaviour. It is not required to subdivide such a segment again.

If the adjacent segment of the first (last) segment cannot be modified, it is treated like the first (last) segment before: it is cut asymmetrically and marked. As a result of repeated failed attempts to modify segments close to start and goal, the mark is propagated into the path, and the approach/departure of the resulting robot motion is fixed for link i_{act}. The principle is illustrated in fig. 5.4.

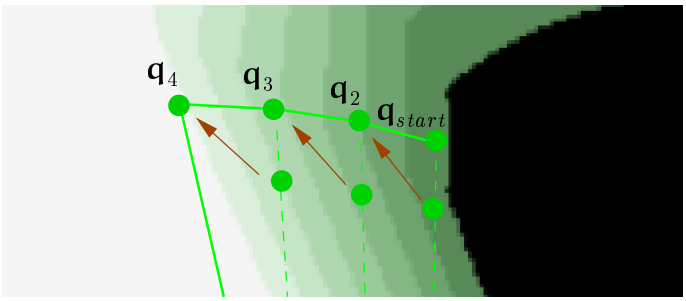

Figure 5.4: Principle of repeated creation and modification of intersection configurations close to the start configuration, visualized in a two-dimensional c-space. It allows to plan a suitable departure away from the obstacle surface. The new short segments are excluded from modification and subdivision.

In case of such an asymmetrical cut, the length chosen for the new, marked segment is calculated using $\delta_{\min}$, the preset parameter that controls the lower length of path segments. The new intersection configuration is placed in such a way that the estimated cartesian motion of link i_{act} along the new segment is approximately $\delta_{\min}$. If the intersection takes place close to $\mathbf{q}_a$, either because $\mathbf{q}_a = \mathbf{q}_{start}$ or all segments preceeding $\mathbf{q}_a$ are marked, $\mathbf{q}_{new}$ is found with:

$$\mathbf{q}_{new} := \mathbf{q}_a + \frac{\delta_{\min}}{\Delta_{i_{act}}(\mathbf{q}_b, \mathbf{q}_a)}(\mathbf{q}_b - \mathbf{q}_a) \tag{5.1}$$

The algorithm **DivideSegmentDist** below shows the complete processing. If the segment to be divided is neither adjacent to $\mathbf{q}_{start}$, $\mathbf{q}_{goal}$ nor to a marked segment, it is divided in relation to the location $\mathbf{q}_{min}$ of minimal rating, using the same mechanism as for collision-free path planning.

Algorithm : **DivideSegmentDist**
Input : $\overline{\mathbf{q}_a \mathbf{q}_b}$
Output : FAILURE or SUCCESS

* **if** $(\Delta_{i_{act}}(\mathbf{q}_a, \mathbf{q}_b) < \delta_{\min})$
Mark segment $\overline{\mathbf{q}_a \mathbf{q}_b}$ $[DoNotModify]$
return FAILURE
endif
if $(\mathbf{q}_a = \mathbf{q}_{start} \vee$ (segments $\mathbf{q}_{start} \ldots \mathbf{q}_a$ marked $[DoNotModify]))$
Calculate $\mathbf{q}_{new}$ close to $\mathbf{q}_a$ using (5.1)
elsif $(\mathbf{q}_b = \mathbf{q}_{goal} \vee$ (segments $\mathbf{q}_b \ldots \mathbf{q}_{goal}$ marked $[DoNotModify]))$
Calculate $\mathbf{q}_{new}$ close to $\mathbf{q}_b$ analogous to (5.1)
else
Calculate $\mathbf{q}_{new}$ using **DivideSegmentScale**$(\overline{\mathbf{q}_a \mathbf{q}_b})$
endif
Insert $\mathbf{q}_{new}$ into path, creating 2 colinear segments
$segnum := segnum + 1$
** **if** $(\Delta_{i_{act}}(\mathbf{q}_a, \mathbf{q}_{new}) \leq \delta_{\min})$
Mark segment $\overline{\mathbf{q}_a \mathbf{q}_{new}}$ $[DoNotModify]$
endif
** **if** $(\Delta_{i_{act}}(\mathbf{q}_{new}, \mathbf{q}_b) \leq \delta_{\min})$
Mark segment $\overline{\mathbf{q}_{new} \mathbf{q}_b}$ $[DoNotModify]$
endif
return SUCCESS

Beside the especially critical situation close to start and goal it may as well be impossible to plan the full safety distances *along* a path. This may be due to a passage that has to be passed, or due to an implicit passage formed by the obstacles and the joint limits. To avoid repeated attempts of modification in such a spatially constrainted region, all segments that fall below the length threshold $\delta_{\min}$ are marked like the dedicated segments close to start and goal, and thus excluded from further planning. The length limit may either be reached because an adjacent segment is modified in a way that the current segment is shortened (handled in line $(*)$ of the algorithm), or as a result of the subdivision itself (lines $(**)$). Note that segments might be prolonged again due to adjacent segment modifications, but the mark is not removed, i.e. these segments are *passive* within further planning for the current link i_{act}.

Note that the case $\mathbf{q}_a = \mathbf{q}_{start}$ and $\mathbf{q}_b = \mathbf{q}_{goal}$ is neglected for clarity in the above algorithm. If it occurs, i.e. safety distances are to be planned for an initially collision-free path without intersection configurations, segments of length $\delta_{\min}$ (or below if $\Delta_{i_{act}}(\mathbf{q}_a, \mathbf{q}_b) < \delta_{\min}$) are cut off on either end of the path and marked accordingly.

Whenever a segment is divided, both new segments inherit the vector $\mathbf{d}_{\bar{\mathbf{P}}}$ of current safety distances, and the component $d_{\bar{\mathbf{P}} i_{act}}$ is updated using Q_{dist}. There may be different distances possible for both parts, and the current segment is subject to subdivision, because its possible safety distance is below the desired value, i.e. $d_{\bar{\mathbf{P}} i_{act}} < d_{\max_{i_{act}}}$.

5.2.5 Planning Algorithm

The planning algorithm combines the mechanisms described above to achieve the desired objective. The outer main loop processes one link after another unless the desired safety distance $d_{\max_{i_{act}}} = 0$. Whenever planning is started for a new link all segments are unmarked. As there may be several links close to the obstacles, e.g. when planning for a forked kinematic chain, the planning has to be able to modify the path as a whole for each link individually.

The planning for lower links also influences the position of the upper links, and the possible safety distances may have increased, but the values in $\mathbf{d}_{\bar{\mathbf{P}}}$ have not been adapted. Therfore, the values $d_{\bar{\mathbf{P}}_k i_{act}}$ are updated in each iteration of the outer main loop.

In contrast to the collision-free path planning, where a dedicated segment is found that has to be improved, the inner loop of safety distance planning repeatedly attempts to shift all segments with a rating below the target value and divides them if no alternation is possible – unless they are marked $[DoNotModify]$. This is repeated as long as any modification can be applied to the path. If the planning switches up to the next link, all path segments are either at an appropriate safety distance, or they are marked because they are close to start/goal, or their estimated cartesian motion falls below the threshold $\delta_{\min}$.

This algorithm sketches the complete planning of paths with safety distances:

Algorithm : **PlanDist**
Input : Collision-free path, $\mathbf{d}_{\max}$
Output : Collision-free path with safety distances

```
Initialize all d_{P̄_k}
for i_act = 2...n, d_{max_{i_act}} ≠ 0          [ Outer loop – all links ]
  Unmark all segments                            [ Allow modification ]
  Update all d_{P̄_k i_act}
  altered := TRUE                                [ Modification flag ]
```

```
*       while altered                    [ As long as any modification applied ]
          altered := FALSE
          for k = 1 ... segnum                                   [ All segments ]
            if (d_{P̄_k i_act} < d_{max_{i_act}} ∧ P̄_k not marked [DoNotModify])
              if (ModifySegmentDist(P̄_k) = SUCCESS)
                altered := TRUE
              elsif (DivideSegmentDist(P̄_k) = SUCCESS)
                altered := TRUE
                k := k + 1                                  [ Skip new segment ]
              endif
            endif
          endfor                                                 [ All segments ]
        endwhile                                     [ Any modification applied ]
      endfor                                                   [ All links i_act ]
```

The safety distances $\mathbf{d}_{\bar{\mathbf{P}}}$ are stored together with the path and are part of the planning result that is used either for further planning (reduction of the path length) or may be used directly within the trajectory generation of the robot itself. This realization follows objective **O3**. The safety distance of the path is increased wherever possible. The path length is not explicitly considered. But the path modification is based on approximate W-orthogonal displacement applied to the intersection configurations, and this results in suitable motions "around" the obstacles or "away" from the obstacles into clear space close to start and goal.

5.2.6 Termination

The inner loop of the planning process terminates if all segments are either in free space or marked not to be further modified, as a result of the length-limited subdivision. This practically guarantees termination, and no non-terminating cases were observed in the practical experiments. But as it is the case with collision-free path planning there are situations where the developed mechanisms may not be sufficient. To enforce termination under any circumstances, the number of iterations of the inner loop (marked with $(*)$ in the above algorithm) is limited to a value practically never reached (order of 1000).

Other mechanisms to assure termination are possible, e.g. limiting the number of path segments, or limiting the run time. It should be noted that the planning of safety distances has any-time characteristics. With longer planning better results are achieved. Planning is based on a collision-free path, no stage of the planning results in a collision along the path and the quality is continuously increased.

5.3 Planning Examples

The planning of safety distances is applied to the solutions found for the sample tasks in the previous chapter. To evaluate the "quality" of safety distance planning, an approximate measure m_{dist} is used. It relates the maximum possible safety distance along an entire path $\hat{\mathbf{P}}$ made up of a sequence of path segments $\bar{\mathbf{P}} = \overline{\mathbf{q}_a \mathbf{q}_b}$ to the actually achieved safety distance:

$$m_{dist} = \frac{\sum_{\bar{\mathbf{P}} \in \hat{\mathbf{P}}} (\parallel \mathbf{q}_a - \mathbf{q}_b \parallel \sum_{i=1}^{n} d_{\bar{\mathbf{P}} i})}{\sum_{\bar{\mathbf{P}} \in \hat{\mathbf{P}}} (\parallel \mathbf{q}_a - \mathbf{q}_b \parallel \sum_{i=1}^{n} d_{\max_i})} \tag{5.2}$$

m_{dist} is 1 (denoted as 100%) if a path allows full safety distances everywhere, and m_{dist} is zero if no safety distance is possible anywhere. All possible percentages in between allow roughly to qualify the path. The "shape" of the path has to be considered as well to really qualify a path as m_{dist} *increases* if arbitrary detours prolong the motion in regions were maximum safety distances are possible.

The 6-DOF Manipulator

The load has to be moved from a position "in touch" with the floor into a position "in touch" with the table. Departure from the floor is almost orthogonal and cannot be improved (see fig. 4.9, p. 67). The main difficulty is passing the edge of the table and the placement of the load. In this example a safety distance of 25 mm was chosen for all links except link one. This is a lot, considering the load (which is part of link 6): its short edge is also 25 mm long, thus the volume of the load is increased by more than a factor of nine when the model is fully expanded.

The initial collision-free path with its three intersection positions is shown in fig. 5.7.1. Its "safety distance quality" is $m_{dist} = 81.7\%$.

The safety distances are easily planned for links two up to five, as these links do not touch the obstacles at start or goal. The two intersection configurations in the gap between the robots base and the table are modified accordingly (5.7.2 and 5.7.3).

When the outer planning loop sets i_{act} to six, the rating for the first and last segment is very low, and modification attempts fail. New intersection configurations are inserted in approximately $\delta_{\min}$ distance to the respective fixed configurations (5.7.4). One of these "special" subdivision configurations is sufficient at the start, it is moved in such a way that the second segment has full safety distance (5.7.5), and the departure is fixed. At the goal, two further intersection configurations are created before planning terminates (5.7.6). The resulting path has a quality measure of $m_{dist} = 98.9\%$.

The planned motion is shown in fig. 5.6. Only the approach is visualized, as it is the most interesting part of the motion. The load is tilted down over the edge as it is in the collision-free motion, but now this happens *above* the table-top.

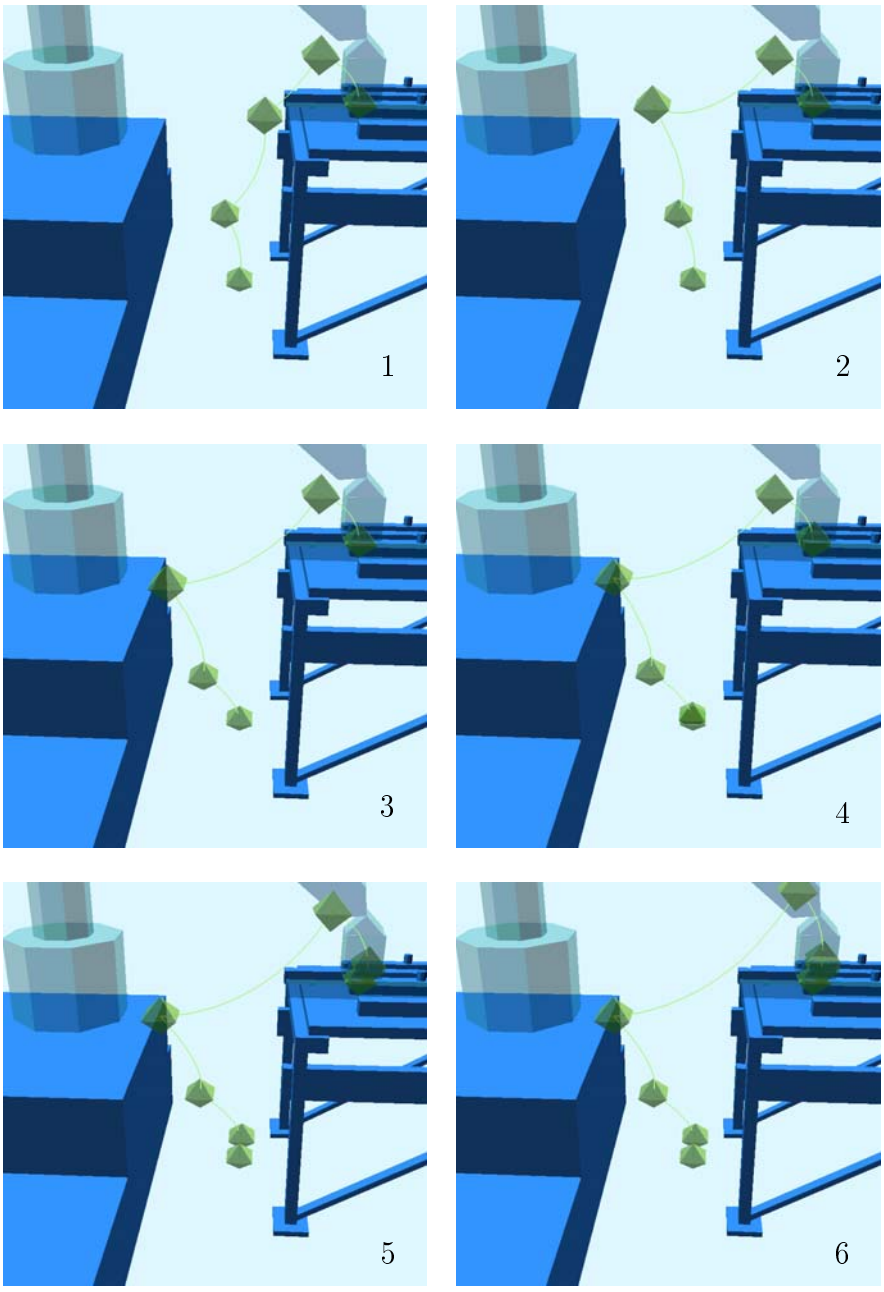

Figure 5.5: Planning for the 6-DOF manipulator. For clarity, only the tip trace and the goal configuration are shown. 1: The initial, collision-free path. 2–5: Intermediate stages of the planning process. Note especially the short segments that are created close to start and goal. 6: The path with safety distances found as a solution.

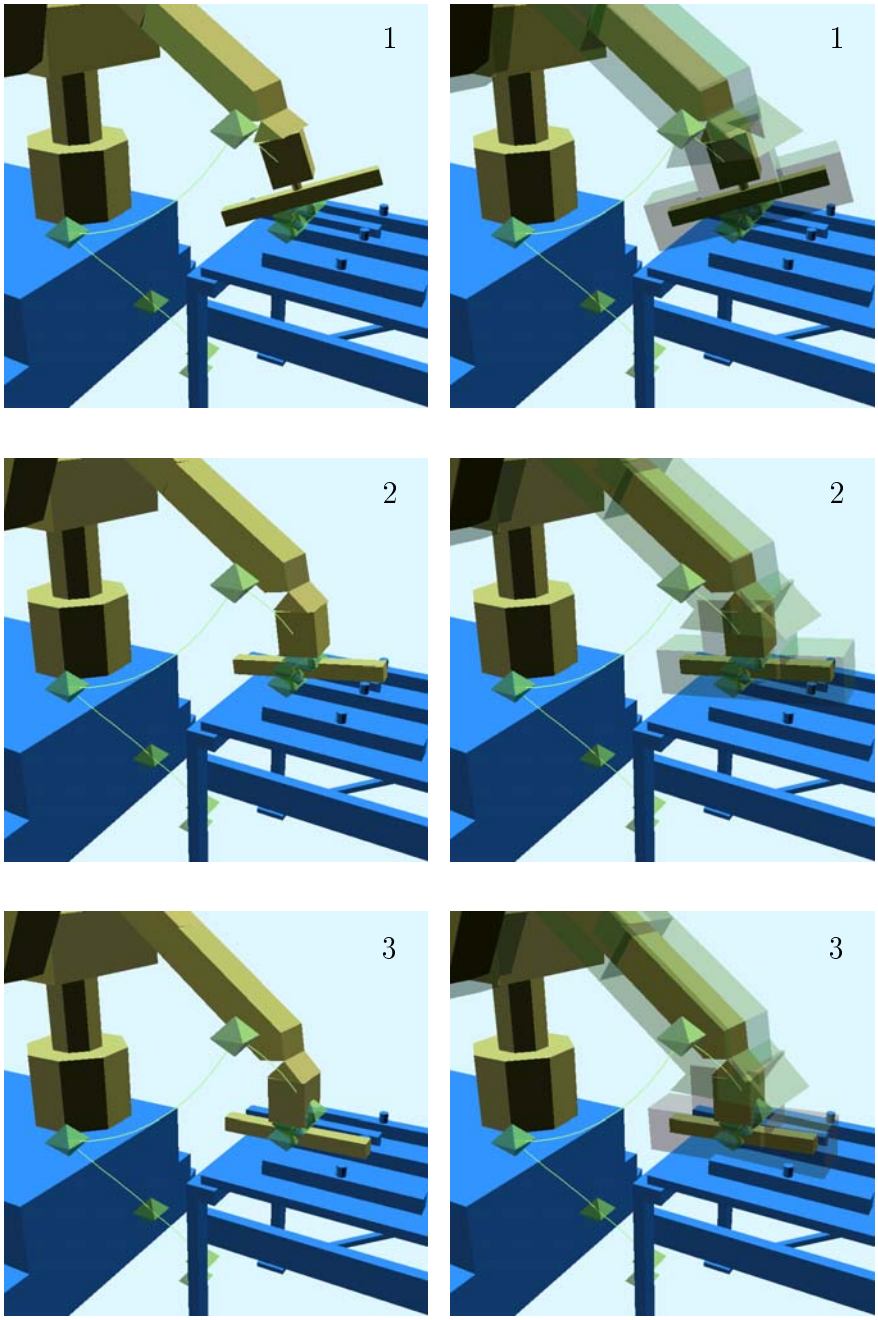

Figure 5.6: Three configurations from the approach sequence of the planned motion. The left row shows the original model, the right row shows the same configurations with the possible expansion applied to the robot model (note that the drawn expanded models are only polyhedral approximations of isotropically expanded geometry models). The load is aligned with its final position before it gets closer to the table than the maximum safety distance allows.

The very last "touch down" of the load is more or less orthogonal, i.e. the load is moved in parallel to its final position.

The 16-DOF Manipulator

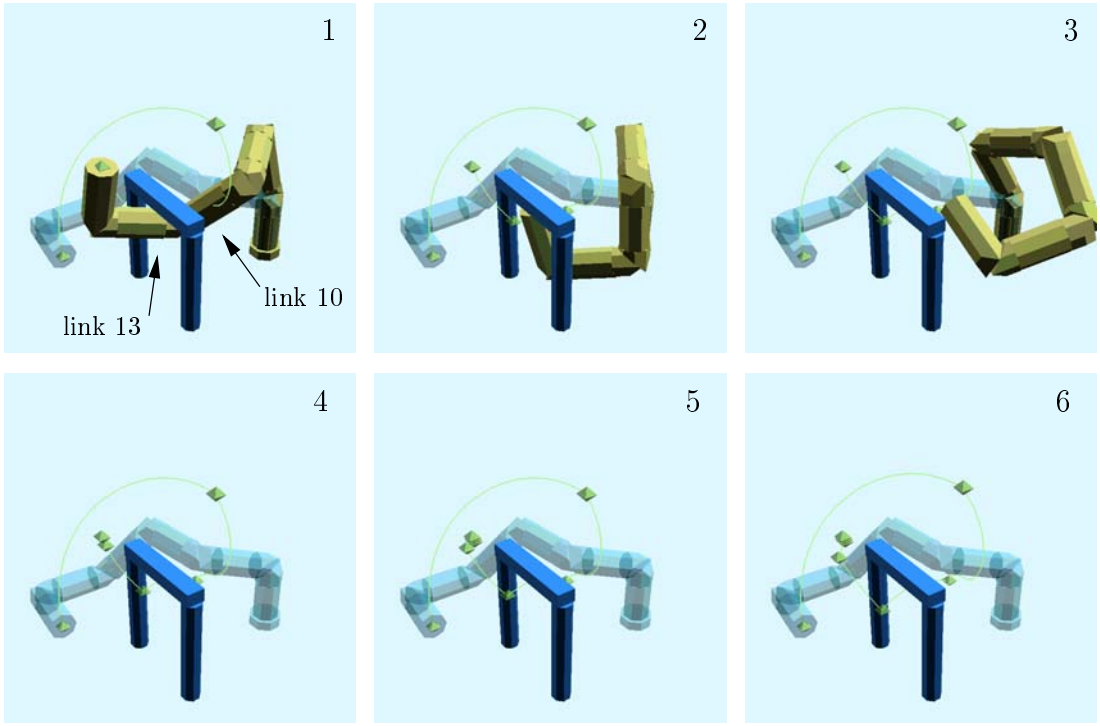

Figure 5.7: Planning for the 16-DOF manipulator. 1–3: Configurations close to collision along the collision-free path. 4–5: Intermediate stages of planning. For clarity, only the goal configuration and the tip trace are shown. Note the short segments that are created close to start. 6: Tip trace of the final path with safety distances.

For the 16-DOF manipulator the situation is completely different. Considering the tool of this robot, start and goal are not close to the obstacle. But links 10 through 13 are very close to the upper bar of the gate at the start configuration (see fig. 5.7.1). And along the initial collision-free path, link 16 passes the upper bar in a minimal distance (5.7.2 and 5.7.3). The parameter chosen for the planning is a desired safety distance of 30 mm for all links (except link 1). Note that this reduces the available motion space for the robot in the gate very much. The intial quality is $m_{dist} = 94.2\%$, as many of the links can move with full safety distance along the collision-free path.

No planning is required for $i_{act} = 2, \ldots, 9$. When link 10 is reached, the segment considered is the first of the path, and it cannot be improved, as link 10 is close to the gate directly at the start. A new intersection configuration is inserted and slightly shifted (5.7.4). The same procedure is repeated for link 12 and 13 (link 11 was displaced sufficiently together with link 10), thus finally three inner configurations exist close to start (5.7.5). Link 14 and 15 do not require planning, and planning for link 16 can be successfully finished with the existing intersection configurations (5.7.6). The segment that passes the gate is adjusted to allow the larger robot to pass.

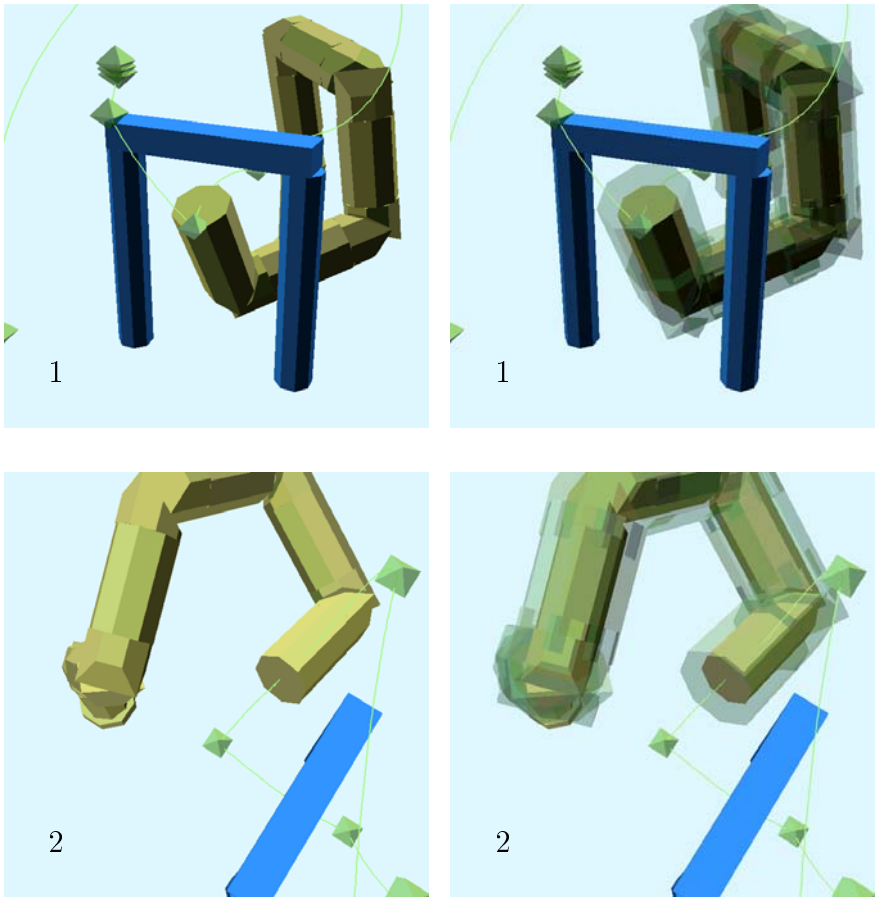

Figure 5.8: Two configurations from the planned motion. The left row shows the original model, the right row shows the same configurations with the possible expansion applied to the robot model.

This is visualized in fig. 5.8 where the robot passes the gate itself and starts its final turn towards the goal in appropriate distance to the obstacle. The resulting path consists of seven path segments, and the quality has reached $m_{dist} = 99.9\%$

since just a very short departure is needed, and almost the entire path can be moved with full safety distances.

5.4 Complexity and Computational Effort

Complexity

In principle the same considerations that have been made for collision-free path planning also apply to the inner loop of planning of safety distances (see 4.6, p. 74). The number of displacement candidates and the number of evaluated alternative paths is linear in the number of degrees of freedom, this dominates the effort of base vector calculation and candidate handling. Thus the effective complexity of planning the safety distance for one link is $O(n)$, i.e. it is linear in the number of joints for robots with a reasonable limited number of joints.

The outer loop processes all links sequentially, the complexity is linear in n. The overall effective complexity of planning of safety distances is quadratic in the number of joints, i.e. results to $O(n^2)$. The memory requirement is very small, as only the current path is kept, together with distance information.

Computational Effort

The computational effort was considered throughout the developement of the planning scheme, and the resulting mechanism is a trade-off between efficiency and the achieved quality of the result: plan the best result possible with the smallest effort necessary.

Compared to the planning of collision-free paths, on the one hand the effort is increased since the rating requires collision detection not only for the links up to link i_{act}, but for all links. On the other hand the effort is decreased, as the required displacement, and thus the number of required modification steps, is typically much smaller in the areas of free space.

The planning of safety distances can be compared to the planning of a collision-free path for a "blown up" model of the robot. Whenever a passage has to be planned that cannot be passed with full size, the planning "fails" for the virtually enlarged robot. Thus planning of safety distances has to adapt to a "local minimum" of the rating function. It is not sufficient to concentrate on the worst segment until "failure" can be recognized and planning can be aborted, but the path has to be bent into regions of highest rating. This results in numerous small segments, and requires an increasing number of modification steps. The planning is well suited to handle such passages successfully, but it cannot be as efficient as collision-free path planning in such a situation.

The effort is therefore inherently controlled by the parameter $\mathbf{d}_{\max}$. The system developed is intended to plan efficiently for relatively small safety distances, as they are realistic in a technical environment (orders of a few centimetres). The

planning is not intended to push a whole path to maximum possible workspace distance for all links. Realistic parameters result in efficient planning.

The developed "continuation" of collision-free path planning is more than just a continuation: the paths found are increased in quality, and the additional information makes them applicable to real robots.

Chapter 6

O3 – Planning Short Paths

A collision-free path planned by using the methods of the previous chapters usually contains unnecessary detours in free space. They are either a result of local planning, where the number of intersection configurations is intentionally kept low, or a result of random subgoals.

If such a path is used to control a robot, the required time and energy is wasted, as a shorter and more direct path might be equally collision-free and keep the same safety distances. It is the intention of the mechanisms developed in this chapter to efficiently find such a shorter path. The scheme is based on the same fundamental principle: create candidates, rate alternative paths, replace intersection configurations if the rating can be improved, and refine the path by insertion of additional intersection configurations to allow increased topological adaption. But the rating used and the modification and refinement employed differ from the schemes used for the other objectives.

Informally speaking, the path is seen as a rope, which is shortened by pulling at its ends until the virtually expanded robot moves touching the obstacles.

6.1 Rating Function Q_{short}

The suitable rating function to plan short paths is obviously the path length in c-space. It can neither be defined for single configurations nor single path segments. Comparative rating is only possible in the context of at least two consecutive segments. For an entire path $\hat{\mathbf{P}}$ made up of a sequence of path segments $\bar{\mathbf{P}} = \overline{\mathbf{q}_a \mathbf{q}_b}$, the length is given by

$$Q_{short}(\hat{\mathbf{P}}) = \sum_{\bar{\mathbf{P}} \in \hat{\mathbf{P}}} \| \mathbf{q}_a - \mathbf{q}_b \|$$

Within the path modification this rating is not calculated explicitly, but implicitly decreased by successful modification steps. The path length is measured in C, in contrast to the other rating functions which were motivated by considering the "virtual" robot in the work space. As the robot moves by driving its joints, a path containing less motion in the individual joints is shorter in execution time and energy consumption. Considering the cartesian motion of dedicated control points of the robot moving along a path, a path that is shorter according to

Q_{short} may result in longer motion in W, but is, in accordance with the objective, a "better" path.

6.2 Path Modification

The initial path available for the length reduction is a collision-free path $\hat{\mathbf{P}}$, consisting of $segnum$ segments $\bar{\mathbf{P}}_k$, $k = 1, \ldots, segnum$. The respective distance vectors $\mathbf{d}_{\bar{\mathbf{P}}_k}$ contain the achieved safety distances for the links moving along the path segments.

6.2.1 Intention and Approach

The rating based on the path length is not the only criteria that needs to be applied when the path is modified. The path must not be shortened "into collision" and the achieved safety distances have to be kept as well.

The approach developed is based on the local polygonal path optimization reported in Berchtold and Glavina (1994). The heuristics employed in this publication are very simplified, and the major disadvantage of Berchtold and Glavina (1994) is eliminated: the path is no longer "pulled" onto the surface of the obstacles.

Considering only two non-colinear path segments $\overline{\mathbf{q}_a\mathbf{q}_b}$ and $\overline{\mathbf{q}_b\mathbf{q}_c}$ within a path, the overall path length can be shortened if the two path segments are replaced by just one path segment, as the triangular inequation applies:

$$\| \mathbf{q}_a - \mathbf{q}_c \| \leq \| \mathbf{q}_a - \mathbf{q}_b \| + \| \mathbf{q}_b - \mathbf{q}_c \|$$

But the two considered segments may have different safety distances for any of their links, thus they must not be unified if only the lower safety distance values are applicable to the new segment $\overline{\mathbf{q}_a\mathbf{q}_c}$. To do so would actually decrease the quality of the overall path, as the known safety distance is decreased in part.

Instead of replacing $\overline{\mathbf{q}_a\mathbf{q}_b}$, $\overline{\mathbf{q}_b\mathbf{q}_c}$ by $\overline{\mathbf{q}_a\mathbf{q}_c}$, a configuration on the linear connection can be chosen to modify the two segments, if the two alternative segments are collision-free for the same safety distances that were achieved for the original segments. This allows two different safety distances along a linear partial path.

There may be relatively long segments within a path. To allow the adaption of the path to the topology of the obstacles, path refinement is required, i.e. the segments have to be divided. There is no active link analogous to i_{coll} within collision-free path planning or i_{act} within the planning of safety distances, therefore the subdivision cannot be based on a location $\mathbf{q}_{min}$ that marks the minimal scaling/expansion. Without this additional information, segments are cut in their middle if they cannot be modified in their current length.

Considering these two aspects a local path length reduction is developed: it is attempted to replace intersection configurations in such a way that triangles

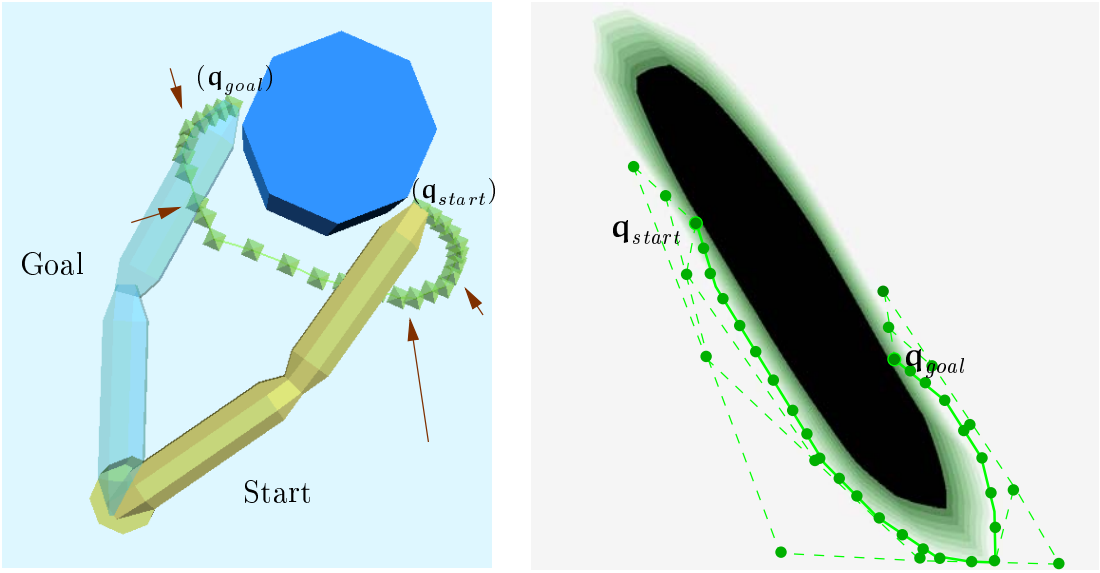

Figure 6.1: Planning for the 2-DOF manipulator visualized in the workspace and in the c-space. Left: The planned path with safety distances "collapses" in such a way so that the robot does not get closer to the obstacle than the $d_{\max_2}$, with the exception of departure and approach. Right: The same process shown in the "rated c-sapce". The path shape iteratively adapts to the outer contour that indicates smaller safety distance. The path length is reduced to below 60% of the length of the initial path.

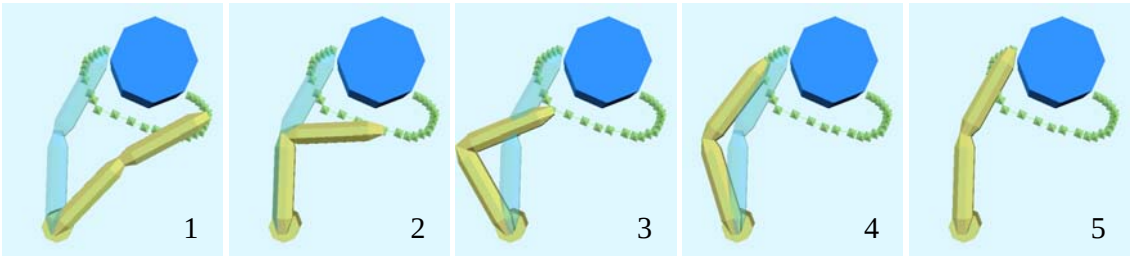

Figure 6.2: A sequence of configurations along the planned motion for the 2-DOF manipulator. The stretching of the manipulator occurs within the departure, the long detour of the tip is removed. Link 2, expanded by the safety distance, "slides" around the obstacle and captures its countour.

are cut off. If this fails, the path segments are divided. These two steps are repeated until no further modification is possible with respect to certain threshold parameters.

The resulting behaviour is shown for the 2 DOF example, see fig. 6.1. The collision-free path with safety distances (see also p.53 and p.81) is shortened until the final path has adapted itself properly to the borderline of safety distance and consists of 26 short path segments. The resulting motion is shown in fig. 6.2.

6.2.2 Candidate Creation

For two path segments $\overline{\mathbf{q}_a\mathbf{q}_b}$ and $\overline{\mathbf{q}_b\mathbf{q}_c}$, just one candidate configuration $\mathbf{q}_{inner}$ needs to be created to form an alternative, shorter path. It is found on the C-linear connection of the outer configurations $\mathbf{q}_a$ and $\mathbf{q}_c$.

The main aspect to consider is the length relation between the two segments formed by the candidate. Consider the following situation: both segments allow different safety distances, and the configuration $\mathbf{q}_b$ is relatively close to one of the outer configurations. Two cases can be differentiated:

- $\overline{\mathbf{q}_b\mathbf{q}_c}$ allows a higher safety distance for at least one of the links. In this case, the center configuration between $\mathbf{q}_a$ and $\mathbf{q}_c$ is not a promising candidate, because it cannot be expected that the prolonged segment $\overline{\mathbf{q}_{inner}\mathbf{q}_c}$ allows the higher safety distance.
- $\overline{\mathbf{q}_a\mathbf{q}_b}$ allows a higher safety distance. In this case, the successful replacement of $\mathbf{q}_b$ with $\mathbf{q}_{inner}$ would reduce the overall quality of the path, as a short segment with lower safety distances might be replaced by a longer segment.

To allow for this, the candidate configuration is calculated in a way that the length relation between the two considered segments remains equal for the two alternative segments:

$$\mathbf{q}_{inner} := \mathbf{q}_a + (\mathbf{q}_c - \mathbf{q}_a)\frac{\| \mathbf{q}_b - \mathbf{q}_a \|}{\| \mathbf{q}_b - \mathbf{q}_a \| + \| \mathbf{q}_c - \mathbf{q}_b \|}$$

This kind of candidate calculation assures that $\mathbf{q}_{inner}$ is on the linear connection between $\mathbf{q}_a$ and $\mathbf{q}_c$, as the fraction is < 1 for any $\mathbf{q}_a$, $\mathbf{q}_b$, $\mathbf{q}_c$. Note that $\mathbf{q}_b = \mathbf{q}_{inner}$ if the segments are colinear.

6.2.3 Replacement Selection

The replacement selection has to decide principally whether the candidate is sufficient, i.e. whether it assures the same safety distances for the alternative segments. This is done by using the rating function Q_{dist} which realizes a collision detection for path segments with safety distances applied.

The two segments considered may already be colinear, or almost colinear. In this case, a replacement is not required, and no modification is applied. This is verified using a preset control parameter $\rho > 0$, that is a lower threshold for the ratio between the displacement from $\mathbf{q}_b$ to $\mathbf{q}_{inner}$ and the possible straight line

length from $\mathbf{q}_a$ to $\mathbf{q}_c$. In other words, it limits the relative height of the triangle: if it is "too flat", no modification is applied. The following algorithm describes the attempted replacement of triangles by straight connections:

Algorithm : **CutTriangle**
Input : $\bar{\mathbf{P}}_{act} = \overline{\mathbf{q}_a \mathbf{q}_b}$, $\bar{\mathbf{P}}_{next} = \overline{\mathbf{q}_b \mathbf{q}_c}$
Output : FAILURE OS SUCCESS

Calculate $\mathbf{q}_{inner}$
if $(\frac{\|\mathbf{q}_{inner} - \mathbf{q}_b\|}{\|\mathbf{q}_c - \mathbf{q}_a\|} < \rho)$
  return FAILURE *[Triangle "flat enough"]*
endif
if ($Q_{dist}(\overline{\mathbf{q}_a \mathbf{q}_{inner}}, \mathbf{d}_{\bar{\mathbf{P}}_{akt}}, 0) = 0 \wedge$
  $Q_{dist}(\overline{\mathbf{q}_{inner} \mathbf{q}_c}, \mathbf{d}_{\bar{\mathbf{P}}_{next}}, 0) = 0$)
  Replace $\mathbf{q}_b$ by $\mathbf{q}_{inner}$ *[Both new segments are collision-free]*
  return SUCCESS
endif
return FAILURE *[No modification possible]*

Whenever this function terminates successfully, the overall path is shortened, i.e. the implicit rating based on Q_{short} is decreased.

6.2.4 Planning Algorithm

The complete planning algorithm works itself along the current path, considers two segments at a time and locally attempts to straighten it. This kind of modification is repeated, i.e. the entire sequence of segments is run through again and again, as long as any pair of segments can be modified.

If it is not possible to cut off a triangle, all segments that do not fall below a preset threshold length λ are divided into two new segments of equal length, and the modification of segment pairs is started again. As opposed to the planning of collision-free paths and safety distances, the subdivision threshold cannot be given as the estimated cartesian motion of a link, as there is no current link that is planned for. Therefore the minimum segment length λ is a control parameter that describes a length in C.

When a segment is divided, both new segments inherit the vector $\mathbf{d}_{\bar{\mathbf{P}}}$ of current safety distances. If no more segments can be divided, because the length of all segments in the path falls below λ, the function is terminated. The complete algorithm can be written as:

Algorithm : **PlanShort**
Input : Collision-free path with safety distances
Output : Shortened collision-free path with safety distances

```
altered := TRUE                                          [ Modification flag ]
while altered                                                  [ Main loop ]
  while altered                                          [ Straighten Loop ]
    altered := FALSE
    for k = 1 ... segnum − 1                             [ All segment pairs ]
      if (CutTriangle(P̄_k, P̄_{k+1}) = SUCCESS)
        altered := TRUE
      endif
    endfor                                               [ All segment pairs ]
  endwhile                                          [ Any triangle was cut ]
  for k = 1 ... segnum                                  [ Intersection loop ]
    if (|| q_k − q_{k−1} ||> λ)
      Cut segment P̄_k in its center, creating 2 colinear segments
      segnum := segnum + 1
      k := k + 1                                         [ Skip new segment ]
      altered := TRUE
    endif
  endfor                                                 [ All path segments ]
endwhile                                          [ Any modification applied ]
```

6.2.5 Termination

The termination of the shortening of paths is controlled by the parameters ρ and λ. When the C-length of all segments falls below λ, and none of the segment pairs that form triangles that are relatively higher than ρ can be modified successfully, the planning is finished.

Both parameters must be greater zero to yield termination. For the segment length λ this is obvious. For the threshold ratio ρ consider three adjacent segments that possibly could be replaced by one straight connection. The repeated straightening can converge against the straight connection only asymptotically, thus a value $\rho > 0$ is required.

The straighten loop terminates, because a minimal non-zero decrease of the path length within each iteration is assured, lower bounded by the length of the shortest segment in the path and the threshold ratio ρ. As there is a minimal path length, the straighten loop cannot run forever. The intersection loop terminates, as the number of possible segments with a limited length along a (continuously shortened) path is finite. Thus the planning of short paths terminates for all possible combinations of $\rho > 0$ and $\lambda > 0$

6.3 Planning Examples

Evaluation of Planning Results

The paths with safety distances planned for the sample sceneries in the previous chapters are locally shortened. The planning success is measured relatively as the length ratio between initial and final path:

$$m_{short} = \frac{Q_{short}(\text{ initial path })}{Q_{short}(\text{ final path })}$$

This relative success depends on the quality of the initial path and does not really capture the quality of the result. But the length of the globally shortest path cannot be calculated, so no absolute measure of quality can be given. The "quality" of the final solution can be estimated by visually considering the motion of the robot in its workspace. The value of m_{short} roughly expresses the absolute success of the length reduction: the shorter the final path is in relation to the initial path, the more time and energy are saved.

The path length reduction will typically slightly *decrease* the measure of safety distance m_{dist} of a path, as the part of the motion in fully free space is mostly shortened.

Parameter Selection

The parameter ρ is easy to specify as it is obvious in meaning. A typical value chosen in the examples is 5%, i.e. triangles with a height/base relation of $\frac{1}{20}$ and below are not modified. The parameter λ is more difficult to chose as "length" in c-space is of a less intuitive meaning. Therefore a value λ_{joint} is used. It specifies the edge length of a hypersquare in C and λ is given by its diagonal:

$$\lambda = \sqrt{n\,{\lambda_{joint}}^2} = \sqrt{n}\lambda_{joint} \tag{6.1}$$

This allows to express the allowed lower length limit by the value of one joint. If the motion is assumed to occur more or less equal in all joints, the resulting segment length is comparable, independent of the number of degrees of freedom. A value that showed good results in practical experiments is $\lambda_{joint} = 10°$.

Translational joints are weightened heuristically, a translation of $1\,mm$ is treated as a turn by 1°. Note that more advanced possibilities exist for segment length limitation, see chapter 9.1, p. 153 for an extended discussion.

The 6-DOF Manipulator

The path length reduction for the 6-DOF manipulator is shown in fig. 6.3. The path is tightened and breaks down to the safety distance niveau. The straighten loop of the planning algorithm is passed 12 times, 40 modifications are applied, and the number of intersection points is increased from 7 to 16.

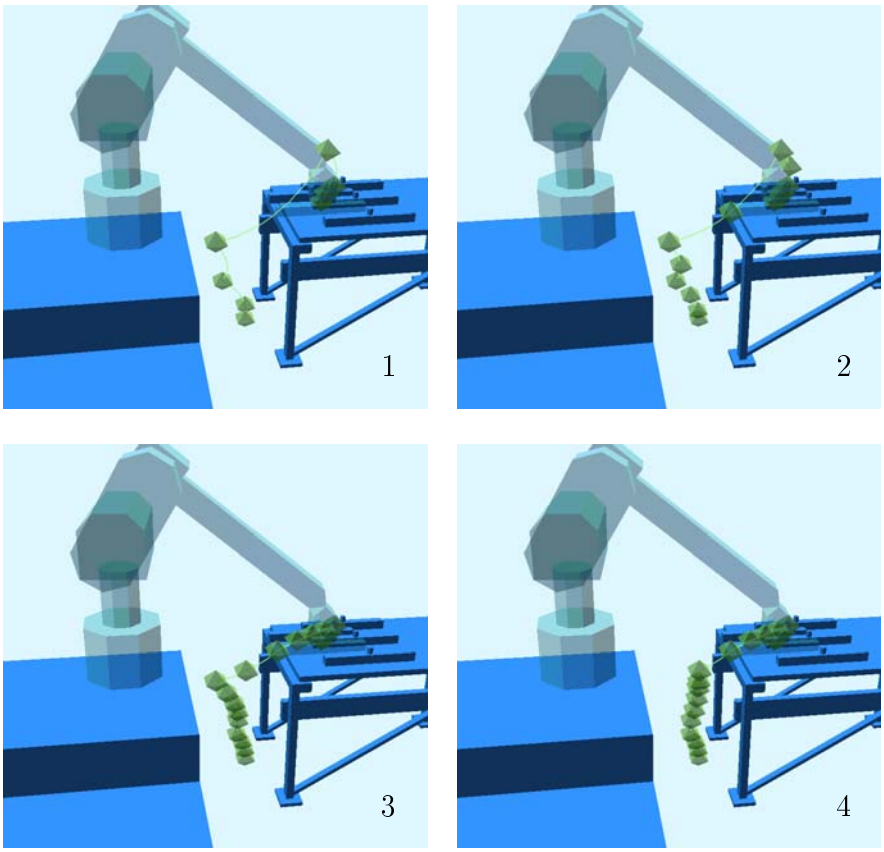

Figure 6.3: Path length reduction for the 6-DOF manipulator. For clarity, only the tip trace and the goal configuration are shown. 1: The initial path with safety distances. 2,3: Intermediate stages of the planning process. 6: The shortened path found as solution.

The length reduction is $m_{short} = 67.7\%$. This large reduction is a result of the reduced turns in the wrist and not very obvious for the tip trace shown in the pictures. The parameters were $\rho = 5\%$ and $\lambda = 10°$, and the example shows that very good results can be achieved with very little effort.

The resulting motion is shown in fig. 6.4. The overall quality is extremely high. The load is lifted in between the table and the robot's base, and it is adjusted in orientation immediately after the start. The turn into the final orientation takes place in a distance above the table, and finally the load is put down very straight.

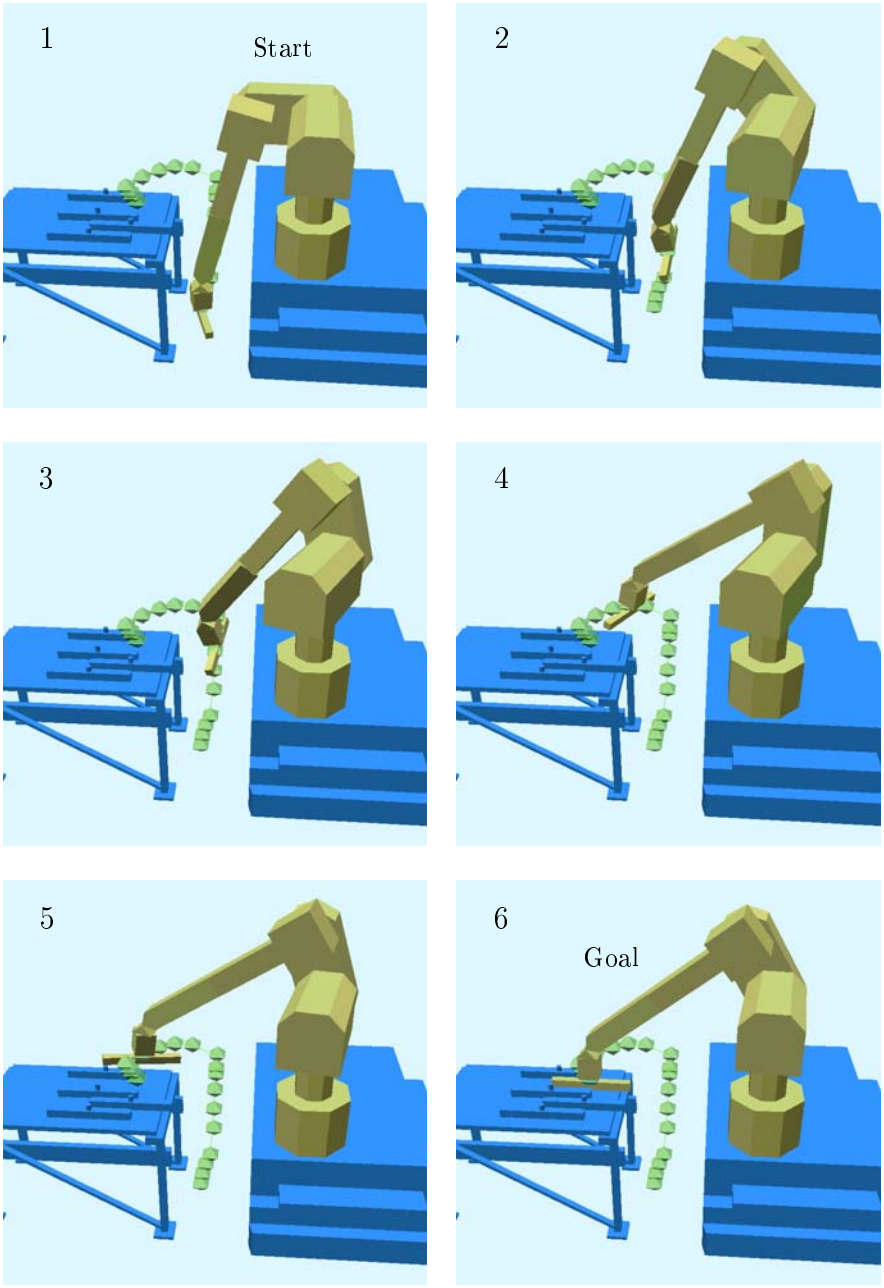

Figure 6.4: A sequence of configurations along the planned path. 1: Start. 2–5: Intermediate postures. The load is lifted straight up right in the center between the table and the robot's base. 6: Goal.

Note that the last approach appears as a detour in W, but is actually a short motion in C, as only joint two is turned.

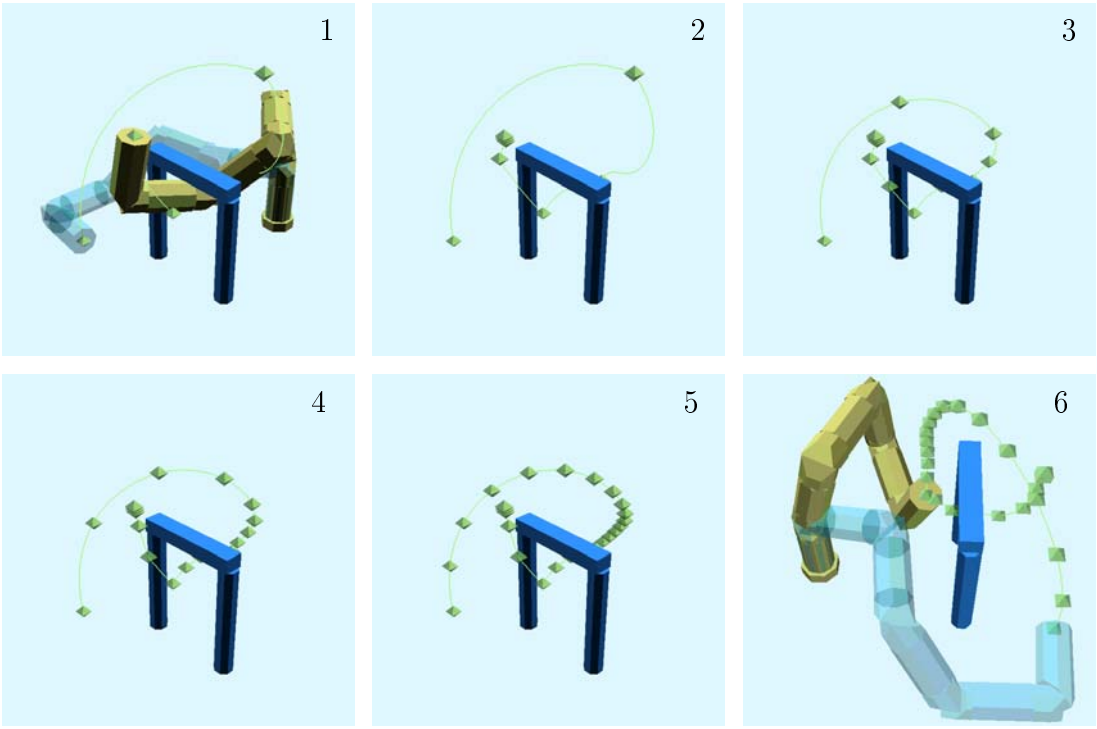

Figure 6.5: Path length reduction for the 16-DOF manipulator. 1: The initial path with safety distances. 2–5: Intermediate stages of the planning process. For clarity, only the tip trace is shown. 6: An intermediate configuration of the planned motion. Note that the scenery is shown in a different perspective as opposed to the previous images. After the manipulator with its safety distances has passed the gate, it turns towards the goal. It can be clearly seen that the foremost link keeps its safety distance when moving upwards in front of the gate.

The 16-DOF Manipulator

The path length reduction was applied with the same parameters to the path with safety distances that has been planned for the 16-DOF manipulator (see fig. 6.5.1). Ten iterations of the inner straighten loop of the planning algorithms are passed, and the number of intersection configurations is increased to 23. A total of 19 triangles are cut off the path (6.5.2 through 6.5.5). With this limited effort the path length is reduced to 84% of the original length. The final result is of high quality and shown in fig. 6.5.6.

6.4 Complexity and Computational Effort

The algorithmic complexity of path length reduction is solely dependent on the control parameters and the input path length, but independent from the number n of joints of the manipulator system. If measured analogous to collision-free path planning (see 4.6, p. 74) and planning of safety distances (see 5.4, p. 93), the planning of short paths has a constant complexity. The memory requirements are as small as for the other planning components: only one currently active path is kept in memory at a time.

The computational effort of the path length reduction is of higher practical interest. It depends on three factors:

- **The "quality" of the initial path in relation to the obstacles**
  Usually the paths created with the BB-method are relatively short, as planning is local and starts off with the shortest possible path, the straight line. In such a situation the shortening has to modify the path "not very much", and the effort is kept low, because the safety distances are in touch with the obstacles after just a few modification steps.
  The path may contain extreme detours if it was planned via a random subgoal. In this case, the expected effort increases as repeated modification steps will occur until the detour is more or less removed.

- **The segment length threshold λ**
  This is a preset control parameter. A reasonable choice for its value can only be found based on technical considerations. An upper maximum of a few hundred path segments appears reasonable even for very complex paths. But even fewer segments may be completely sufficient. The trajectory generation of a real robot may be able to highly optimize paths consisting of quite long segments. The extended optimization capabilities of such a system that consider the robot's dynamic behaviour might even be limited if the segments are too short.
  The planning effort is influenced by λ, as a higher number of segments will result in more modification steps. The resulting effort is proportional to the desired topological adaptability of the path shortening.

- **The threshold ratio ρ**
  A smaller value for this control parameter also increases the effort and the topological adaptability. The choice has to be based on technical considerations and quality requirements. The practical experiments show that relatively high values for both parameters result in relatively good paths which are found with extremely small effort.

The developed mechanism for the planning of short paths is a heuristic polygon modification under a constraint: do not decrease the safety distances. It allows to find paths of high quality with loittle effort, and the results are well suited to be passed to the trajectory generation of a real robot.

Chapter 7

Swept Volume Collision Detection and Path Rating

The fundamental evaluation of paths is the detection of collision. The volume that is swept out *by the robot in motion within the workspace has to be free of interferences with the obstacles in its environment, and the robot is not allowed to collide with itself.*

For this purpose a novel scheme is presented, the swept volume collision detection*, that explicitly models an approximation of the swept volume and checks it for collision. This scheme is a possible realization basis for the rating functions presented in the previous chapters, namely Q_{scale} and Q_{dist}.*

The swept volume collision detection for linear path segments in c-space is developed stepwise. An algorithm to calculate intermediate configurations close enough to limit the rotational divergency with respect to a tolerance parameter τ is the base for the workspace-linear sweeping of slightly expanded geometry models. The calculation of expanded models based on polyhedrons is described and integrated into a collision detection algorithm.

The chapter ends with a discussion of the complexity and the computational effort required for collision detection and the rating functions. It is shown that swept volume collision detection reduces the number of required facet intersection tests if compared to a sophisticated discretization scheme.

7.1 Intention and Approach

The use of polygonal paths reduces the collision detection problem to the collision detection for path segments $\bar{\mathsf{P}}$ linear in C. This elementary problem will be addressed first and then extended to a realization of the rating functions.

Considering a Single Point

To test an entire robot – moving along a path segment – for collision, principally each point of its model has to be tested for interference at each possible configuration (an infinite number).

One point $\mathbf{u}$ within a link i of a robot moves along a curve in W if the robot moves along a C-straight path segment $\bar{\mathsf{P}}$ due to the accumulated rotations around

the previous joints I_i. The analytic description of this curve in W is complex, and to use it explicitly for collision detection would result in unbearable high computational costs.

Typically two discretization methods are employed in the motion planning literature. Both have several major drawbacks:

- One possible way is to slightly expand the robot's geometry model in such a way that for the point $\mathbf{u}$ the sphere $\mathbf{u}^{+\tau}$ is covered. If the path segment is discretized with intermediate configurations so close together that no cartesian motion is longer than 2τ for $\mathbf{u}$, configuration-wise collision detection at the intermediate points is a sufficient test for the entire path segment. The main problem with this method is that it requires a lot of tests, especially if high precision motions close to obstacles are desired. Schemes like this are used e.g. in Glavina (1991), Overmars and Svestka (1994).
- Another possibility is to discretize the path segment dynamically. At each intermediate configuration the closest distance between the moving robot and the obstacles is calculated, and a step along the path segment to the next intermediate configuration is estimated that assures a motion below this distance. This scheme requires distance calculations between moving objects, which is computationally a lot more expensive than interference detection (see e.g. Quinlan (1994)).

It is worth noting that the problem of correct collision detection is often neglected in the motion planning literature. Heuristic stepsizes are used in the c-space, based on the assumption that either no collision is missed or is "minor and can be ignored" (Chen and Hwang (1992)).

The objectives for the collision detection suggested here are to use only a few intermediate discretization configurations to minimize the required effort, to avoid expensive distance computation and to enable motions very close to obstacles. This should be realized without detecting too many false collisions due to rough discretization, and without missing any real collision. The idea followed to achieve these objectives is to approximate the C-linear path segment by a sequence of W-linear line segments. For one point $\mathbf{u}^{+\tau}$, intermediate configurations have to be found that the W-linear sweeping in between covers the real curved line segment of point $\mathbf{u}$. The idea is illustrated in fig. 7.1. The sketch shows the expected reduction of computational effort in comparison to discretized steps.

Testing a Manipulator System

Based on the collision condition (3.2), a basic collision detection algorithm using the swept volume has the following form:

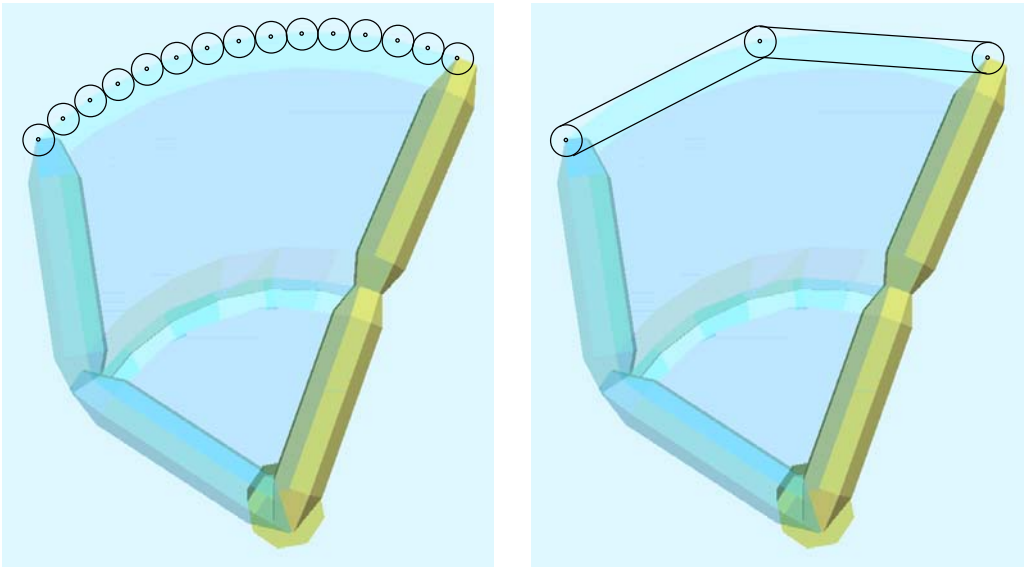

Figure 7.1: Left: A dedicated point of the robot is tested for collision by testing a suitably dense sequence of intermediate configurations with an expanded point. Right: The linear sweeping of the expanded point between just three configurations covers the real curve of the point with less effort.

Algorithm : **SegmentColl**
Input : $\bar{\mathbf{P}}$
Output : COLLIDING or FREE

for $i = 2 \ldots n$
  for $j = 0 \ldots i-1, j \neq v_i$
$*$ **if** $(\exists \mathbf{q} \in \bar{\mathbf{P}} : {}_j\mathsf{T}_i(\mathbf{q})\, G_i \cap G_j \neq \emptyset)$
    return COLLIDING
  endif
  endfor
endfor
return FREE

The return value of **SegmentColl** indicates collision or no collision. The selfcollision between links that are not connected by a joint has to be done only once for each pair. Within **SegmentColl**, it is checked "backwards", i.e. for each link only the links with smaller indices are tested. This algorithm is exact, i.e. it detects a collision if and only if the motion yields a collision.

The realization of the swept volume collision detection has to replace line $(*)$. The relative motion of a link i with respect to an object j (either the static

environment, if $j = 0$, or another link) results in a swept volume that will be approximated.

The following terms are used: an **intermediate configuration** of a path segment is a configuration on the path segment itself. An **intermediate segment** is a segment connecting two intermediate configurations and thus a part of the whole segment.

To realize the swept volume approximation a number ${}_jz_i$ of intermediate path segments $\bar{\mathbf{P}}_{part}$ is calculated that allows to cover the swept volume of the real, curved motion of G_i by the W-linear sweeping of $G_i^{+\tau}$. For each intermediate path segment the following condition has to hold if the intermediate segment $\bar{\mathbf{P}}_{part}$ connects the configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$:

$$\bigcup_{\mathbf{q}\in\bar{\mathbf{P}}_{part}} {}_j\mathbf{T}_i(\mathbf{q})G_i \;\subset\; \bigcup_{\substack{t\in[0,1]\\ \mathbf{u}\in G_i^{+\tau}}} ({}_j\mathbf{T}_i(\mathbf{q}_{k-1})\mathbf{u})\,t + ({}_j\mathbf{T}_i(\mathbf{q}_k)\mathbf{u})\,(1-t)$$

The allowed **tolerance** τ is a preset control parameter that allows the scaling of precision. A smaller value will result in a closer approximation of the real swept volume, requiring more intermediate segments. To keep the computational effort low, the number of intermediate segments must be as small as possible for a given τ.

7.2 Discretizing a Path Segment

C-Step Size Conditions

To find a suitable discretization, the rotational divergency along an intermediate segment $\bar{\mathbf{P}}_{part}$ between two intermediate configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$ is overestimated. This is done by assuming the worst case for a certain *direction* in C, considering only the relative motion $\mathbf{q}_k - \mathbf{q}_{k-1}$, instead of the actual configurations and the real positions of the links in the workspace.

Each pair of objects j, i within a manipulator system is connected by a linear kinematic chain, and only a subset of the n configuration components contribute to the relative motion.

To simplify the following formulas it is assumed that all rotational joints of this relative linear chain are labeled with indices from 1 up to h, thus joint h moves link i, and the object at the beginning of the linear chain is object j. Translational joints do not contribute to rotational divergency and do not need to be regarded as movable, they contribute to the effective radii only. Thus the step size condition is made up without loss of generality for the C-step $\Phi = (\phi_1, \ldots \phi_h)$.

The **effective radii** r_l used for the step size conditions are defined as follows:

- r_h is the maximum distance of any point within G_i from the origin of joint h. This radius will always be larger than the tolerance, i.e. $r_h \gg \tau$.

- r_l, $l = 1, \ldots, (h-1)$ is the maximum distance between the joint origin of joint l and the origin of the attached joint $l+1$. If there actually are translational joints between these two rotational joints l and $l+1$, they are considered to be stretched to maximum extent, thus contributing to the radius by the largest possible amount.

The maximum effective radius ${}_jr_i$ is set to the sum of all effective radii:

$$ {}_jr_i \;=\; \sum_{l=1}^{h} r_l $$

This value overestimates the maximum possible radius of any point within link i with respect to the object j, as the relative orientation of the joint axes is not considered. All axes are assumed to be parallel, the kinematic chain can be seen as stretched onto a line.

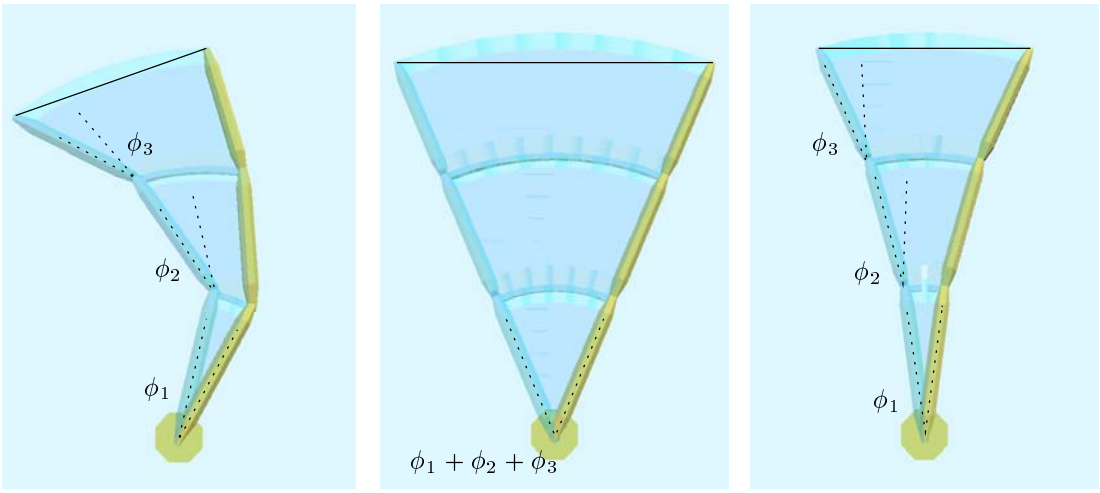

Figure 7.2: Illustration of the C-step conditions. Left: C-step Φ applied at an arbitrary configuration and the implied rotational divergency. Center: All components are summed up and applied with maximum radius. The rotational divergency is overestimated by far. Right: Each component of the C-step has its maximum possible effect on the rotational divergency if the kinematic chain moves through full stretch. The rotational divergency is overestimated.

In the case of just one joint, $h = 1$, that turns one link of radius r_1, the maximum rotational divergency can be exactly calculated and gives a condition for the respective angle ϕ_1 (of course, only angles $\mid \phi_1 \mid < \pi$ are considered):

$$ r_1 \left(1 - \cos\frac{\phi}{2}\right) \;\leq\; \tau \qquad (7.1) $$

This can be extended for the general case $h > 1$ of several joints by assuming that all rotation occurs with maximum radius. The absolute value of all components of Φ is summed up and "turns" with ${}_jr_i$. If the resulting divergency is smaller than τ, Φ cannot result in a larger divergency (with $\sum_{l=1}^{h} \mid \phi_l \mid < 2\pi$) for any point in link i independent of the actual configuration:

$$ {}_jr_i \left(1 - \cos\left(\frac{\sum_{l=1}^{h} \mid \phi_l \mid}{2}\right)\right) \leq \tau \tag{7.2} $$

This is a very strong condition, a better one – allowing larger C-steps – can be found based on (7.1). Only the first joint angle ϕ_1 rotates with maximum radius ${}_jr_i$, higher labeled joints imply less divergency and need not to be considered with the full radius as in (7.2). This can be written as:

$$ {}_jr_i - \sum_{l=1}^{h} \left(r_l \cos\frac{\sum_{m=1}^{l} \mid \phi_m \mid}{2}\right) \leq \tau \tag{7.3} $$

This condition can be explained as follows: if the motion of a kinematic chain leads from contraction through maximal extent and again into contraction, the total divergency is the highest possible one. All joint divergencies sum up, none of the rotations compensate each other. For any C-step Φ that fulfils (7.3), no point within link i can get further away from the W-straight connection between an arbitrary configuration $\mathbf{q}$ and the configuration $\mathbf{q} + \Phi$. The C-step conditions are illustrated in fig. 7.2.

Calculating the Number of Intermediate Segments

The conditions above are used to calculate a number z of required intermediate segments to discretize a path segment $\bar{\mathbf{P}}$ that connects $\mathbf{q}_a$ and $\mathbf{q}_b$. With

$$ \Phi = \frac{1}{z}(\mathbf{q}_b - \mathbf{q}_a) \tag{7.4} $$

the inequation (7.3) defines up a condition for z. The smallest integer value $z^{(solve)}$ for z that fulfils this condition is the suitable number of intermediate segments. It cannot be calculated directly, thus a bisection scheme is used to find this value.

An initial guess $z^{(high)} \geq z^{(solve)}$ is found by resolving (7.2), using (7.4):

$$ \Rightarrow \quad z^{(high)} = \frac{\sum_{l=1}^{h} \mid q_{b_l} - q_{a_l} \mid}{2 \arccos\left(1 - \frac{\tau}{{}_jr_i}\right)} \tag{7.5} $$

This formula can be explained as follows: it calculates the number of edges of equal length that are required to approximate a partial circle of radius ${}_jr_i$ with

a maximum radial error of τ. The sum of the components of the C-step implied by $z^{(high)}$ is certain to be much smaller than π, as ${}_jr_i \gg \tau$.

$z^{(low)} = 0$ is the second initial value used for the bisection, $z^{(low)} < z^{(solve)}$. The bisection repeatedly cuts the interval $[z^{(low)}, z^{(high)}]$ in its center:

$$z^{(mid)} := \frac{z^{(high)} - z^{(low)}}{2}$$

If $z^{(mid)}$ results in a step size that fulfils (7.3), $z^{(mid)}$ replaces $z^{(high)}$, otherwise $z^{(mid)}$ replaces $z^{(low)}$. If $z^{(high)} - z^{(low)} \leq 1$, the bisection terminates, and $z^{(high)}$ is rounded up to the solution $z^{(solve)}$.

The values handled within this bisection are very small, i.e. it requires only a few iterations to terminate. To illustrate this: If a manipulator with an effective radius of $2\,m$ is considered, the tolerance shall be $5\,mm$ and all joint rotations sum up to 2π within the path segment, the initial value $z^{(high)}$ for the bisection is below 50. And if all this rotation takes place in the wrist joints with a radius of $50cm$, the value $z^{(solve)}$ that is found within a maximum of 8 iterations will be around 23. The cheap bisection reduced the effort of swept volume approximation by a factor of roughly 0.5, as only half the number of intermediate segments is needed.

Calculating Intermediate Configurations

The above algorithm is used to calculate the number ${}_jz_i$ for each pair j, i of objects that are to be tested along a path segment $\bar{\mathbf{P}}$, connecting $\mathbf{q}_a$ and $\mathbf{q}_b$. Based on this number, equal-length intermediate segments $\bar{\mathbf{P}}_{part}$ along $\bar{\mathbf{P}}$ are given by the resulting sequence of ${}_jz_i + 1$ intermediate configurations $\mathbf{q}_k$, $k = 0, \ldots, {}_jz_i$:

$$\mathbf{q}_k = \frac{k}{{}_jz_i}(\mathbf{q}_b - \mathbf{q}_a) + \mathbf{q}_a$$

There are two special cases where no intermediate configurations beside the first and last configurations of the path segment are required:

- If there are translational joints only between link j and link i, ${}_jz_i$ is set to one as the relative motion along the path segment is equal to the linear sweeping.
- If the segment length is zero, i.e. $\mathbf{q}_a = \mathbf{q}_b$, ${}_jz_i$ is set to zero and no intermediate configurations are calculated. No explicit sweeping has to be tested, the collision detection reduces to the test of a single configuration.

7.3 Sweeping Polyhedral Objects

7.3.1 Principle Algorithm

Using the intermediate configurations, the condition in line $(*)$ in the algorithm **SegmentColl** can be replaced by a call of the following algorithm:

Algorithm : **SweptObjColl**
Input : link numbers i, j; path segment $\bar{\mathbf{P}}$
Output : TRUE or FALSE

for $k = 1 \ldots {}_j z_i$
$**$ **if** $(\bigcup_{\substack{t\in[0,1]\\ \mathbf{u}\in G_i^{+\tau}}} (({}_j\mathbf{T}_i(\mathbf{q}_{k-1})\mathbf{u})\,t + ({}_j\mathbf{T}_i(\mathbf{q}_k)\mathbf{u})\,(1-t)) \cap G_j \neq \emptyset)$
return TRUE
endif
endfor
return FALSE

Using this approximation the collision detection is sufficient, i.e. if there is a collision it is detected, but if an obstacle is too close, a false collision is indicated.

To be able to realize line $(**)$, means have to be developed to expand geometry models algorithmically and to explicitly test their linear sweeping for collision.

7.3.2 Expanded Geometry Models

For path planning polyhedral geometry models are used. They consist of vertices, edges and facets. The vertices are specified in the local frame. Edges and facets are just structural information.

To efficiently expand a polyhedral model G by a possibly changing size τ, principally each vertex $\boldsymbol{\nu}$ gets an individual displacement vector $\mathbf{p}_\nu$ of proper size and direction. The displacement vectors are multiplied with τ and added to the vertices. The resulting expanded polyhedral geometry model $\bar{G}^{+\tau}$ – consisting of the shifted vertices and using the original structural information – bounds the isotropically expanded geometry model $G^{+\tau}$.

To enable this expansion by vertex displacement a polyhedral model has to be locally preprocessed. Besides the necessary calculation of the displacement vectors the structural information has to be adapted. The preprocessing can be summarized as follows:

- If more than three facets intersect in a vertex insert either a new edge (in the case of 4 facets) or a new facet of zero size with outward-bound normal (more than 4 facets). This is done by copying the vertex information and accordingly adapting the structural information. The resulting structure has exactly three facets intersecting within each vertex.

- For each vertex $\boldsymbol{\nu}$ shift all adjacent facets by unitlength in the direction of their normals and intersect the supporting planes. The difference between the intersection point and $\boldsymbol{\nu}$ constitutes $\mathbf{p}_\nu$.

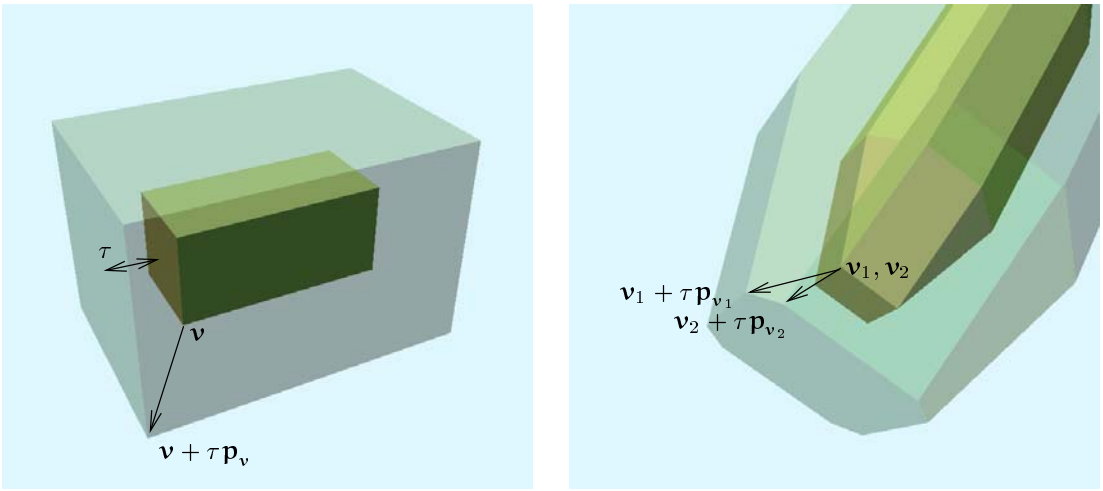

Figure 7.3: Polyhedral models expanded by vertex displacement. Left: If three facets intersect in a vertex $\boldsymbol{\nu}$, a displacement vector $\mathbf{p}_\nu$ can be found that moves all facets outwards equidistantly. Right: If more than three facets intersect the vertex has to be copied. In case of four intersecting facets an additional edge is inserted and two displacement vectors are calculated.

This kind of preprocessing is easily realized and does not require complex computations. It is obvious that $\bar{G}^{+\tau}$ bounds $G^{+\tau}$. In effect, the facets of the original model are pushed outwards to form $\bar{G}^{+\tau}$, see fig. 7.3 for illustration. "Sharp" edges and vertices are shifted much further away than necessary to fulfil the criteria of bounding $G^{+\tau}$. To prevent this, additional facets can be added within the preprocessing.

7.3.3 Collision Detection for Swept Edges

The linear sweeping of the expanded polyhedral model $\bar{G}_i^{+\tau}$ between two intermediate configurations has to be tested for collision with the static geometry model G_j. As usually there are rotations involved, this sweeping is not necessarily bounded by planar facets. The collision detection between polygonal facets can be implemented very efficiently, thus it is intended to use this mechanism for the sweeping as well. To enable this, the linear sweeping is approximated by a bounding model of planar facets. This is constructed dynamically, based on the edges of $\bar{G}_i^{+\tau}$.

For each edge the convex hull of its linear sweeping is used for the intersection detection, see fig. 7.4 for illustration. Taking one edge given by $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ out of the original geometry model, and using two intermediate configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$, the following four points are calculated:

$$\begin{aligned}
\mathbf{u} &= {}_j\mathbf{T}_i(\mathbf{q}_{k-1})\,(\boldsymbol{\nu}_1 + \tau\mathbf{p}_{\nu_1}) \\
\boldsymbol{w} &= {}_j\mathbf{T}_i(\mathbf{q}_{k-1})\,(\boldsymbol{\nu}_2 + \tau\mathbf{p}_{\nu_2}) \\
\mathbf{u}' &= {}_j\mathbf{T}_i(\mathbf{q}_k)\,(\boldsymbol{\nu}_1 + \tau\mathbf{p}_{\nu_1}) \\
\boldsymbol{w}' &= {}_j\mathbf{T}_i(\mathbf{q}_k)\,(\boldsymbol{\nu}_2 + \tau\mathbf{p}_{\nu_2})
\end{aligned}$$

The convex hull is formed by the four facets $\mathbf{f}_1, \ldots, \mathbf{f}_4$:

$$\mathbf{f}_1 = (\mathbf{u}, \boldsymbol{w}, \mathbf{u}'),\ \mathbf{f}_2 = (\mathbf{u}, \boldsymbol{w}, \boldsymbol{w}'),\ \mathbf{f}_3 = (\mathbf{u}, \mathbf{u}', \boldsymbol{w}'),\ \mathbf{f}_4 = (\boldsymbol{w}, \mathbf{u}', \boldsymbol{w}')$$

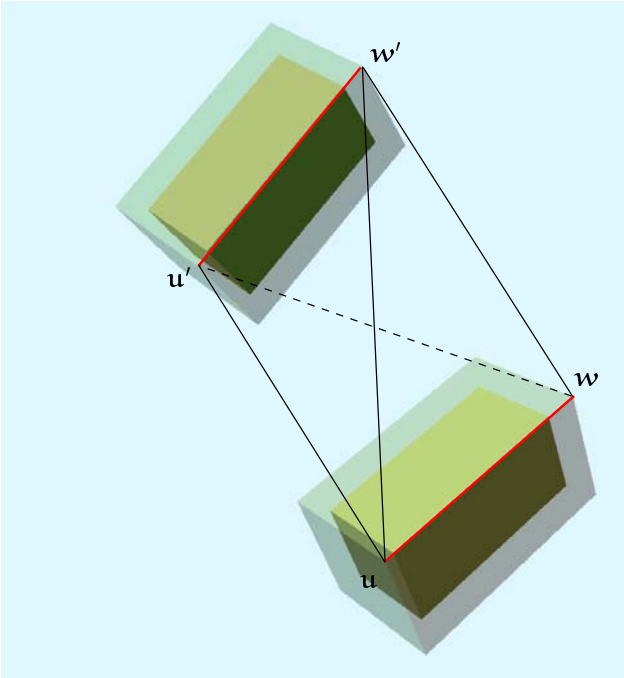

Figure 7.4: To test a bounding volume of the linear sweeping of an expanded model, the convex hull of the sweeping of each edge of the expanded model is tested. It is formed by the edge's vertices at the intermediate configurations and consists of four facets.

By construction these facets cover the linear sweeping of the respective edge. The following algorithm **SweptStepColl** realizes the collision detection of the linear sweeping between two intermediate configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$. Each

convex hull that bounds the swept edges of the expanded polyhedral model of link i is tested with each facet of the model of j.

Algorithm : **SweptStepColl**
Input : intermediate configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$
Output : TRUE or FALSE

for all edges $\boldsymbol{e}_i$ in G_i
  for all facets $\mathbf{f}_j$ in G_j
    if convex hull of expanded $\boldsymbol{e}_i$ for
        configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$ intersects $\mathbf{f}_j$
      return TRUE
    endif
  endfor
endfor
return FALSE

Collision Detection for Single Configurations

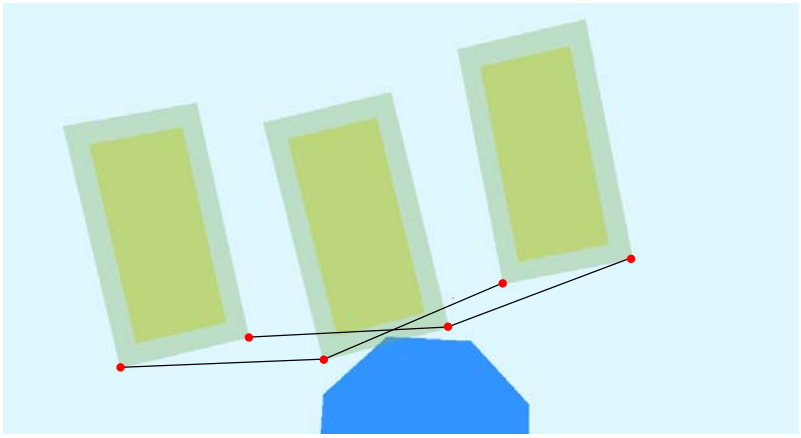

Figure 7.5: Possible collisions might be missed if the expanded model itself is not tested at the intermediate configurtions. The figure shows an object at three different intermediate configurations. Testing the intermediate configurations as well as the swept edges closes the approximative model of the swept volume.

Testing the convex hull of each facet does not cover the linear sweeping completely, because the tested model is not *closed* at the configurations $\mathbf{q}_{k-1}$ and $\mathbf{q}_k$. The situation is sketched in fig. 7.5. It is possible that the moving model reaches into the static object and a collision is not detected. To allow for this, the expanded model itself has to be tested at each intermediate configuration as well.

This is done by intersecting all facets of the respective models. An algorithm **ObjColl** works analogous to **SweptStepColl**: it tests all facets of the expanded polyhedral model of link i with each facet of the model of link j.

7.3.4 Collision Detection for Swept Polyhedral Objects

By using both **SweptStepColl** and **ObjColl**, the algorithm **SweptObjColl** can be rewritten as follows:

Algorithm : **SweptObjColl**
Input : link numbers i, j; path segment $\bar{\mathbf{P}}$
Output : TRUE or FALSE

```
if ObjColl(q_a)
  return TRUE                                  [ First configuration collides ]
endif
for k = 1 ... jz_i
  if SweptStepColl(q_{k-1},q_k)
    return TRUE                                [ Intermediate segment collides ]
  endif
  if ObjColl(q_k) collides
    return TRUE                                [ Intermediate configuration collides ]
  endif
endfor
return FALSE                                   [ No collision ]
```

This realization tests a polyhedral model that bounds the sweeping of the real swept volume – and of course a lot of facets within this model. By construction it is safe, i.e. possibly it detects false collisions and it does not miss real collisions. The model created by shifting the facets of a polyhedral model outwards with a certain preset tolerance τ bounds the expanded geometry model. The step size used and the convex approximation of the swept edges assures that no single point of the original model ever gets "outside" the polyhedral swept volume along the path segment.

SweptObjColl can be used within **SegmentColl** (see 7.1, p. 107 above), to calculate the collision of a complete manipulator system, i.e. all relative object motions are tested.

7.4 Realization of the Rating Functions

7.4.1 Intention and Approach

The rating functions Q_{scale} (see p. 51) and Q_{dist} (see p. 79), used to measure the required scaling factor for collision-free motion and the maximum possible

expansion respectively, are an extension to collision detection. They both involve a maximization operation for the swept volume of a moving link.

The rating functions are used to compare different path segments. The alternative segments are calculated in a way that the sweeping of the link is displaced a certain cartesian distance $\geq \delta_{\min}$ (at least at one end of the segment). For comparison, it is sufficient to approximate the rating if the approximation is of adequate precision.

Approximation in case of the size-changing rating functions means that only a limited number of different sized models is considered for the tests. The scaling/expansion of the largest collision-free model found in this limited set is used as rating approximation. As the desired precision is related to the motion of the link in the workspace, it is controlled by two preset control parameters γ_{scale} and γ_{dist} that express the "difference of the size" of the models. These *granularities* are expected to be equal or smaller than the smallest displacement stepsize $\delta_{\min}$ to allow for a comparison between alternative paths. Different values are used for collision-free path planning and safety distance planning as safety distances typically require a finer adaption, i.e. shorter alternative paths with smaller displacements have to be compared.

This difference of model size is obvious for the expansion of links. For a given link i and its desired safety distance $d_{\max_i}$ an integer value m_{exp} can be calculated by rounding up $\frac{d_{\max_i}}{\gamma_{dist}}$. With this number a list of possible expansions $d_k,\ k = 0, \ldots, m_{exp}$ can be calculated that implicitly defines a set of $m_{exp} + 1$ models that are about γ_{dist} different in size:

$$d_0 = 0,\ d_1 = \frac{d_{\max_i}}{m_{exp}},\ d_2 = \frac{2d_{\max_i}}{m_{exp}},\ \ldots,\ d_k = d_{\max_i}$$

For the scaling the effective radius r_i of link i (i.e. the largest distance of any point within the model of link i from its local origin) is used to calculate m_{scale} as the rounded up value of $\frac{r_i}{\gamma_{scale}}$. A set of $m_{scale} + 1$ models is implicitly defined by the following list of scaling factors $s_k,\ k = 0, \ldots, m_{scale}$:

$$s_0 = 0,\ s_1 = \frac{1}{m_{scale}},\ s_2 = \frac{2}{m_{scale}},\ \ldots,\ s_{m_{scale}} = 1$$

The models scaled with these factors have a maximum size difference of γ_{scale}.

To find the appropriate maximum value within this list, a bisection scheme based on integers is used. To calculate the largest possible scaling factor between $s_0 = 0$ and s_k the bisection principally works as follows:

Algorithm : **BisectScale**
Input : s_k from the list of possible scaling factors
Output : s_{low}, highest collision-free value $\leq s_k$

if (model collides if scaled with factor s_k)
  $high := k$
  $low := 0$
  while $(high - low > 1)$ *[Bisection loop]*
    $mid =$ rounded down value of $\frac{high+low}{2}$
    if (model collides if scaled with factor s_{mid})
      $high := mid$
    else
      $low := mid$
    endif
  endwhile
  return s_{low}
endif *[No collision for scaling s_k]*
return s_k

This algorithm can be used with the algorithms **ObjColl** and **SweptStepColl** as underlying collision detection where the scaling factor is assumed to be applied to the transform of link i, i.e. ${}^{s_k}_{i}\mathbf{T}_j$ is used instead of ${}_i\mathbf{T}_j$. This allows estimation of the largest possible scaling factor of objects at single locations, or of linear swept objects between two intermediate configurations.

Typically the order of the bisection depth is below 10. Given a link of $1\,m$ length, and a granularity $\gamma_{scale} = 5$, the number of possible differently scaled models is 200, and thus the bisection depth would be 8.

7.4.2 Realization of Q_{scale}

Within Q_{scale} the maximization of the scaling factor for a geometry model sweeping along a path segment $\bar{\mathbf{P}}$ has to be found:

$$scale \;=\; \max_{s\in[0,1]} \left(\forall \mathbf{q} \in \bar{\mathbf{P}} \;:\; {}^{s}_{j}\mathbf{T}_i(\mathbf{q})\, G_i \cap G_j \;=\; \emptyset\right)$$

For the models of the links holds ${}^{s_1}\mathbf{T}G_i \subset {}^{s_2}\mathbf{T}G_i$ for $s_1 < s_2$. Therefore it is possible to test different intermediate segments with different scaling factors, as the segment tested with the larger scaling factor without colliding is also collision-free for any smaller scaling factor. To calculate the scaling of a link, an algorithm that operates on the same principle as **SweptObjColl** is used. It starts with the highest possible scaling, $s_{m_{scale}} = 1$, works itself along the path segment, and if a collision occurs, the scaling is reduced using the bisection scheme. The algorithm has the following form:

Algorithm : **SweptObjGetScale**
Input : link numbers i, j; path segment $\bar{\mathbf{P}}$
Output : *scale*

scale := $s_{m_{scale}}$ *[Start with the full size,* $s_{m_{scale}} = 1$ *]*
scale := **BisectScale**(*scale*) used with **ObjColl**($\mathbf{q}_a$)
for $k = 1 \ldots {}_j z_i$
  scale := **BisectScale**(*scale*) used with **SweptStepColl**($\mathbf{q}_{k-1}$,$\mathbf{q}_k$)
  scale := **BisectScale**(*scale*) used with **ObjColl**($\mathbf{q}_k$)
endfor
return *scale*

This algorithm calculates an approximation of an approximation: The sweeping is approximated by a slight expansion τ, the scaling uses discrete levels induced by γ_{scale}. As the expansion that covers the tolerance is also scaled, the approximation might even be faulty for scaling factors < 1, i.e. the step size condition might not be fulfilled. These "errors" can be neglected, as they do not occur for full size models and are minor otherwise.

The position of minimal scaling $\mathbf{q}_{min}$ is also only approximated. Whenever the bisection reduces the scaling factor within the main loop of **SweptObjGetScale**, the value of $\mathbf{q}_{min}$ is updated: if the scaling is reduced for a single intermediate configuration, its value is taken for $\mathbf{q}_{min}$. If the scaling is reduced within an intermediate segment, the center configuration of the respective segment is taken as the value for $\mathbf{q}_{min}$.

7.4.3 Realization of Q_{dist}

The rating function Q_{dist} requires the collision detection for models with safety distance and the maximization of the safety distance possible for a certain link.

Collision Detection for Models with Safety Distances

The following intersection test is used within Q_{dist}:

$$\forall \mathbf{q} \in \bar{\mathbf{P}} \; : \; {}_j\mathbf{T}_i(\mathbf{q})\, G_i^{+d_{\bar{\mathbf{P}}i}} \cap G_j$$

It can be realized by adding the expansion $d_{\bar{\mathbf{P}}i}$ to the tolerance τ if the approximative expanded model is constructed and tested for the sequence of intermediate configurations (see **SweptObjColl** above). This is a safe collision detection: no collision for the real sweeping of the isotropically expanded model $G_i^{+d_{\bar{\mathbf{P}}i}}$ will be missed, if the test is executed with the expanded polyhedral model $\bar{G}_i^{+d_{\bar{\mathbf{P}}i}+\tau}$.

Maximizing the Safety Distance

The safety distance has to be maximized for a dedicated link. The value *dist* is defined as follows:

$$dist \;=\; \max_{d\in[0,d_{\max_i}]} \Big(\forall \mathbf{q} \in \bar{\mathbf{P}} \;:\; {}_j\mathbf{T}_i(\mathbf{q})\, G_i^{+d} \cap G_j \;=\; \emptyset\Big)$$

This can be calculated approximatively by using schemes analogous to the ones employed for the scaling factor calculation. A bisection that works with the list of possible distance values is used in conjunction with the collision detection for intermediate segments and intermediate configurations. It is realized analogous to **BisectScale**. In turn, this can be used within an algorithm that works as the algorithm **SweptObjGetScale** above: it starts off with the highest safety distance and works itself along the sequence of intermediate configurations and segments, reducing the safety distance with the bisection if necessary to avoid collisions. The position of minimal distance $\mathbf{q}_{min}$ can also be approximated as it is done within Q_{scale}: whenever the bisection reduces the current safety distance, the respective configuration or the respective midpoint of the intermediate segment is chosen.

With the mechanisms developed for swept volume collision detection an effective realization of the rating functions based on *virtually modified* geometry models is achieved. It should be noted that it is just *one* possible realization – but a very efficient one.

7.5 Complexity and Computational Effort

The following consideration of complexity and computational effort analyses the algorithms employed for collision detection and the rating functions. It is shown that the swept volume approach is superior to the collision detection at intermediate configurations with respect to the required number of facet intersection tests.

7.5.1 Manipulator Collision Detection at a Single Configuration

Two main elements are involved in the collision detection: the geometry model of the environment and the geometry model of the manipulator. The modelling complexity is given by the number of geometric primitives, i.e. the number of facets, edges and vertices that make up the geometry models. If c_0 denotes the modelling complexity of the environment, and c_m denotes the modelling complexity of the robot, the complexity of collision detection at a single configuration is overestimated as follows:

$$O(\ \textit{Collision Test at Configuration}\ ) \;=\; O(\ c_0 \times c_m \;+\; c_m \times c_m\ )$$

The robot has to be tested with the environment and with itself to detect possible selfcollisions. This quadratic term is a worst-case estimate, not every single pair of geometric primitives has to be tested. It also depends on the number of links but the inherent complexity of the modelling is the dominant factor.

An intelligent implementation uses every possibility to reduce the effort without missing real collisions. Looking at typical robots usually many pairs of links exist that can never intersect by construction and thus do not need to be tested explicitly, contrary to the general concepts presented in this work. This can be implemented by a boolean matrix stored with the robot kinematics, which marks the pairs of links that do not need to be tested. Note that it would be very expensive to calculate this matrix, as it requires a complete evaluation of the c-space in the general case. For a human operator or robot constructor, this relation is obvious for most link pairs.

7.5.2 Collision Detection for Path Segments

Motion planning has to consider paths – path segments in the case of the BB-method – instead of single configurations. The resulting complexity depends on two main factors: the path segment length $l_{\bar{\mathbf{p}}}$, and, as an approximation scheme is used, the tolerance parameter τ. Based on these considerations the complexity of the collision detection for path segments can be written as:

$$O(\text{ Collision Test for Path Segment }) = O(\,(c_0 \times c_m + c_m \times c_m) \times f(\,l_{\bar{\mathbf{p}}},\,\tau\,)\,)$$

The function f is obviously linear with respect to the segment length $l_{\bar{\mathbf{p}}}$, i.e. the effort doubles if the length is doubled. And if an appriximation scheme is used that is based on the segment length, e.g. collision detection for expanded geometry models at intermediate configurations which are calculated in a proper distance, f is approximately linear with respect to $\frac{1}{\tau}$. This is obvious: reducing the tolerance by half will require twice the number of collision tests.

This does not hold true for the swept volume collision detection. Its approximation scheme is based on the *curvature* of the path segments. And in result it is *sublinear* with respect to $\frac{1}{\tau}$. If the tolerance is reduced by half, the effort is less than doubled. It can be seen from (7.5) (see p. 112): if τ is reduced by half, the denominater on the left-hand side is less than halfed due to the arccos. This is a valuable property of the swept volume collision detection, as it allows to increase the precision of the approximation with a sublinear increase of the computational effort.

To show the gain in efficiency the swept volume collision detection is compared to discretized steps. For both schemes the number of facets that need to be tested for intersection with the environment to check a path segment is calculated.

In the case of discretized steps it is assumed that each link moves 2τ in W between each test, i.e. the travelled cartesian distance for each link is divided

by 2τ, and the rounded up result is multiplied with the number of facets of the link model. This is summed up for all links and results in the total number of facets that have to be tested for the path segment. Note that this is the smallest number of facets that is possible with discretized steps, as an *optimal* discretization is assumed.

In the case of swept volume collision detection, the respective number is found by multiplying the number ${}_0z_i$ with the number of facets – to consider the tests at the intermediate configurations – plus four times the number of edges – as each edge requires four facet tests for the swept intermediate segment. This is summed up for all links.

Robot Type	Tolerance τ					
	$1mm$	$2mm$	$3mm$	$4mm$	$5mm$	$10mm$
4 joint planar	19.6%	28.1%	34.9%	40.4%	45.5%	66.8%
6 joint PUMA	27.8%	39.8%	49.2%	57.2%	64.3%	93.7%
10 joint LBR	17.2%	24.7%	30.6%	35.7%	40.3%	58.7%
16 joint snake	38.6%	54.8%	67.4%	78.1%	87.4%	125.0%
26 joint LBR/hand	29.1%	41.4%	50.8%	58.8%	65.9%	94.1%

Table 7.1: Comparison of swept volume and discretized steps for the collision detection of path segments. The comparison is based on the number of facets that have to be tested for intersection with the environment. The percentage shows the relation between the number of facets needed within swept volume collision detection and the number of facets that are needed for discretized steps. Several types of robots and different tolerance values τ were evaluated for one hundred random path segments.

In the comparison, 100 random path segments, i.e. pairs of random configurations, were evaluated, and the relative effort for swept volume collision detection was found as the relation of the number of facets required to be tested. The results are shown for tolerances between 1 and $5\,mm$ and for $\tau = 10\,mm$ in table 7.1. They can be interpreted as follows:

- The sublinear increase of effort of the swept volume collision detection becomes obvious: the smaller τ, the relatively better is the swept volume scheme.
- Tolerances below $5\,mm$ are required for realistic motion planning. The swept volume collision detection implies fewer tests and thus less effort in all cases. Even for a tolerance of $10\,mm$, it is superior in most cases.
- The efficiency is more or less independent from the number of joints or the robot architecture.

The rating functions are of the same complexity. The involved bisection is depth limited, thus only a constant factor increases the computational costs.

The rating functions benefit from the swept volume scheme analogous to collision detection.

7.5.3 Hierarchical Modelling

The model complexity can be high for realistic application sceneries, and to actually perform all facet intersection tests becomes very expensive. Therefore, hierarchic collision detection can be employed. Such a scheme is briefly described as follows: At the root node of a bounding volume tree a very simple geometry model is stored that covers the complete model. If the root node intersects with an object (either a geometric primitive or another bounding tree), the successing nodes are tested for intersection. These successors contain partial approximations of the full geometry model with higher precision. If they intersect, their respective successors that contain a further refinement of the model are tested. The leaves of such a hierarchical tree contain the geometric primitives. If there is no intersection on a high level, tests with these kinds of hierarchical structures can terminate very quickly without touching all facets. If there is a collision, the descent of the hierarchie "guides" the detection to the colliding primitive. Several suggestions for efficient hierarchies can be found in literature, using e.g. bounding boxes (see Gottschalk et al. (1996)) or spheres (see Pobil and Serna (1995), Quinlan (1994)) as hull bodies.

The prototype implementation uses a hierarchical model for the fixed objects of the environment G_O, made up of axes-aligned bounding boxes in a binary tree (see Baginski (1998)). The collision detection for transformed and swept models is speeded up by dynamically forming one bounding box, and only if this intersects, the full test is performed. This reduces the computational effort extremely and yields sufficient results for geometry models of low and medium complexity (up to a few hundred facets). For many realistic sceneries this is fully sufficient.

Further Discussion

Very complex geometry models often neither increase the available free space nor are they required with respect to a realistic lower bound of modelling uncertainty, i.e. these models are *too exact*. This typically happens if CAD-models are transformed from freeform patches into polyhedrons with very low tolerance. This problem is addressed in literature and it is easily possible to drastically decrease the modelling complexity without leaving a certain tolerance interval, see e.g. Cohen et al. (1997).

If more complex models (orders of thousands of facets) are necessary for planning, it is required to model the static *and* the moving objects hierarchically. The swept volume can also be structured hierarchically. This is not only possible for translational sweeps (as in Xavier (1997)), but also for arbitrary motions if the hierarchy is constructed based on the *edges* of the model. The collision test for an intermediate segment would use the "edge tree" twice, at both adjacent

configurations. Actual hull bodies that cover the W-linear sweeping can then be constructed "on the fly" within the hierarchical descent. But this is beyond the scope of this work and subject of future research.

Chapter 8

Experimental Results

Numerous motion planning examples of different topology and complexity are presented and the behaviour of the BB-method is shown for different tasks and parameters. The achieved results are compared to selected results from the literature.

8.1 Practical Aspects

Some practical aspects of the realized prototype implementation are discussed and the control parameters and the measured figures of the experiments are introduced.

Prototype Implementation

The prototype implementation realizes all components of the BB-method, based on the swept volume collision detection. To increase the efficiency, two main techniques are employed:

- **Collision detection instead of rating**
  Whenever an alternative path is rated, it is sufficient to execute a collision detection with a slightly larger model, and to start the rating only if this model is collision-free. This avoids unnecessary bisections.
- **Modified link order**
  The order of links within collision detection can be modified. If an alternative segment is rated, the collision detection is first of all performed for the current link. Other links need to be considered only if it is collision-free. This avoids an enormous number of unnecessary tests for "lower" links.

The prototype implementation is limited in its abilities to calculate expanded geometry models. It is only capable to create vertex displacement vectors for vertices with up to four intersecting facets, i.e. it is capable to insert an additional edge, but no additional facet (see section 7.3.2, p. 114). Whenever links include such vertices, no expansion is calculated for the complete model. No safety distances can be planned for such links with the prototype implementation. In regard to collision detection, it is assumed that the existing model is expanded by τ beyond the size of the physical link to allow a correct collision detection with this tolerance value.

Hardware Platform

The planning routines are coded in standard C and compiled with the GNU C-compiler Version 2.7. All results reported were achieved on UNIX(LINUX) workstations with an INTEL Pentium II, running with 266 MHz.

Parameter

The following parameters control the planning within the prototype implementation, and they are listed together with the results for all experiments reported:

- τ
  controls the tolerance used within the swept volume collision detection and the rating functions. It is set to a value of 5 mm and below to allow precise planning close to obstacles. In the tables the unit for τ is millimetre.
- γ
  controls the difference in size between the scaled models used for the rating of colliding paths, i.e. is the value used for γ_{scale} (see section 7.4, p. 118). It is denoted in millimetre. The size difference between expanded models is set to one millimeter for all reported experiments, i.e. $\gamma_{dist} = 1\,mm$.
- $\delta_{\min}$ and $\delta_{\max}$
  are also denoted in millimetre in the tables. They are used for collision-free path planning to limit the candidate displacement. The parameter $\delta_{\min}$ is used as lower displacement limit for safety distance planning as well.
- z_{random}
  Whenever random global planning is used, this parameter controls the maximum number of possible random subgoals that are created to find a solution. Note that unless otherwise stated, the results are all achieved with local planning only.
- $d_{\max}$
  is the desired safety distance for all links except the links connected to the environment. The prototype implementation does not support a vector of different safety distances, but one value only. In the tables its unit is millimetre.

The path length reduction is utilized with constant parameter values within all presented experiments. The "relative straightness" ρ that is accepted when the path is shortened is set to 5%. The value of λ_{joint} that is used to calculate the actually used segment length limit using equation (6.1) (see p. 101) is set to 10°. These values were chosen as they resulted in effective length reduction in all sceneries and for all kinds of robots. This choice is a compromise between quality and performance. Larger values for either parameter decrease the planning time as well as the quality. Smaller values may drastically increase the planning time without an appropriate increase in quality.

The task itself is given by the start and goal configurations $\mathbf{q}_{start}$ and $\mathbf{q}_{goal}$. The prototype implementation actually allows arbitrary initial paths, i.e. an ordered sequence of configurations, see section 8.2.7 below for an example.

Measurements

The following values are listed in the tables (in bold typeface, to differentiate parameters and results):

- t_{scale}
  The time required to plan a collision-free motion (or to realize a failure situation) with local planning. All times are given in seconds.
- $\overline{t_{succ}}$, $\overline{t_{fail}}$, t_{max}, *avg. sg.*
  If there are several tasks planned with the same parameters, the average time in case of successful planning is listed as $\overline{t_{succ}}$, the average time in case of failure is listed as $\overline{t_{fail}}$ and the maximum time is denoted as t_{max}. The average number of subgoals required for global planning is abbreviated with *avg. sg.*
- t_{dist}
  The time required to plan safety distances $d_{\max}$ along the path. For multiple tests the average planning time is listed as $\overline{t_{dist}}$.
- m_{dist}
  The "quality" of the achieved safety distances, calculated using (5.2). The highest possible value is task-dependent and usually below 100%. It is listed as percentage in the tables, and no average is given for multiple tasks, as this would not be a significant figure.
- t_{short}
  The time spent in shortening the path according to the parameters. For multiple tests the average time is listed as $\overline{t_{short}}$.
- m_{short}
  The relative length reduction achieved through the path length reduction, shown as percentage in the tables. No average is given for multiple tasks.

8.2 Examples

This section presents a variety of very different examples and the achieved results when using the BB-method. Examples with less than six joints were neglected, as they are not of practical interest.

8.2.1 6-DOF Manipulator with Difficult Load

The same manipulator was used as illustrating example in the previous chapters. For the tasks presented here, additional obstacles were placed in the workspace (see fig. 8.1). There is a second table with parts on it, a machine between the tables, and additional boxes behind the manipulator that limit its freedom to

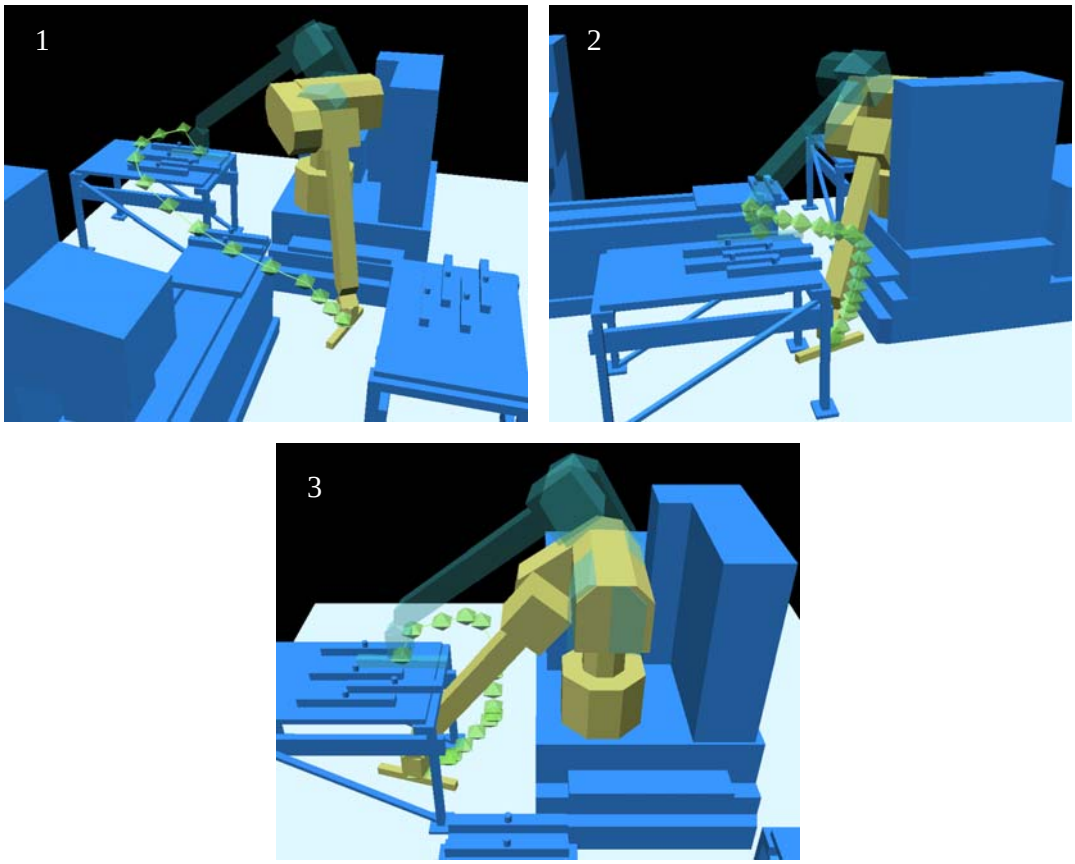

Figure 8.1: Different tasks for the 6-DOF manipulator with difficult load in a complex environment. For each task a solution planned with the BB-method is shown. 1: Task TABLE1. The load has to be lifted up from the floor onto the first table. 2: Task TABLE2. The load has to be moved up from below the second table. 3: Task TABLE3. The load has to be moved up from under the first table. This is a difficult variation of the task that was presented in the previous chapters.

move. Within this scenery, three selected tasks and a set of random tasks are planned. The selected tasks were intentionally chosen to be difficult tasks, the more obvious transfer tasks between the tables and the machine are planned even faster than the tasks shown.

Selected Task - TABLE1

The task TABLE1 is illustrated in fig. 8.1.1. The load has to be lifted up from the floor in between the machine and the second table and has to be placed on the first table, in the same goal position that was used in the illustrating example in the previous chapters.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	50	5	**0.35**	30	**0.18**	**96.7%**	**0.04**	**94.9%**
5	5	100	10	**0.25**	30	**0.19**	**97.2%**	**0.07**	**91.2%**
5	5	200	20	**0.17**	30	**0.23**	**98.3%**	**0.08**	**89.5%**
3	3	200	20	**0.24**	30	**0.18**	**97.6%**	**0.09**	**80.1%**
1	1	200	20	**0.35**	30	**0.62**	**97.1%**	**0.21**	**67.0%**

Table 8.1: Planning results for different parameters for task TABLE1. The best total planning time for a collision-free short path with safety distances less than half a second only.

Achieved results for the task TABLE1 using different parameters are listed in table 8.1. The tolerance τ, the scaling "granularity" γ and the step size parameters were varied. This influences the performance of all planning components. Planning works fine with all parameters. The average time to find a high quality solution, i.e. a path with safety distances and locally shortened, is about half a second. The planning time is larger for smaller displacement step size parameters $\delta_{\min}$ and $\delta_{\max}$, as more steps are needed to find a solution. Planning times are also increased for smaller values of τ and γ, as this increases the effort for the swept volume calculation and the bisection used in the rating function.

Selected Task - TABLE2

The task TABLE2 is shown in fig. 8.1.2. The load has to be moved up from below the second table. This task is especially difficult, as the robot is stretched more than in other tasks, and the box on its base limits its ability to move freely.

The tight situation yields longer planning times compared to the previous example, see table 8.2. Especially the planning of safety distances requires considerably more effort, as link 3 has to be kept clear from the box and the lower links and the load has to be keot in distance to the table. Note the distance quality m_{dist}, it indicates that the found solution cannot keep the distance due to the

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	50	5	**0.38**	30	**0.65**	**91.6%**	**0.34**	**82.3%**
5	5	100	10	**0.86**	30	**1.86**	**87.1%**	**0.90**	**80.2%**
5	5	200	20	**0.74**	30	**2.15**	**90.0%**	**1.20**	**60.8%**
3	3	200	20	**0.84**	30	**2.02**	**96.1%**	**0.46**	**79.2%**
1	1	200	20	**0.80**	30	**2.23**	**82.7%**	**1.13**	**73.2%**

Table 8.2: Planning results for different parameters for task TABLE2. The task is difficult, as the robot is very limited in its movability. Therefore, planning is quickest with small displacement steps.

obstacle configuration. In a practical implementation it might not be required to plan safety distances of such magnitude between the lower links of the robot and the well known fixed obstacles near by, but only for the load.

The best performance is achieved with small displacement step sizes $\delta_{\min}$ and $\delta_{\max}$, another indication of a very tight situation. If the parameters are larger, the segments have to be subdivided several times to yield the required small displacements, this delays the solution in these cases. But nonetheless, local planning is reliable and a solution of high quality is planned in about $4\,sec$ at maximum.

Selected Task - TABLE3

The task TABLE3 is shown in fig. 8.1.3. It is quite similar to the illustration task presented in the previous chapters, but more difficult. The load is deeply under the table. At the start configuration link four is almost in touch with the edge of the table.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	50	5	**1.10**	30	**1.21**	**84.6%**	**0.68**	**68.1%**
5	5	100	10	**0.39**	30	**0.91**	**89.6%**	**0.33**	**82.3%**
5	5	200	20	**0.34**	30	**0.32**	**84.9%**	**0.53**	**67.8%**
3	3	200	20	**0.43**	30	**1.33**	**93.1%**	**0.59**	**72.1%**
1	1	200	20	**0.51**	30	**2.09**	**95.3%**	**0.99**	**51.0%**

Table 8.3: Planning results for different parameters for task TABLE3. It is planned in less than two seconds for suitable parameters. This is sufficient for real-time applications.

As there is no obstruction for the robot in its lower links, this task is planned faster than the previous one, see table 8.2. The load is transported “carefully”,

and the planning time is about two seconds using suitable parameters. In flexible manufacturing applications, this is sufficient for real-time operation of manipulators. Note that the same parameters are used for all tasks and show equally good results.

Set of Random Tasks

To validate the applicability of the BB-method beyond the scope of a few manually selected tasks, as set of 500 random tasks is created. To choose arbitrary random configurations would not yield reasonable tasks. Therefore, random "pick-and-place"-tasks are used. Configurations where the load is about 5 to 10 mm away from the obstacles are generated. Pairs of such configurations that cannot be connected by a collision-free straight line in c-space are selected as random pick-and-place tasks. Note that these are typically rather difficult tasks. They start and end close to contact, and no straight connection is possible (this filters out more than 50% of the obstacle-close configuration pairs).

τ	γ	$\delta_{\max}$	$\delta_{\min}$	SUCCESS	$\overline{t_{succ}}$	$\overline{t_{fail}}$	$d_{\max}$	$\overline{t_{dist}}$	$\overline{t_{short}}$
5	5	50	5	**90.6%**	**0.21**	**0.59**	25	**0.74**	**0.20**
5	5	100	10	**92.2%**	**0.15**	**0.46**	25	**0.71**	**0.20**
5	5	200	20	**93.6%**	**0.15**	**0.56**	25	**0.89**	**0.26**
3	3	200	20	**95.2%**	**0.17**	**0.91**	25	**1.09**	**0.28**
1	1	200	20	**96.4%**	**0.23**	**1.20**	25	**1.40**	**0.36**

Table 8.4: Planning results for different parameters for 500 random tasks, local planning only. SUCCESS indicates the percentage of successfully planned tasks.

The results achieved with local planning for the random tasks using the same sets of parameters as used for the selected tasks are shown in table 8.4. Local planning is not sufficient to solve all tasks, but the success rate is above 90%. Larger steps increase the success rate, as they reduce the risk to get stuck locally. More precise collision detection and rating (smaller τ and γ) also increases the success rate as it allows to pass through more difficult narrow passages.

Collision free path planning times are short in case of success as well as in case of failure. More precise collision detection and rating increases the time of failure detection as the path can be adapted more precisely to a "local maximum". The average effort for safety distance planning and path length reduction is also increased for more precise collision detection and rating.

The tasks that cannot be solved locally are subject of global planning based on random subgoals. This allows to solve 100% of the tasks, independent of the parameters (see table 8.5). The average number of subgoals is less than three. The average time to solve all 500 tasks is in the range of 0.24 sec to 0.39 sec for the different parameters.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	z_{random}	SUCCESS	*avg. sg.*	$\overline{t_{succ}}$	t_{max}
5	5	50	5	100	**100%**	**2.3**	**2.09**	**7.3**
5	5	100	10	100	**100%**	**1.8**	**1.27**	**3.5**
5	5	200	20	100	**100%**	**2.1**	**1.68**	**13.3**
3	3	200	20	100	**100%**	**1.8**	**2.14**	**20.54**
1	1	200	20	100	**100%**	**2.9**	**2.64**	**17.41**

Table 8.5: Planning results for different parameters for global planning of the tasks out of the random set that failed to plan locally. On average, less than 3 random subgoals are needed to find a solution.

These results, achieved for a realistically modeled 6-DOF robot in an environment full of obstacles, carrying a lengthy load and considering only "difficult" tasks, show the practical applicability of the BB-method.

8.2.2 7-DOF Redundant Manipulator

The 7-DOF manipulator shown in fig. 8.2 is developed at the robotics institute of the German Aerospace Center (DLR), see Ackermann and Hirzinger (1996). It is intended for the use in orbital space laboratories. The kinematic structure is very different to the 6-DOF manipulator, all links are roughly of the same size and the joint axes are perpendicular to the respective preceeding joint axis. The convex hull of the original geometry model is used in the experiments. Due to the structure of these geometry models – more than four facets intersect in the vertices – no safety distances can be planned with the prototype implementation.

Three tasks were selected for the tests, see fig. 8.2 for an illustration. The robot operates in a room with boxes and a shelf and has to move between different storage and operation locations.

Task	τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	t_{short}	m_{short}
BOX-SHELF 1	5	5	200	20	**0.14**	**0.05**	**69.0%**
	5	5	300	30	**0.10**	**0.06**	**71.4%**
BOX-SHELF 2	5	5	200	20	**0.47**	**0.17**	**71.7%**
	5	5	300	30	**0.63**	**0.12**	**53.7%**
BOX-BOX	5	5	200	20	**0.56**	**0.33**	**44.0%**
	5	5	300	30	**0.83**	**0.48**	**53.7%**

Table 8.6: Planning results for different parameters/different tasks for the 7-DOF manipulator. Even the most difficult task is planned and shortened in under 1.5 sec.

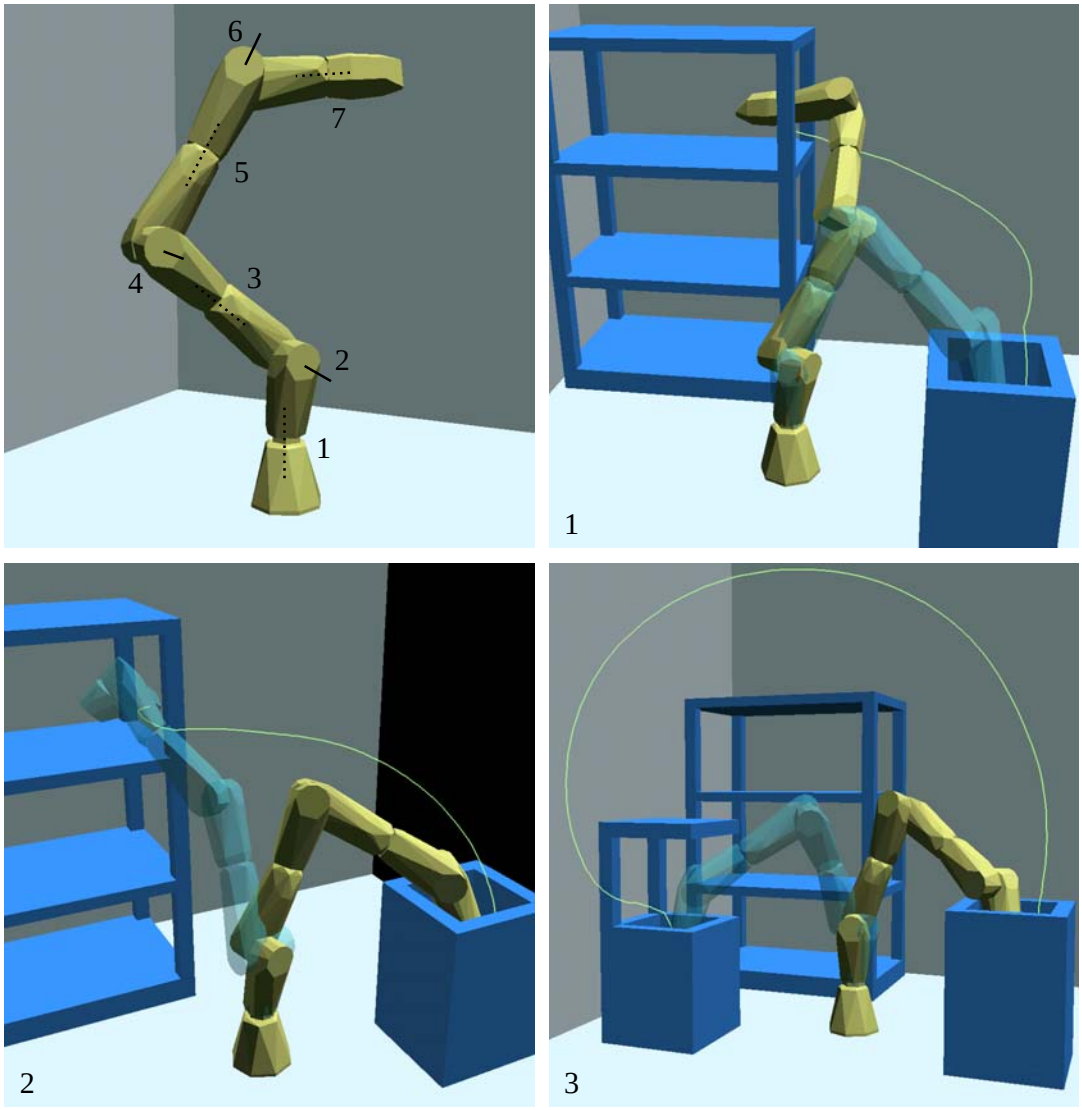

Figure 8.2: 7-DOF redundant manipulator. The top left picture shows the 7 perpendicular joint axes. The joints with axes *along* the kinematic chain (1,3,5,7) have a motion range of 500°, the other joints have a motion range of 300°. 1: Task BOX-SHELF 1 with a planned solution. The robot has to move out of a shelf into a tight box. 2: Task BOX-SHELF 2. The robot has to move out of the box into the shelf in a different orientation that requires to move around one of the supports. 3: Task BOX-BOX. The robot has to move out of the tight box into a second equally tight box which is also covered.

The table 8.6 shows planning results achieved for the different tasks using different parameters. The robot is about 1.8 m long when fully stretched, therefore larger stepsize parameters were chosen, compared to the 6-DOF manipulator. All tasks are planned and shortened in less then 1.4 sec. With the BB-method, real-time planning is possible for redundant manipulators.

8.2.3 7-DOF Redundant Manipulator on 3-DOF Gantry (10-DOF)

To be able to operate more flexibly in a space laboratory the 7-DOF robot is mounted on a 3-DOF cartesian gantry that allows to move the base of the manipulator at an arbitrary position within the workspace of the gantry. This is illustrated in fig. 8.3. The resulting manipulator system has 10 degrees of freedom. The geometry data of the gantry was supplied by the DLR.

A prototype workspace setup was created to evaluate motion planning for the 10-DOF system. A group of seven boxes, one of them very deep, are fixed to the rear wall and are fully accessible by the robot. Three tasks are evaluated, they are shown in fig. 8.3. The transfer between the boxes can either be executed with the same orientation of the arm (task BOX1), or with a changing orientation of the arm (task BOX2). The latter case requires the manipulator to stretch itself along the motion. An even more difficult situation is given when the orientation has to be changed and one of the boxes is the deep one (task BOX3).

Task	τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	t_{short}	m_{short}
BOX1	5	5	100	10	**0.88**	**0.29**	**91.5%**
	5	5	200	20	**0.38**	**0.23**	**97.7%**
	5	5	300	30	**0.27**	**0.32**	**94.6%**
BOX2	5	5	100	10	**3.73**	**0.54**	**87.8%**
	5	5	200	20	**1.13**	**1.00**	**84.6%**
	5	5	300	30	**1.12**	**0.66**	**81.2%**
BOX3	5	5	100	10	**3.24**	**0.99**	**92.5%**
	5	5	200	20	**2.62**	**1.19**	**77.0%**
	5	5	300	30	**1.63**	**1.31**	**75.6%**

Table 8.7: Planning results for different parameters/different tasks for the 10-DOF manipulator. Any transfer task between the boxes is solved in about 4 sec or less.

The achieved results for the three tasks using different parameters are shown in table 8.7. Even the most difficult task BOX3 can be solved and shortened in about 4 sec. This also is the upper limit when planning motions between the deep box and all other boxes (including change of orientation) with $\delta_{\min} = 300\,mm$ and $\delta_{\max} = 30\,mm$. The results show that effective planning for complex kinematic structures is possible with the BB-method.

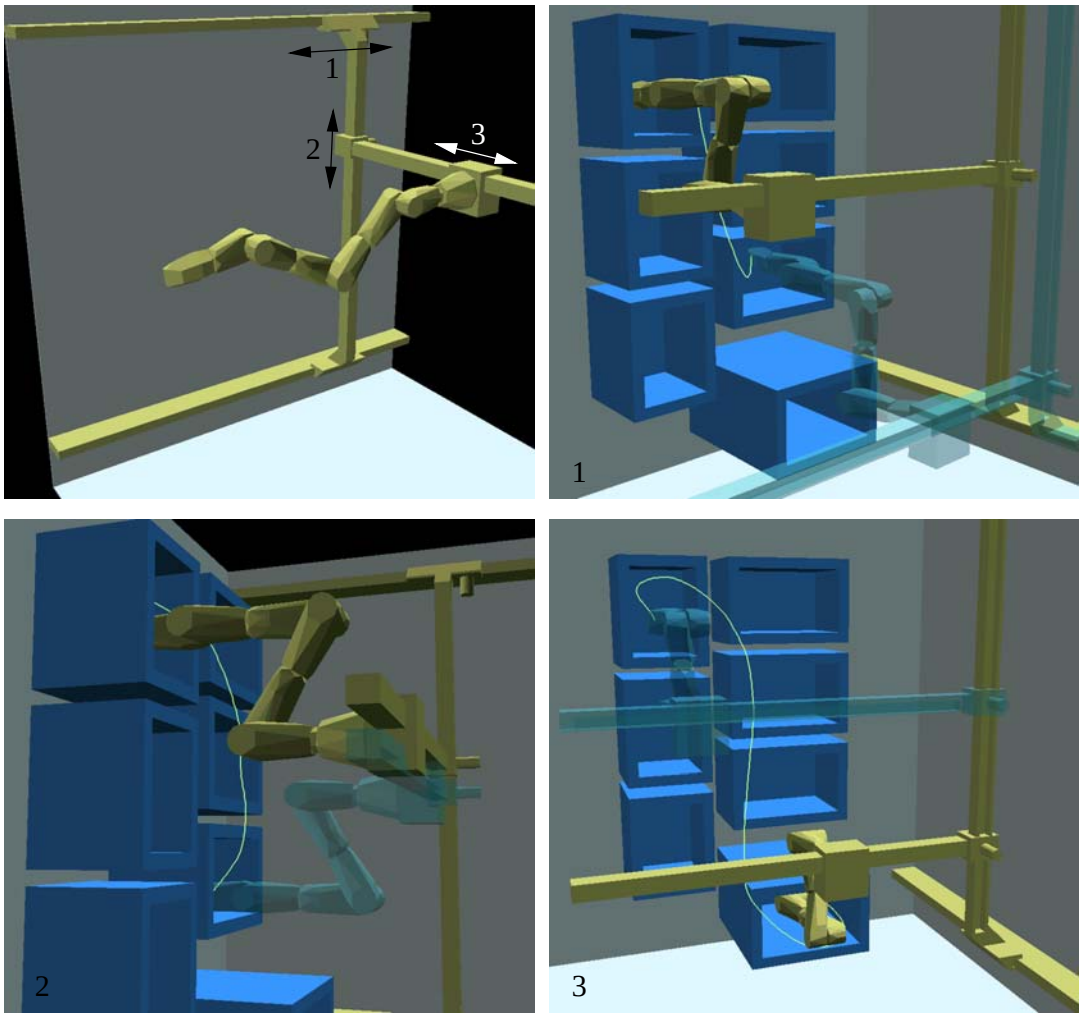

Figure 8.3: 7-DOF redundant manipulator on 3-DOF gantry. The top left picture shows the kinematic structure of the gantry where the 7-DOF manipulator is fixed. It allows to translate along all three axes in the space, i.e. the base of the manipulator can be positioned arbitrary in a cubic workspace. 1: Task BOX1. The robot has to move between two boxes without changing the arm's orientation. The tip trace of a solution is shown. 2: Task BOX2: The same two boxes are start and goal respectively, but in this task the arm has to change its orientation. The tip trace is nonetheless quite similar. 3: Task BOX3: Again, the orientation has to be changed, and the start position is within the very deep box.

8.2.4 7-DOF Redundant Manipulator on 3-DOF Gantry with 4-Fingered Hand (26-DOF)

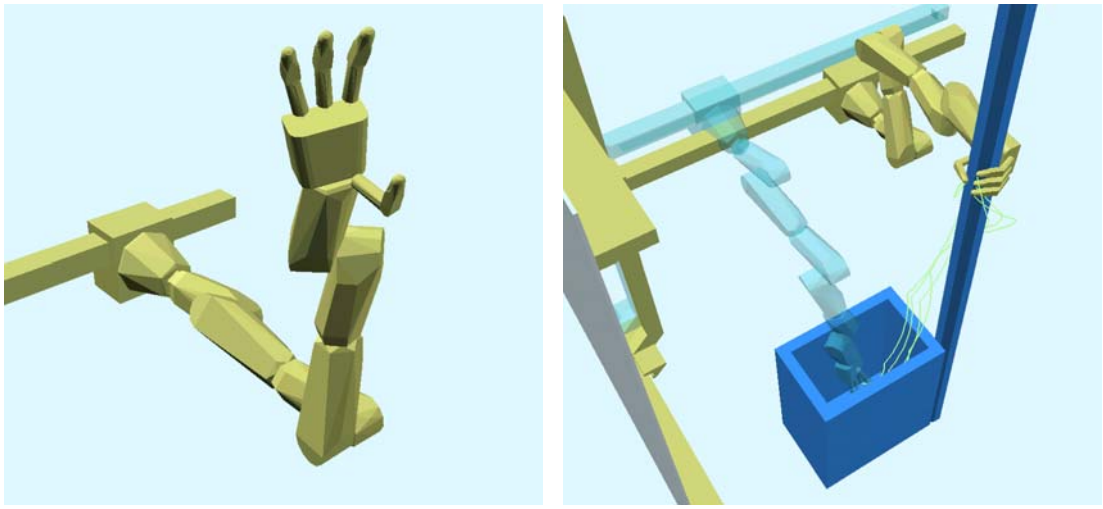

Figure 8.4: 7-DOF redundant manipulator on 3-DOF gantry with 16-DOF 4-fingered hand. Right: The 4-fingered hand is mounted as tool on the 7-DOF manipulator that is attached to the gantry. Each finger has four degrees of freedom: moving left/right at the base, moving forward/backward at the base, and moving forward/backward with the two inner joints of the finger (equal to the kinematic structure of a human finger). Left: The example task. The hand has to release a grip to a column and move into a tight box. The solution is indicated by the motion traces of the finger tips.

To allow arbitrary manipulation, a very flexible tool is developed at the German Aerospace Center. It is a 4-fingered hand, and each finger has four degrees of freedom. The fingers have the same kinematic structure as human fingers, two degrees of freedom at the base and two inner joints along the finger. The hand, mounted as a tool on the 7-DOF manipulator on the gantry, is shown in fig. 8.4. Again, the original geometry data was available for the experiments. The overall number of joints in the kinematic structure is 26.

A sample task was created and planned with the BB-method. The hand has to move out of a tight grip around a column down into a deep box. The planning times listed in table 8.8 show that the effort is about the same as for the 10-DOF scenery above. This is a very valuable property of the BB-method, and probably unrivaled by other motion planning approaches. It allows to plan out of or into grasp positions with the motion planner itself, as no additional planner is needed for the approach/departure. And it allows a coordinated motion of the tool and the arm, intead of requiring a fixed position for the gross motion.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	t_{short}	m_{short}
5	5	100	10	**7.18**	**2.85**	**76.0%**
5	5	200	20	**2.27**	**1.75**	**73.6%**
5	5	300	30	**2.89**	**1.96**	**57.3%**

Table 8.8: Planning results for different parameters for the 10-DOF manipulator with 4-fingered hand for the example task. Although the number of joints and the complexity of geometry modelling has increased very much compared to the previous examples, the planning performance is almost equal.

8.2.5 16-DOF Snake Manipulator

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
1	1	200	20	**1.16**	30	**2.07**	**>99.9%**	**0.46**	**83.5%**
2	2	200	20	**0.84**	30	**0.81**	**>99.9%**	**0.26**	**83.9%**
3	3	200	20	**1.09**	30	**1.73**	**>99.9%**	**0.40**	**85.1%**
4	4	200	20	**1.20**	30	**0.58**	**99.9%**	**0.39**	**74.6%**
5	5	200	20	**1.24**	30	**1.02**	**99.8%**	**0.51**	**71.1%**
2	2	400	40	**6.29**	30	**3.51**	**99.8%**	**0.85**	**49.5%**
2	2	300	30	**3.97**	30	**2.53**	**99.9%**	**0.85**	**49.5%**
2	2	100	10	**2.20**	30	**3.71**	**99.9%**	**0.52**	**78.0%**
2	2	50	5	**12.04**	30	**3.38**	**99.9%**	**0.44**	**72.2%**
2	2	25	2	**8.66**	30	**2.49**	**>99.9%**	**0.30**	**91.0%**

Table 8.9: Different parameters and results for planning the 16-DOF manipulator/gate task that was used as illustrating example in the previous chapters. $> 99.9\%$ indicates a percentage in the interval $[99.95, 100.0[$.

The example task of the previous chapters (see fig. 3.5, p. 45) was tested for a wide variety of parameter values, the results are listed in table 8.9. This shows the reliability and robustness of the BB-method. The parameters allow to increase the planning performance, but planning is successful independent of the parameter values.

The safety distances and the path length reduction works well with all the paths found for the different step size limit parameters. The latter influence the planning effort of collision-free path planning: Planning takes more time for larger and for smaller step size limits, and is most effective for intermediate values. This is an obvious behaviour. Small step size limits require more modification steps, and thus it takes longer to reach a solution. Larger step size limits allow either displacements that are "too far" and have to be corrected by additional

modification steps. Or they do not allow any modification (consider the tight gate), and the path has to be subdivided several times to yield displacement step sizes that allow a rating improvement.

8.2.6 31-DOF Snake Manipulator

To test the capabilities of the BB-method for robots with even more joints a 31-DOF snake manipulator was created. It is based on the same principle as the 16-DOF manipulator, with a sequence of ten 3-joint connections and one base joint. The intermediate links are shorter than the one used for the 16-DOF manipulator, its overall length when fully stretched is $2.7\,m$ (compared to $2\,m$ of the 16-DOF robot). The robot, the sample sceneries and planned solutions are shown in fig. 8.5.

Scenery ONE GATE

The gate is of same size and relative position to the robots' base as in the 16-DOF example. In the case of the 16-DOF manipulator, links 11 through 16 were in or behind the gate. Using the 31-DOF robot, links 15 through 31 have to be retracted successfully. Considering the tight obstacle, the large and lengthy robot, and the high number of joints, this task is extremely difficult.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	100	10	**16.87**	20	**12.36**	**>99.9%**	**2.71**	**79.0%**
5	5	200	20	**9.57**	20	**6.54**	**99.9%**	**0.85**	**89.5%**
5	5	300	30	**13.10**	20	**8.89**	**99.2%**	**2.12**	**74.5%**

Table 8.10: Planning results for different parameters for the 31-DOF manipulator in scenery ONE GATE.

The BB-method is able to solve this task, achieved results are listed in table 8.10. With suitable parameters, a solution that keeps full safety distances and is shortened can be found in $17\,sec$.

Scenery THREE GATES

This task is even more difficult than the previous one. The manipulator is caged by three gates, see fig. 8.5.2. The retraction motion has to be aligned with these gates, thus it can be assumed that the passage in c-space is much tighter than for one gate.

Planning results in this scenery are listed in table 8.11. Local planning fails for several sets of parameters. This is the failure situation identified as "loss of degrees of freedom" (see section 4.4.2, p. 71). The foremost links cannot be displaced without yielding collisions for preceeding links. The candidate creation

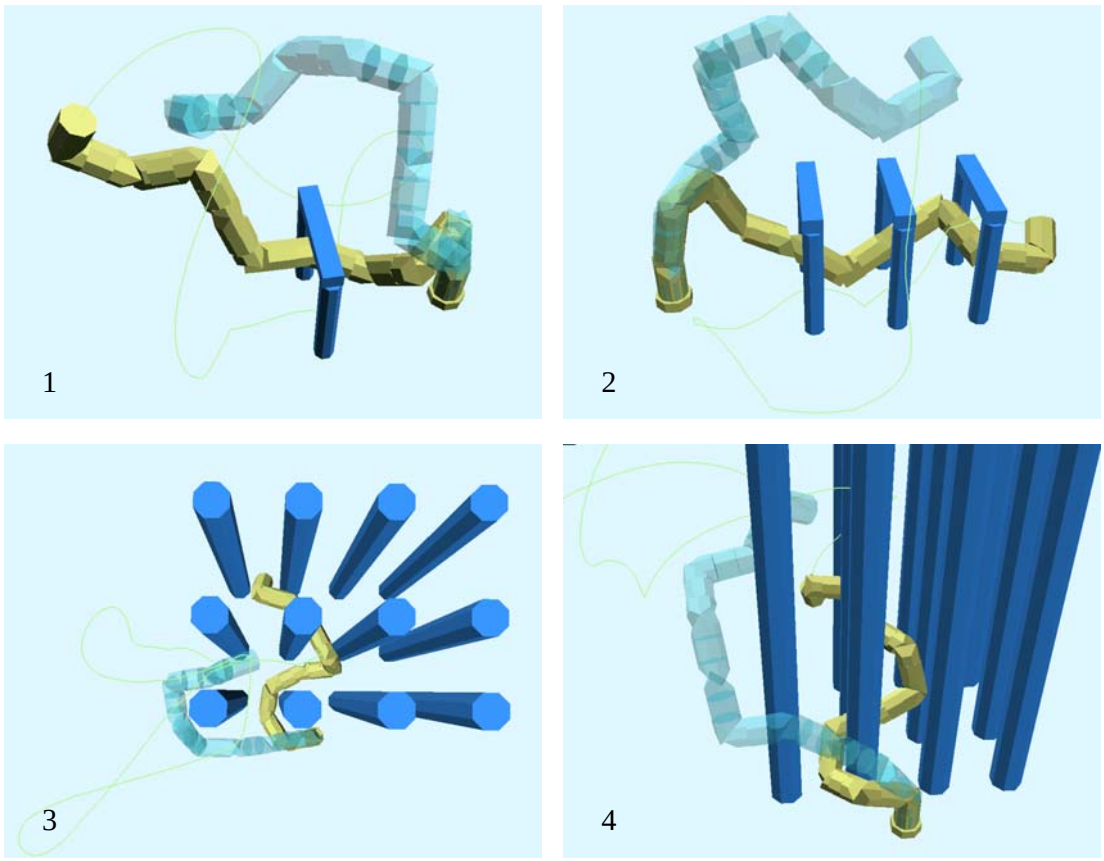

Figure 8.5: The 31-DOF Snake Manipulator is an extended version of the 16-DOF manipulator presented in the previous chapters. It consists of 10 equally modeled three joint connections and an additional joint at the base. Planning examples and solutions for three different sceneries are shown. The solutions may appear strangely shaped, but they are short in the 31-dimensional c-space. 1: Scenery ONE GATE. This scenery equals the 16-DOF task, but the robot is much longer and thus it has to retract much further to get out of the gate. 2: Scenery THREE GATES. The entire robot is caged by three gates that it has to escape, its abilities to retract are further limited. 3,4: Scenery COLUMNS. The robot operates between tightly placed columns.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
2	2	100	10	**95.99**	20	**111.70**	**99.9%**	**8.97**	**81.4%**
5	5	100	10	**66.83**	*local planning failed*				
5	5	200	20	**23.84**	20	**19.22**	**99.7%**	**4.34**	**65.9%**
5	5	300	30	**8.64**	*local planning failed*				

Table 8.11: Planning results for different parameters for the 31-DOF manipulator in scenery THREE GATES. For several parameters planning fails due to "loss of degrees of freedom".

for collision free path planning does not consider possible collisions in lower links. These collisions with the first or second gate disqualify a lot of – and possibly all – alternative paths. Note that failure is detected fast, an important property of the local planning. Using suitable parameters, a solution of very high quality is found in about 47 *sec*.

Scenery COLUMNS

In this scenery, the robot has to wriggle itself out of a dense arrangement of columns, see fig. 8.5.3 and 8.5.4. The distance between the columns is just twice the diameter of the robot.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	100	10	**31.23**	20	**72.46**	**93.7%**	**32.7**	**54.3%**
5	5	200	20	**33.94**	20	**54.83**	**98.7%**	**6.9**	**72.2%**
5	5	300	30	**51.66**	20	**66.31**	**98.7%**	**15.7**	**59.6%**

Table 8.12: Planning results for different parameters for the 31-DOF manipulator in scenery COLUMNS. This extremely difficult task, especially with respect to safety distances, is planned in high quality in less than two minutes.

This task is the most difficult one considered in the series of examples. Especially the safety distance planning between the columns is extremely complex for 30 links. But nonetheless a solution can be found in less than two minutes and the safety distances are almost fully achieved (start and goal are too close to the obstacles to allow full safety distances). The problem of "loss of degree of freedom" does not occur in this example. Between the columns, there is always the possibility of vertical movement, as opposed to the scenery with the gates.

The results achieved for the 31-DOF robot show that the BB-method also is successfully applicable to hyperredundant manipulators in extremly cluttered environments.

8.2.7 Planning for a Mobile Manipulator

Mobile robots can be modeled with two translational and one rotational joint prior to the first physically existing "link" (see section 3.2.2, p. 33). But in general, paths for mobile robots cannot be planned with the BB-method, as they do not fulfil condition (4.1), i.e. the origin of the scaled link is not located within the previous link as the previous link of link three is non-existing (see section 3.2.2, p. 33). To illustrate the problem: all paths through a wall would be rated equally zero, no alternative path can be found locally that improves this rating.

But if there is no wall in between start and goal, i.e. if the local origin of the mobile platform itself can move without collision, alternative paths can be compared effectively and motion planning is possible. This is validated by planning motions for the mobile robot ROMAN in two different sceneries.

ROMAN was developed in the joint research project SFB 331 (1986-1997), for details see Daxwanger et al. (1996). It is a 3-DOF holonomic mobile platform equipped with a 7-DOF manipulator, see fig. 8.6. The original geometry model of the robot was available for the experiments.

Scenery ROMAN1

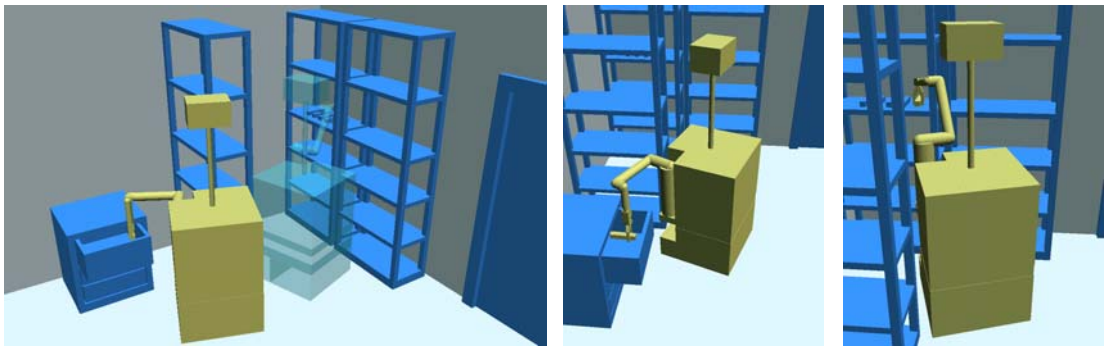

Figure 8.6: Scenery ROMAN1. Robot ROMAN is a mobile platform with three degrees of freedom, equipped with a 7-DOF manipulator. In this scenery, it has to pick up a load out of an open drawer and place it on a shelf. Left: Overview of the complete task. Center: Start configuration with the load in the drawer. Note that it is not sufficient to lift it straight up, as the drawer is not opened wide enough. Right: Goal Configuration with load in the rear shelf. The position is hard to reach as there is another shelf that limits the movability of the mobile platform.

In this scenery a load has to be transported out of a drawer onto a shelf. The C-linear motion yields collision for the arm and the body. But the local origin of the body is located in its center, therefore no zero-scaling occurs.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	100	10	**8.34**	20	**11.10**	**98.7%**	**1.99**	**64.6%**
5	5	150	10	**6.18**	20	**14.74**	**98.9%**	**1.66**	**82.7%**
5	5	200	10	**4.14**	20	**6.47**	**99.5%**	**1.56**	**77.6%**
5	5	250	10	**2.69**	20	**7.16**	**98.8%**	**0.94**	**78.3%**
5	5	300	10	**6.29**	20	**8.09**	**99.5%**	**1.20**	**72.5%**

Table 8.13: Results for scenery ROMAN1. The difficult task consisting of manipulation and transfer motion can be planned in about 11 sec with default parameters.

Planning results for a variety of parameter values are listed in table 8.13. As observed for other examples, planning is most efficient for a certain upper displacement threshold $\delta_{\max}$ that does not require too many intersections or too many steps. Planning can be completed in about 11 sec and yields a high quality motion, see fig. 8.8. Note that the resulting motion is a combined transfer motion of the mobile platform and simultaneously a manipulation executed by the attached manipulator. The control spaces of two more or less separate systems are considered as a combined planning space by the BB-method without differentiating the components.

Scenery ROMAN2

Scenery ROMAN2 consists of two separate rooms, connected by a door. The workspace and a cross-section of the configuration space are visualized in fig. 8.7. The task is basically the same as in scenery ROMAN1. The robot has to transport a load out of a drawer onto a shelf. But it has to move around an obstructing shelf and through the door to get to the goal.

The initial C-straight path is not sufficient to start planning with the BB-method as it passes a region of zero-scaling. The prototype implementation allows to use arbitrary sequences of configurations as initial paths. For this example, two configurations were manually selected in a way that the center point of the body is able to move without collision. The configuratuions are shown in the Workspace and in a c-space cross-section in fig. 8.7. These intersection configurations along the initial path are treated like any other intersection configuration created in the ongoing planning process, i.e. they are modified if necessary and possible.

Using this sequence of four configurations, the BB-method successfully solves the task in scenery ROMAN2, results are listed in table 8.14. The effort is comparable to the effort required in scenery ROMAN1. A solution is shown in fig. 8.8.

This example demonstrates two important aspects:

- The BB-method is suited for interactive planning as it can be used with arbitrary initial paths.

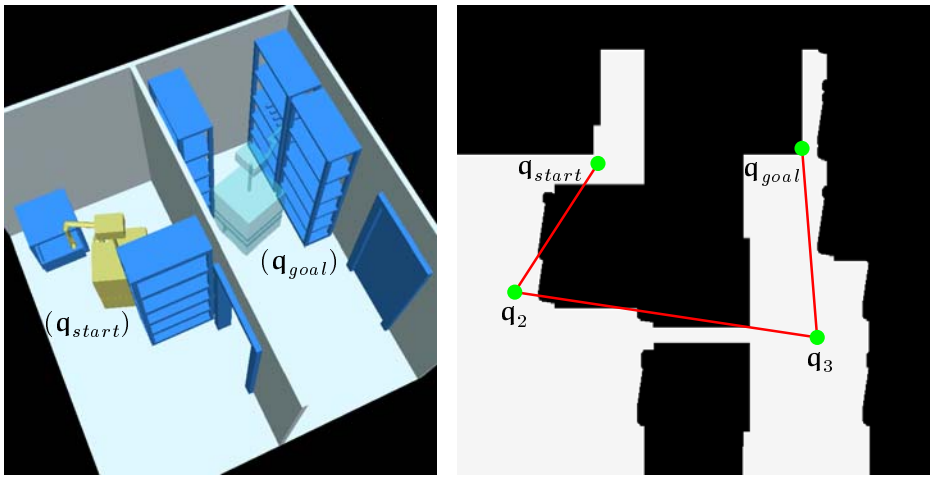

Figure 8.7: Scenery ROMAN2 is an extension of scenery ROMAN1. The direct connection is blocked by a wall, and a shelf close to the start requires a detour to reach the door that allows to acces the goal. Left: Overview of the scenery. and the approximate intersection configurations chosen for the initial path. Right: A cross-section of the c-space. The two translational joints 1 and 2 of the mobile platform are varied, all other joints are constant. The obstacles appear "blown up" by the thickness of the robot itself, but the ground-plan of the rooms can be recognized. The initial path used for planning is shown.

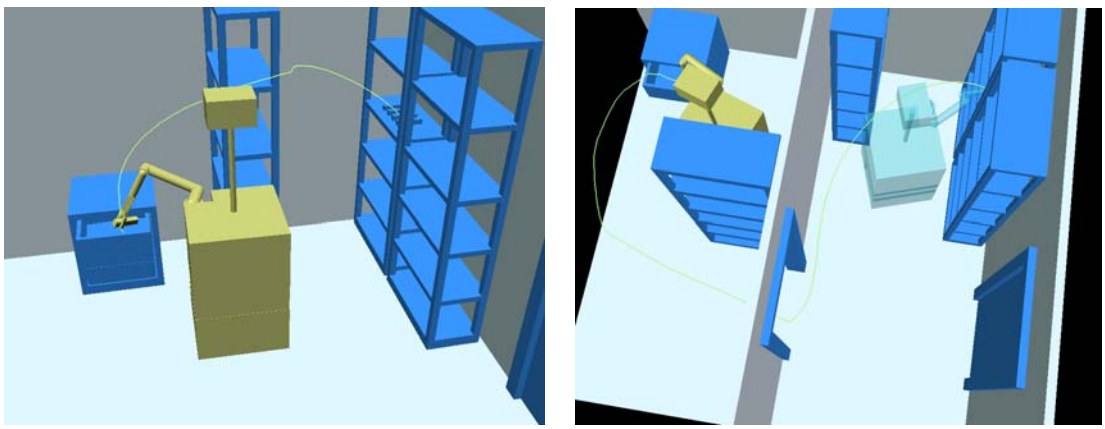

Figure 8.8: Solutions for robot ROMAN. Left: A typical solution found in scenery ROMAN1. The load is lifted up to pass the shelf. Note how the load is turned during the departure to assure the safety distance. Right: A solution found in scenery ROMAN2, based on the initial path consisting of four configurations.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	$d_{\max}$	t_{dist}	m_{dist}	t_{short}	m_{short}
5	5	100	10	**8.56**	20	**13.42**	**97.2%**	**0.97**	**84.1%**
5	5	150	10	**3.98**	20	**8.74**	**98.1%**	**1.28**	**94.8%**
5	5	200	10	**2.53**	20	**3.73**	**95.8%**	**1.11**	**83.9%**
5	5	250	10	**4.08**	20	**3.90**	**99.0%**	**0.93**	**90.1%**
5	5	300	10	**2.34**	20	**2.82**	**98.5%**	**1.18**	**81.9%**

Table 8.14: Results for scenery ROMAN2. Using an initial path of four configurations, planning does not take more time than in scenery ROMAN1.

- The BB-method can be extended to other areas of robotic applications. Combined with a coarse planner for the navigation of a mobile platform it is well suited to be used for mobile manipulation planning.

8.3 Comparison

8.3.1 Problems of Comparison

It is extremely difficult to compare results published in literature with own results. In general, each author will select examples that fit best the capabilities of the presented algorithm. There is no set of accepted "benchmark-problems" in the area of motion planning (there is an ongoing activity from the IEEE Robotics and Automation Society and V. Lumelsky, but only some previous results and guidelines were published in Hwang (1996). These were picked up in Wurll et al. (1997), see section 8.3.3 below).

Exact planning results for tasks of the complexity as presented above, based on realistic three-dimensional geometry models, are not yet published in the motion planning literature. Therefore, only a few recent results could be found to allow comparison. A more precise comparison is possible with the ZZ-method, as the prototype implementation of the BB-method was realized using the existing environment of the ZZ-method.

8.3.2 Comparison with *RPP* – Randomized Path Planner

The randomized path planner RPP is based on a combination of potential field planning and random minima escape, see section 2.7, p. 23, for details. The "original" implementation of this planner which is available in the public domain only allows planning for robots consisting of line segments. This is not suited for comparison. A more recent implementation of this planner is reported in Challou et al. (1995) and Gini (1996), using fully three-dimensional geometry models. The implementation parallelizes RPP, using special hardware. Each processing node attempts to solve the task, the first solution found is taken. As RPP involves

random motion, this approach is promising and increases the performance of RPP. The disadvantage is the required high number of processing nodes (e.g. a workstation cluster) or special hardware to achieve a good performance.

The reported implementation postprocesses the planned path to shorten it locally, but safety distances are not considered. Two sceneries were created acording to the pictures shown in Gini (1996), and the achieved planning times for collision-free path planning and path length reduction are compared. RPP requires preprocessing to built up potentials prior to planning. No details are reported concerning the computational costs of preprocessing and it is not considered in the comparison. But it should be noted that there is no such preprocessing required to plan with the BB-method.

Scenery HOLE IN THE WALL

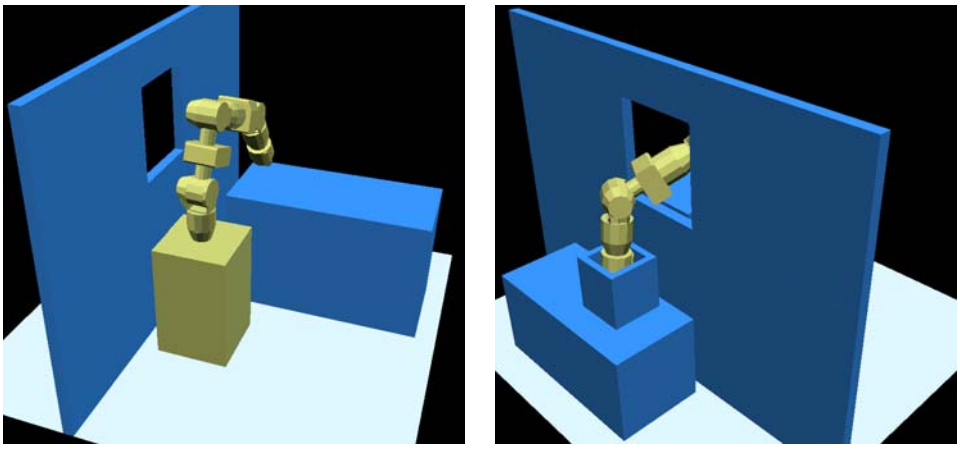

Figure 8.9: Scenery HOLE IN THE WALL. The 7-DOF robot has to move from a table through an opening and down into a tight box. Left: Start configuration at the table. Center: Goal configuration in the box on the other side of the wall. Right: Complete task from above.

The task in scenery HOLE IN THE WALL is shown in fig. 8.9. A 7-DOF manipulator has to move from a table through an opening in a wall into a box.

Results for different parameters are shown in table 8.15. A solution is planned and shortened in about 2 *sec*. The results published for the parallel implementation of RPP show a planning time of 2.47 *sec* for this task – if the planner runs on 512 processors of a CM-5 multicomputer. If running on a single processor, planning requires about 60 *sec*. This shows that the BB-method allows real-time planning on standard hardware, a result not achievable with RPP.

Another major difference is the "planning resolution". As opposed to RPP, which uses a discretization grid for the workspace (to calculate the control point

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	t_{short}	m_{short}
5	5	100	10	**1.23**	**0.31**	**84.6%**
5	5	150	15	**0.95**	**0.26**	**62.9%**
5	5	200	20	**1.50**	**0.79**	**73.0%**
5	5	250	25	**1.56**	**0.95**	**52.9%**
5	5	300	30	**1.26**	**0.22**	**53.2%**

Table 8.15: Results for scenery HOLE IN THE WALL. A high quality solution is found in less than two seconds. This is suitable for real-time applications.

potentials) and of the c-space (were the potentials are mapped to to "move" the robot), the BB-method works on continuous joint variables and uses a very small tolerance value in the workspace. This can be expressed in figures: the RPP-implementation supports only 128 positions for each joint, and the workspace is discretized in voxels with an edge length of $2.1\,cm$. This limits the planning abilities of RPP close to obstacles and in narrow passages. The BB-method avoids any discretization and allows adaption of paths as close to the obstacles as specified by the tolerance τ, a value set to $5\,mm$ or below in all experiments.

Scenery TWO SLOTS

An artificial task for a 6-DOF planar robot is shown in fig. 8.10. The robot has to pass two narrow passages and an obstacle very close to its base. The robot has to retract itself very far without yielding selfcollisions. On first sight this task seems almost unsolvable. The robot is modeled to be $2\,m$ in length when fully extended.

τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}	t_{short}	m_{short}
5	5	100	10	**0.57**	**0.09**	**85.8%**
5	5	150	15	**0.43**	**0.17**	**85.1%**
5	5	200	20	**0.40**	**0.11**	**71.3%**
5	5	300	30	**0.53**	**0.23**	**71.9%**
5	5	400	40	**0.53**	**0.24**	**59.1%**

Table 8.16: Results for scenery TWO SLOTS. A shortened solution is found in less than $1\,sec$ with arbitrary parameter values.

As the geometry models are very simple, thus collision detection and rating is very fast, the BB-method solves this task almost instantly in less than one second, independent of the parameter values.

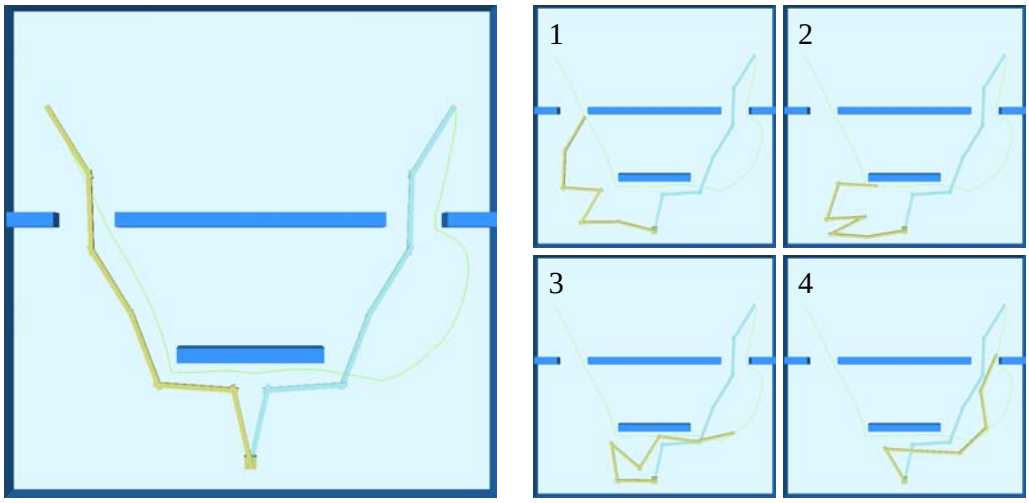

Figure 8.10: Scenery TWO SLOTS. This artificial simplified scenery contains a 6-joint planar robot. It has to retract through the left slot, pass an obstacle very close to its base and extend through the right slot. Left: Task and tip trace of a solution. Right: Intermediate stages of solution path. The robot contracts itself in the lower left part of its workspace to pass the obstacle and avoid selfcollision.

The parallelized implementation of RPP needs 180 *sec* to solve this task when running on 1024 processors. Planning on a single processor takes about 10 h on average. This example, although rather artificial, shows the ability of the BB-method to plan efficiently in very cluttered and tight environments.

8.3.3 Comparison with parallelized A*-Search in C-Space

An opportunistic resolution complete planner is developed at the Institute for Process Control and Robotics (IPR) in Karlsruhe, see Wurll et al. (1997), Wörn et al. (1998). Based on an implicit c-space grid, an adapted A*-search is performed. It works with multiple resolutions for the c-space grid to increase the performance, i.e. to minimize the number of required nodes in the search graph. The resulution is adapted using efficient hierarchical distance calculation (see Henrich and Cheng (1992)). As a result nodes in c-space are skipped if the robot is at a sufficient distance to the obstacles in the workspace.

To further increase the performance, the algorithm can run in a parallelized version on a dedicated cluster of workstations. Parts of the c-space grid are assigned to the computing nodes based on cyclic heuristic schemes and the A*-search is distributed. The planned path is neither shortened nor does it assure any explicit safety distances, thus only planning of collision-free paths is considered in the following.

Based on the considerations made for two-dimensional configuration spaces

published in Hwang (1996), a set of benchmark sceneries for a 6-DOF manipulator was developed and the geometry and kinematic data was published on the internet. As a result the experiments reported can be reproduced very precisely. Note that there are no results of the A*-search for more than 6 DOF. This marks an upper limit for the abilities of the A*-search. The number of nodes that has to be expanded to escape local minima increases exponentially, thus requiring too much memory to store the search tree if planning for robots with more joints.

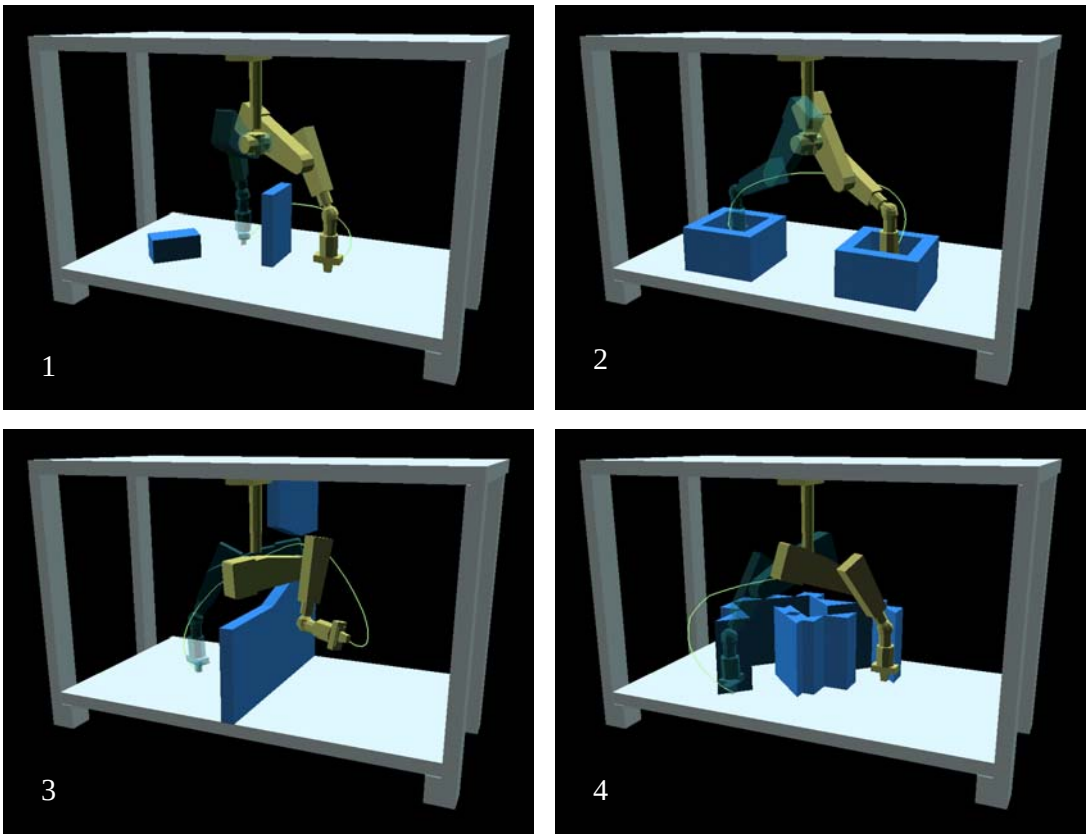

Figure 8.11: The Karlsruhe benchmark sceneries and solutions planned with the BB-method. A PUMA 260 manipulator with gripper is mounted top-down in a workcell. 1: Scenery SIMPLE. The robot has to pass a small obstacle. 2: Scenery STAR. The topology of the c-space can be assumed to be star-shaped, the robot has to retract, move towards the goal, and extend again. 3: Scenery BOTTLENECK. A tight passage has to be passed to reach the goal. 4: Scenery DETOUR. A maze-like group of obstacles allows to plan a very complicated path or to pass above the set of obstacles.

Four sceneries were used for the tests, they are illustrated in fig. 8.11. They are all based on the same workcell equipped with a top-down mounted PUMA

260 with 6 joints. Scenery SIMPLE is a very basic task where a small obstacle has to be passed (8.11.1). Scenery STAR is "starshaped": the robot has to be retracted, moved to a position above the goal and extended there (8.11.2). Scenery BOTTLENECK containes a tight passage that has to be found and passed (8.11.3). Scenery DETOUR implies two possible solutions: either a maze-like path is planned among the group of obstacles, or the robot avoids the detour and moves over all the obstacles towards the goal.

Scenery	τ	γ	$\delta_{\max}$	$\delta_{\min}$	t_{scale}
SIMPLE	5	5	200	20	**0.06**
STAR	5	5	200	20	**0.59**
BOTTLENECK	5	5	200	20	**0.65**
DETOUR	5	5	200	20	**0.70**

Table 8.17: Planning times for collision-free path planning in the Karlsruhe benchmark sceneries.

The planning times achieved with the BB-method with standard parameter values are listed in table 8.17. These parameters were efficient in other sceneries, it was not attempted to find "best" parameters.

For the sceneries SIMPLE, STAR and BOTTLENECK, the planning times are about a third of the results achieved with the parallelized A*-search running on a dedicated cluster of 8 workstations (each of it of approximately half the power as the machine used for the BB-method). In case of the scenery detour, the BB-method is about 17 times faster – no planned paths of the A*-search are published, thus it can only be speculated that this planner actually takes the "detour".

Again it has been shown that the BB-method is a suitable planning scheme for real-time motion planning for industrial manipulators in arbitrary environments, using standard hardware only.

8.3.4 Comparison with *ZZ*-Planning with Slidesteps

The only planner that is capable to plan for redundant and hyperredundant manipulators and is available for comparison is the current implementation of the ZZ-method which was developed by Glavina (1991) and further developed and optimized by the author (see Baginski (1996, 1998)). The method is explained in section 2.7, p. 24.

The prototype implementation of the BB-method and the current ZZ implementation are integrated in one program, using a common graphical interface and the same geometry and task data as input. This makes it possible to compare the performance of the planning algorithms exactly.

ZZ-Planning						*BB-Method*
Scenery/Task	DOF	Runs	*avg. sg.*	$\overline{t}$	t_{max}	t_{scale}
TABLE3	6	1000	**137.1**	**36.4**	**242**	**0.25**
GATE	16	100	**33.8**	**139.0**	**636**	**1.24**
ONE GATE	31	10	**293.1**	**9317.7**	**23480**	**9.57**

Table 8.18: Comparison for three selected tasks/sceneries. "Runs" indicates the number of planning runs that were measured to calculate the average time and average number of subgoals. The BB-method solves complex planning tasks two to three orders of magnitude faster than the ZZ-method.

Three tasks in three sceneries out of the above reported series of experiments were planned with the ZZ-method. The selected tasks are TABLE3 for the 6-DOF manipulator (see 8.2.1, p. 129) and the 16-DOF manipulator and the 31-DOF manipulator passing the gate (see 8.2.6, p. 140).

The ZZ-scheme is able to solve all of these tasks, but none of it can be solved locally. With increasing complexity, the required computational effort grows rapidly. The 31-DOF task is solved almost 1000 times faster with the BB-method.

This shows that the effort put into local planning within the BB-method is justified for manipulators with many degrees of freedom. If a solution can be found locally, the relative effort becomes smaller the larger the number of dimensions of the search space grows.

Chapter 9

Conclusion

The goals set up in the introduction of this work were reached, as we have shown, with the results achieved with the BB-method. The planning scheme was developed by utilizing *the* real motion *of chained rigid bodies among obstacles. The configuration space is abstract: the modifications applied to the path were seen as a* projection *of workspace features into the robot's control space.*

This chapter summarizes and discusses the results of this work. The aspects that had fallen short are pointed out, and ideas are given to further develop and extend this motion planning concept.

The discussion picks up the requirements, objectives and desired extensions developed in the first chapter. The results are analysed with respect to their practical usefulness within the frame set up for a motion planning system.

The final section presents some interesting aspects that are promising for future work with the BB-method beyond the possibilities discussed with regard to the desired extensions. Ideas are sketched to apply or extend this work towards related areas of robotic applications.

9.1 Discussion of Results

9.1.1 Requirements

The requirements were set up to capture the practical necessities. Any "global" preprocessing should be avoided to allow planning for changing environments and changing loads, and the motion planner should be independent from any specific kinematic structure.

Requirement R1 – Use the Available Geometry Data

The developed schemes are completely based on geometry data and do not build up an internal world model prior to planning. Thus they are perfectly suited to work in arbitrary environments and with ever changing loads, as it is typical for manipulator applications. A higher level planning system that controls the geometric world model can create tasks by rearranging the obstacles and assigning desired loads to the robot, i.e. making it part of its foremost link. The motion planning is able to immediately plan based upon this data.

The preprocessing that is required for the robot itself is minimal. Each link requires a "tip point" for the candidate creation. This can either be set manually or be assigned automatically to a suitable point within the current link and the succeeding joint axis. The "seed point" for the scaling can as well be created automatically. The robot links have to be decomposed into convex (starshaped) subparts or the model has to be replaced by the convex hull. Both can easily be realized, and have to be done only once.

The loads itself have to be processed analogously: either planning works with the convex hull, or the load is decomposed into suitable subparts in a fixed arrangement. Each subpart has an individual scaling origin and is assigned an individual index that assures a sufficient "scaling sequence", i.e. each part is scaled with respect to its origin that is situated in a lower indexed part.

The major benefit of the planning based on geometry data is the achievable precision: planning can start off and end "in touch" with obstacles. Planning approaches that use an internal model usually transform with an inherit loss of precision, and the possible application is limited to "gross motion planning". The tasks have to start and end in relatively free space, and additional planners are required for approach and departure. The BB-method is unlimited in its application and plans gross and fine motion simultaneously, thus it is perfectly suited to cooperate with grasp planners.

Requirement R2 – Plan Independent of a Specific Kinematic Structure

No consideration of any specific arrangement of joints and links was included in the planning algorithms. The practical results show that the planning is applicable to all kinds of manipulators. This is the power of the developed candidate calculation scheme: it is built upon its intended effect (orthogonal diplacement in the workspace) rather than considering the robot to find approriate displacements, e.g. by using the jacobian to relate cartesian and joint motion. The approximation scheme takes *any* pair of configurations and finds a set of candidates that suits the demands: all joints are varied, thus the full subspace of possible displacement is explored. And the candidates are placed approximately equidistant in a W-orthogonal plane, thus promising for rating improvement and well suited for relative comparison. But they do not necessarily explore all possible directions within the W-plane: if the kinematics do not allow to move in a certain direction, it is implicitly not even attempted at all to find a displacement in this direction.

Thus the planning scheme uses the abilities of the manipulator it plans for. And this is the best way to handle arbitrary kinematic structures: no rules have to be created, no dedicated kinematic equations have to be solved, planning can start when the forward chained model is created. A possible application that can especially benefit from this efficient simplicity is a simulation system used to develop new manipulators. In any stage, motions can be immediately planned to

evaluate the possible abilities of the robot under construction.

9.1.2 Objectives

The objectives were identified as the most important features of a basic motion planning system. Beside the obvious collision avoidance, two fundamental additional qualities were selected for their importance: the clearance between the moving robot and the obstacles, and the relative length of the motion in the c-space, the robot's technical control space. And there is one aspect of the practical applicability9 that is worth to form an objective of its own: high efficiency.

Objective O1 – Plan a Collision-Free Motion

The local planning based on path segment rating and incremental modification is presumably the most powerful scheme developed for manipulator motion planning so far. It is capable to "reshape" paths even through very tight passages in spaces of high dimension, i.e. for robots with a lot of joints. Its ability to handle forked kinematic chains and multifingered tools are unprecedented, no previously published motion planning system showed comparable features.

The planning scheme can be seen as a combination of (virtual) retraction that is "replaced" step by step by real retraction or avoidance motion. This novel approach is perfecly suited for manipulators with a lot of joints – as more joints are available, retraction or collision avoidance is *easier* in practice, and the local planning uses this successfully. By construction, an arbitrary number of joints can be handled without an exploding computational load. Especially for realistic devices with up to a maximum of 30 to 40 joints, the developed scheme is certain to work with suitable efficiency.

The planning is based on heuristics and local rating and therefore it is subject to fail. The use of a limited number of random subgoals has shown very good results. If local planning failed due to the inherent coarseness of the scheme – only a very limited set of the infinite number of all possible path variations is considered – very often a solution can be found with acceptable effort.

Summarizing, the BB-method fulfils objective **O1** for the desired areas of application, and for robot motion planning in general.

Objective O2 – Keep a Specified Safety Distance Wherever Possible

The idea of collision-free motion planning – incrementally increase the space that is available for the robot to move – is "continued" beyond the real size of the robot to plan a "motion quality". The BB-method employs the mechanisms of collision-free path planning and adds additional components to handle the inherent difficulties of safety distance planning.

The BB-method is presumably the first motion planning system that explicitly *plans* the safety distance. Other approaches try to keep certain distances, but do neither handle upper or lower thresholds nor do they treat the distance as a part

of the result. Within the BB-method, each path segment is attributed with the appropriate data, the possible safety distance of each individual link.

The objective was shown to be fulfilled with the developed planning system. For a real application, a more complex model of relations might be required, but its realization can be based on the existing schemes. The safety distances were modeled to compensate for modelling and control uncertainties. The control uncertainties are captured well with safety distances, as a trajectory generation system can use them as tolerance parameter when moving along a C-straight path segment. But modelling uncertainties might differ for several objects, depending on the model source and its state within the world model. Static workspace obstacles might be known exactly, known parts placed in the workcell are modeled exactly, but their position is somewhat uncertain, and sensor-scanned models are uncertain in shape and position. A matrix of relative safety distances, possibly extended by a *collision probability modelling* for sensor based data, can replace the purely link-dependent desired distance. The BB-method proves that applications like this are possible in practice.

Objective O3 – Prefer Short Motions

There is no possibility to measure the length of a motion by changing the size of the robot's geometry model. But the means of expansion can nonetheless be used to assure that the path keeps its "quality" when it is shortened. Thus, the realized path length reduction is a combination of polygon manipulation in the configuration space and path segment verification with expanded models in the workspace.

The realized scheme is a very basic one, but shows very good results. The formulated objective is fulfilled, the paths are shortened considerably with high efficiency. It can easily be extended to fulfil more sophisticated demands. One aspect that is worth to be considered is the relative "weightening" of the joint axes. Consider the following situation: based on the real energy consumption of the links, paths might be cheaper if lower joints are turned less and obstacle avoidance should preferably be done by the upper joints. Accordingly weightened joint axes will implicitly "flatten" triangles that imply a motion in the upper joints, and any directional change of lower joints results in "sharper" triangles. Because of that more effort will be spent to reduce these more expensive directional changes.

Another extension worth regarding is the "path shape". The path length reduction does not remove sharp turns from the path. This could be achieved if the implicit rating is not only based on the path length, but also considers the C-angles between adjacent path segments. This could move intersection configurations close to sharp turbs "outwards", and the overall required directional change can be distributed over a series of intersection configuration.

Objective O4 – Plan Fast

The planning scheme itself is highly efficient, the results achieved show that real-time planning (a few seconds at maximum) is possible for real industrial manipulators in obstructed environments. Fast path planning is also achieved for redundant and hyperredundant manipulators in arbitrary environments. Among the published results the BB-method is rated at the very top.

The effective performance was found to be roughly proportional to the "task difficulty". That means that tasks which require longer detours and more directional changes take longer to plan, compared to tasks that require less modifications of the initial path. But the most influential factor in regard to computational effort is the complexity of the geometric modelling.

The novel scheme of swept volume collision detection can reduce the required effort considerably. For relatively simple sceneries a speedup of the factor 3 to 5 can be achieved with respect to schemes typically used in the motion planning literature. For more complex modeled sceneries, the swept volume collision scheme can be extended to fully work with hierarchical models, and a further increase in efficiency is possible.

9.1.3 Extensions

Two extensions were discussed in the first chapter that should be possible in future developement. Interactive planning is required to allow the higher level planning system to gain more influence on the planning process, and additional constraints should be includable in the planning.

Extension E1 – Allow Interactive Planning

The BB-method can take *any* path, either colliding or not, and either with safety distance data or not, and start planning immediately. Any sequence of intersection configurations can be further processed. The intersection configurations can either be fixed, and planning consists of a sequence of local planning runs. Or the intersection configurations are unfixed (with the exception of start and goal), and are used to form an initial path, i.e. the input data is subject of modification.

This is a unique feature of the BB-method: It is able to improve any kind of partial/incomplete or uncertain solution. Consider a higher level planning system that stores approximations of collision-free paths (e.g. in neural networks, see Kinder (1995) or Eldracher (1995)). It can pass a "sketch" of the solution to the motion planner, that will be refined to a feasible path – and this will certainly be faster than a complete replanning based on start and goal only. And it is well suited for changing loads: the replanning adapts the path accordingly.

Interaction in a human operated system is also possible up to any desired degree. As the path is reshaped incrementally and according to intuitively understandable schemes, the operator can supervise the planning and interactively influence it. Among the possibilities to do so, the following are sketched:

- Extend the task by a sequence of intersection configurations.
- Define preferred displacement directions. Within the candidate creation, certain cartesian directions can easily be preferred, i.e. only z-positive displacements are accepted if the operater wants to assure a collision avoidance motion *above* the obstacles.
- If local planning fails, create appropriate intersection configurations or path segments to pass the failure location and planning can be continued.

In summary, the BB-method is perfectly suited for interactive motion planning.

Extension E2 – Allow Planning Under Additional Constraints

Constraints limit the movability of a manipulator. The task constraints considered in the first chapter do not influence the joint configurations, but the cartesian position of the tool. They are usually defined with a certain tolerance, i.e. the z-axis of the tool should be aligned with an axis in the world frame plus/minus a certain angle.

Looking at the BB-method, the planning of short paths *is* path modification under a constraint – the constraint is to avoid collisions. This can be seen as an abstract principle: try to modify the path to improve it with respect to a certain rating, and keep other ratings at least equal. This allows to include constraints in several ways if they are modeled as rating functions Q_c. The initial path can be modified (without checking for collisions) until the constraint holds for the path $\hat{\mathbf{P}}$, i.e. $Q_c(\hat{\mathbf{P}})$ falls below the tolerance threshold. In the next phase, collision avoidance is planned, and the constraint is supervised. It can also be done the other way round: plan a collision-free path first, and then modify it according to the constraint without shifting it into collision again.

The planning under task constraints is a promising field of future research. The BB-method supplies basic means to address certain constraint planning problems efficiently.

9.2 Suggestions for Future Work

This section sketches some ideas to extend the scope of applications of the BB-method beyond the suggestions of the previous section.

Planning with Partially Specified Goal Positions

One aspect that has not been considered as part of the basic motion planning system is the planning of *single configurations*. The goal of a task might only be specified as a cartesian position for the tool instead of being specified as a full configuration. This requires the planning of an appropriate goal configuration prior to motion planning. If the manipulator is non-redundant, the number of

solutions of the *inverse kinematics* is fixed. Possible goal configurations are found using collision detection at the respective discrete configurations.

A continuous subspace of the c-space that solves the inverse kinematics exists for redundant manipulators – the so-called *nullspace*, see e.g. Yoshikawa (1990) for mathematical details. Based on an arbitrary, possibly colliding solution for a given cartesian tool position, displacement steps can be applied within the nullspace to improve the rating of the configuration. It can be moved out of collision and even pushed away to keep additional safety distances before motion planning itself starts. As the rating of single configurations is much cheaper than rating of complete path segments, this type of configuration planning can be expected to be highly efficient.

Planning for Given Cartesian Trajectories

The above idea can be extended to use the BB-method for the planning of *cartesian trajectories*, another aspect that has not been considered as part of a basic motion planning system. A task planning system might create a tool trajectory for a given task, e.g. applying paint to an object. The position and orientation of the tool are given for the complete motion and cannot be varied (this is the strongest possible constraint, see above **E2**). Collisions might occur for the robot's links. If the robot is redundant, planning is possible to avoid them.

An initial path can be found as a suitable dense sequence of configurations and all modification can be applied within the respective nullspace of these configurations until no more collision is found along the path. Note that neither start nor goal need to be fixed configurations – they can also be altered in their nullspace.

Modified and more general schemes can be employed for planning under constraints. If only a limited number of tool coordinates is fixed, a respective "nullspace" of higher dimension can be used as the subspace available for path displacement, and alternative path segments have to be found in a way that the constraint holds for the complete path.

Using the BB-Method for Digital Mock-Up

A completely different domain of possible application is digital mock-up (DMU). In the industrial construction, DMU is used to evaluate possible assembly sequences of complex structures in a purely virtual environment. Consider the construction of an automobile: the different parts are tightly packed and assembly often requires very complex "motions" for the individual parts. Today's DMU systems offer collision detection only. The operator attempts to virtually "move" the part, and the system indicates interferences. This is time-consuming and ineffective work.

Automated motion planning for "free flying" parts can be realized with the BB-method if a "seed point" for the scaling of an arbitrary complex arrangement

of parts can be moved without collision (otherwise, the respective path segment and alternatives in the neighbourhood are rated zero and no successful modification is possible). But a possible trajectory for a single point is usually easy to find, either automatically or by a small set of points preset by the user.

For DMU, extended rating functions might be useful to handle (typically non-convex) assembly parts. A possibility is to scale several subparts simultaneously, or to set up even more complex dependencies. The path modification/candidate creation has to be adapted to handle three translational and three (unlimited) rotational configuration parameters. As for manipulator motion planning, successful concepts can certainly be developed if the object itself and its motion among the obstacles is examined and utilized.

References

Ackermann, J., & Hirzinger, G. (1996). Institute for robotics and system dynamics. *IEEE Robotics and Automation Magazine*, 47–51. [Cited on pp. 4, 134]

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1983). *Data structures and algorithms.* Addison-Wesley. [Cited on p. 16]

Amato, N. M. (1996). A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 113–120). Minneapolis, Minnesota. [Cited on p. 20]

Baginski, B. (1996). The Z^3–method for fast path planning in dynamic environments. In *Proceedings of IASTED Conference Applications of Control and Robotics* (pp. 47–52). Orlando, Florida. [Cited on pp. 24, 151]

Baginski, B. (1998). *Fortschritte bei der Bahnplanung mit dem ZZ-Verfahren: Parallelisierung und Weiterentwicklungen* (Tech. Rep. No. TUM *in print*). Munich, Germany: Institut für Informatik, Technische Universität München. [Cited on pp. 24, 73, 125, 151]

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. (1993). *The quickhull algorithm for convex hull* (Tech. Rep. No. GCG53). Minneapolis, Minnesota: The Geometry Center, University of Minnesota. [Cited on p. 51]

Barraquand, J., Langlois, B., & Latombe, J.-C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man and Cybernetics*, *22*(2), 224-241. [Cited on p. 23]

Barraquand, J., & Latombe, J.-C. (1990). A monte-carlo algorithm for path planning with many degrees of freedom. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 1712–1717). Cincinnati, Ohio. [Cited on p. 23]

Berchtold, S., & Glavina, B. (1994). A scalable optimizer for automatically generated manipulator motions. In *Proccedings of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS'94* (pp. 1796–1802). Munich, Germany. [Cited on pp. 25, 96]

Bodduluri, R. M. C. (1990). *Design and planned movement of multi-degree of freedom spatial mechanisms.* Doctoral dissertation, University of California, Irvine, California. [Cited on p. 21]

Braun, B., & Corsépius, R. (1996). AMOS: Schnelle Manipulator-Bewegungsplanung durch Integration potentialfeldbasierter lokaler und probabilistischer globaler Algorithmen. In G. Schmidt & F. Freyberger (Eds.), *Proceedings of 12. Fachgespräche über Autonome Mobile Systeme* (pp. 150–159). Munich, Germany: Springer Verlag. [Cited on p. 24]

Brooks, R. A. (1983). Planning collision-free motions for pick-and-place operations. *Int. Journal of Robotics Research, MIT Press*, *2*(4), 19–44. [Cited on p. 3]

Buckley, C. E. (1989). A foundation for the "flexible–trajectory" approach to numeric path planning. *Int. Journal of Robotics Research, MIT Press*, *8–3*, 44–64. [Cited on p. 27]

Buckley, C. E., & Leifer, L. J. (1985). A proximity metric for continuum path planning. In *9th International Conference on Artificial Intelligence* (pp. 1096–1102). Los Angeles, California. [Cited on p. 27]

Cameron, S. (1997). Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 3112–3117). Albuquerque, New Mexico. [Cited on p. 27]

Canny, J. F. (1988). *The complexity of robot motion planning.* Cambridge, Massachusetts: MIT Press. [Cited on p. 16]

Challou, D., Boley, D., Gini, M., & Kumar, V. (1995). A parallel formulation of informed randomized search for robot motion planning problems. In *Proceedings of IEEE Conference on Robotics and Automation.* Nagoya, Japan. [Cited on pp. 23, 146]

Chang, H. (1996). A new technique to handle local minimum for imperfect potential field based motion planning. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 108–112). Minneapolis, Minnesota. [Cited on p. 23]

Chen, P. C., & Hwang, Y. K. (1992). SANDROS: A motion planner with performance proportional to task difficulty. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2346–2353). Nice, France. [Cited on pp. 21, 23, 108]

Chen, P. C., & Hwang, Y. K. (1996). *Performance of the SANDROS planner* (Tech. Rep.). Albuquerque, New Mexico: Sandia National Laboratories. [Cited on p. 21]

Cheung, E., & Lumelsky, V. (1990). Motion planning for a whole-sensitiv robot arm manipulator. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 344–349). Cincinnati, Ohio. [Cited on p. 28]

Cohen, J., Manocha, D., & Olano, M. (1997). *Simplifying polygonal models using successive mappings* (Tech. Rep. No. TR97-011). Chapel Hill, North Carolina: University of North Carolina. [Cited on p. 125]

Craig, J. J. (1986). *Introduction to robotics.* Addison-Wesley. [Cited on pp. 8, 31]

Dai, F. (1989). Collision–free motion of an articulated kinematic chain in a dynamic environment. *IEEE Computer Graphics & Applications*, *9*(1), 70–74. [Cited on p. 27]

Daxwanger, W., Ettelt, E., Fischer, C., Freyberger, F., Hanebeck, U., & Schmidt, G. (1996). ROMAN: Ein mobiler Serviceroboter als persönlicher Assistent in belebten Innenräumen. In G. Schmidt & F. Freyberger (Eds.), *Proceedings of 12. Fachgespräche über Autonome Mobile Systeme* (pp. 314–333). Munich, Germany: Springer Verlag. [Cited on p. 143]

Duelen, G., & Willnow, C. (1991). Path planning of transfer motions for industrial robots by heuristically controlled decomposition of the configuration space. In *Proceedings of the European Robotics and Intelligent System Conference EURISCON'91* (pp. 217–224). Corfu, Greece. [Cited on p. 21]

Dupont, P. E., & Derby, S. (1988). An algorithm for CAD-based generation of collision–free robot paths. In B. Ravani (Ed.), *CAD Based Programming for Sensory Robots* (Vol. F50, pp. 433–465). Springer Verlag. [Cited on pp. 26, 27]

Eldracher, M. (1994). Neural subgoal generation with subgoal graph: An approach. In *Proceedings of World Conference on Neural Networks WCNN '94* (pp. II-142 – II-146). Portland, Oregon. [Cited on p. 20]

Eldracher, M. (1995). *Planung kinematischer Trajektorien für Manipulatoren mit Hilfe von Subzielen und neuronalen Netzen.* Doctoral dissertation, Technische Universität München, Munich, Germany. [Cited on pp. 20, 157]

Fink, B., & Wend, H.-D. (1991). Fast collision–free motion–planning for robot manipulators based on parallelized algorithms. In *Proceedings of Symposium on Robot Control* (pp. 681–686). Vienna, Austria. [Cited on p. 18]

Fischer, M. (1996). *Aufgabenspezifikation und Bahnplanung für kooperierende Manipulatoren.* Doctoral dissertation, Technische Universität München, Munich, Germany. [Cited on p. 35]

Gilbert, E. G., & Ong, C. J. (1994). New distances for the separation and penetration of objects. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 579–586). San Diego, California. [Cited on p. 27]

Gini, M. (1996). Parallel search algorithms for robot motion planning. In K. Gupta & A. P. del Pobil (Eds.), *Practical Motion Planning in Robotics, Workshop WT2 at the IEEE International Conference on Robotics and Automation.* Minneapolis, Minnesota. [Cited on pp. 146, 147]

Glavina, B. (1990). Solving findpath by combination of goal-directed and randomized search. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 1718–1723). Cincinatti, Ohio. [Cited on p. 24]

Glavina, B. (1991). *Planung kollisionsfreier Bewegungen für Manipulatoren durch Kombination von zielgerichteter Suche und zufallsgesteuerter Zwischenzielerzeugung.* Doctoral dissertation, Technische Universität München, Munich, Germany. [Cited on pp. 24, 108, 151]

Gottschalk, S., Lin, M., & Manocha, D. (1996). OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of ACM Siggraph '96.* New Orleans, Louisiana. [Cited on p. 125]

Gupta, K. K., & Guo, Z. (1995). Motion planning for many degrees of freedom: Sequential search with backtracking. *IEEE Transactions on Robotics and Automation*, *11* (6), 897–906. [Cited on p. 26]

Gupta, K. K., & Zhu, X. (1994). Practical global motion planning for many degrees of freedom: A novel approach within sequential framework. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2038–2043). San Diego, California. [Cited on p. 26]

Henrich, D., & Cheng, X. (1992). Fast distance computation for on-line collision detection with multi-arm robots. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2514–2519). Nice, France. [Cited on p. 149]

Hirzinger, G. (1993). Rotex – the first space robot technology experiment. In *Preprints of the Third International Symposium on Experimental Robotics* (pp. 302–320). Kyoto, Japan. [Cited on p. 4]

Hörmann, A., & Rembold, U. (1991). Development of an advanced robot for autonomous assembly. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2452–2457). Sacramento, California. [Cited on p. 4]

Horsch, T., Schwarz, F., & Tolle, H. (1994). Motion planning with many degrees of freedom - random reflections at c-space obstacles. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 3318–3323). San Diego, California. [Cited on p. 20]

Hwang, Y. K. (1996). Completeness vs. efficiency in real applications of motion planning. In K. Gupta & A. P. del Pobil (Eds.), *Practical Motion Planning in Robotics, Workshop WT2 at the IEEE International Conference on Robotics and Automation.* Minneapolis, Minnesota. [Cited on pp. 146, 149]

Hwang, Y. K., & Ahuja, N. (1992). Gross motion planning — a survey. *ACM Computing Surveys*, *24*(3), 219–291. [Cited on pp. 15, 17]

Kavraki, L., & Latombe, J.-C. (1994). Randomized preprocessing of configuration space for fast motion planning. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2138–2145). San Diego, California. [Cited on p. 20]

Kavraki, L., Svestka, P., Latombe, J.-C., & Overmars, M. (1994). *Probabilistic roadmaps for path planning in high-dimensional configuration spaces* (Tech. Rep. No. UU-CS-1994-32). Utrecht, Netherlands: Utrecht University. [Cited on p. 20]

Kavraki, L. E., Kolountzakis, M. N., & Latombe, J.-C. (1996). Analysis of probabilistic roadmaps for path planning. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 3020–3025). Minneapolis, Minnesota. [Cited on p. 74]

Kinder, M. (1995). *Pfadplanung für Manipulatoren in komplexen Umgebungen mittels generalisierender Pfadspeicherung in Ellipsoidkarten.* Doctoral dissertation, Technische Universität München, Munich, Germany. [Cited on p. 157]

Koga, Y., & Latombe, J.-C. (1994). On multi-arm manipulation planning. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 945–952). San Diego, California. [Cited on p. 35]

Kugelmann, D. (1994). Autonomous robotic handling applying sensor systems and 3D simulation. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 196–201). San Diego, California. [Cited on p. 24]

Lamiraux, F., & Laumond, J. P. (1996). On the expected complexity of random path planning. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 3014–3019). Minneapolis, Minnesota. [Cited on p. 74]

Latombe, J.-C. (1991). *Robot motion planning.* Kluver Academic Publishers. [Cited on pp. 2, 15, 16, 17, 23, 29, 37]

Latombe, J.-C. (1993). Geometry and search in motion planning. *Annals of Mathemathics and Artificial Intelligence*, *8*, 215–227. [Cited on p. 18]

Liu, J. (1996). Spatial reasoning about robot compliant movements and optimal paths in qualitatively modeled environments. *Int. Journal of Robotics Research, MIT Press*, *15*(2), 181–210. [Cited on p. 6]

Lozano-Pérez, T. (1986). A simple motion planning algorithm for general manipulators. In *Proceedings of national conference on artificial intelligence AAAI '86* (pp. 626–631). Philadelphia, Pennsylvania. [Cited on pp. 18, 19]

McLean, A. (1994). A framework for global planning for redundant manipulators. In *Proccedings of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS'94* (pp. 525–530). Munich, Germany. [Cited on p. 26]

McLean, A., & Cameron, S. (1996). The virtual springs method: Path planning and collision avoidance for redundant manipulators. *Int. Journal of Robotics Research, MIT Press*, *15*(4), 300–319. [Cited on p. 26]

Mukerjee, A. (1996). *Neat vs. scruffy: A review of computational models for spatial expressions* (Tech. Rep.). Kanpur, India: Center for Robotics, Indian Institute of Technology. [Cited on p. 6]

Ong, C. J. (1995). Properties of penetration between general objects. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2293–2298). Nagoya, Japan. [Cited on p. 27]

Ong, C. J., & Gilbert, E. G. (1994). Robot path planning with penetration growth distance. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 2146–2152). San Diego, California. [Cited on p. 27]

Overgaard, L., Petersen, H. G., & Perram, J. W. (1994). Motion planning for an articulated robot: A multi–agent approach. In *Distributed Software Agents and Applications, 6th European Workshop on Modelling Autonomous Agents in a Multi Agent World, MAAMAW '94* (pp. 206–219). Odense, Denmark. [Cited on p. 26]

Overmars, M. H., & Svestka, P. (1994). *A probabilistic learning approach to motion planning* (Tech. Rep. No. UU-CS-1994-03). Utrcht, Netherlands: Utrecht University. [Cited on pp. 20, 108]

Paljug, E., Ohm, T., & Hayati, S. (1995). The JPL serpentine robot: a 12 dof system for inspection. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 3143–3148). Nagoya, Japan. [Cited on pp. 3, 45]

Pobil, A. P. del, & Serna, M. A. (1995). *Spatial representation and motion planning.* Berlin, Heidelberg, New York: Springer. [Cited on pp. 21, 125]

Pobil, A. P. del, Serna, M. A., & Llovet, J. (1992). A new representation for collision avoidance and detection. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 246–251). Nice, France. [Cited on p. 21]

Quinlan, S. (1994). *Real-time modification of collision-free paths.* Doctoral dissertation, Stanford University, Palo Alto, California. [Cited on pp. 108, 125]

Ralli, E., & Hirzinger, G. (1994). Fast path planning for robot manipulators using numerical potential fields in the configuration space. In *Proccedings of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS'94* (pp. 1922–1929). München, Germany. [Cited on p. 19]

Ralli, E., & Hirzinger, G. (1996). A global and resolution complete path planner for up to 6DOF robot manipulators. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 3295–3302). Minneapolis, Minnesota. [Cited on p. 19]

Reif, J. H. (1979). Complexity of the mover's problem and generalizations. In *Proceedings of 20th IEEE Symposium on Foundations of Computer Science* (pp. 421–427). Los Alamitos, California: CS Press. [Cited on p. 16]

Reznik, D., & Lumelsky, V. (1995). Sensor-based motion planning in three dimensions for a highly redundant snake robot. *Advanced Robotics*, *9*(3), 255–280. [Cited on p. 26]

Schwartz, J. T., Sharir, M., & Hopcroft, J. (1987). *Planning, geometry and complexity of robot motion.* Norwood, New Jersey: Ablex. [Cited on p. 16]

Sciavicco, L., & Siciliano, B. (1996). *Modelling and control of robot manipulators.* McGraw-Hill. [Cited on p. 8]

Stryk, O. von, & Schlemmer, M. (1993). *Optimal control of the industrial robot manutec r3* (Tech. Rep. No. 467). Technische Universität München, Mathematisches Institut. [Cited on p. 8]

Tarokh, M. (1996). Implementation of a fast path planner on an industrial manipulator. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 436–441). Minneapolis, Minnesota. [Cited on p. 28]

Tarokh, M., & Hourtash, A. (1997). Implementation of a fast path planner on an industrial manipulator. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 562–568). Albuquerque, New Mexico. [Cited on p. 28]

Warren, C. W. (1989). Global path planning using artificial potential fields. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 316–321). Scottsdale, Arizona. [Cited on p. 19]

Warren, C. W. (1990). Multiple robot path coordination using artificial potential fields. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 500–505). Cincinnati, Ohio. [Cited on p. 19]

Warren, C. W. (1993). Fast path planning using modified A∗ method. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 662–667). San Diego, California. [Cited on p. 22]

Wörn, H., Henrich, D., & Wurll, C. (1998). Motion planning in dynamic environments - a parallel online approach. In *Proceedings of 3rd International Symposium on Artificial Life and Robotics.* Oita, Japan. [Cited on pp. 22, 149]

Wurll, C., Henrich, D., & Wörn, H. (1997). *Parallele Bewegungsplanung in dynamischen Umgebungen* (Tech. Rep. No. 20/97). Karlsruhe, Germany: Institut für Informatik, Universität Karlsruhe. [Cited on pp. 22, 146, 149]

Xavier, P. G. (1997). Fast swept-volume distance for robust collision detection. In *Proceedings of IEEE Conference on Robotics and Automation* (pp. 1162–1169). Albuquerque, New Mexico. [Cited on p. 125]

Yoshikawa, T. (1990). *Foundations of robotics.* Cambridge, Massachusetts: MIT Press. [Cited on pp. 29, 159]

Zhang, J. (1994). *Ein integriertes Verfahren zur effizienten Planung und Ausführung von Roboterbewegungen in unscharfen Umgebungen.* Doctoral dissertation, Universität Karlsruhe, Karlsruhe, Germany. [Cited on p. 18]

Zhang, Y., & Münch, H. (1993). Modellgestützte optimale Bewegungsplanung für Industrieroboter. In *Intelligente Steuerung und Regelung von Robotern* (pp. 219–230). Langen, Germany. [Cited on p. 18]

T

U

V

W

Z