University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems (TAMS)

Diploma thesis

A navigation algorithm based on laser scans and stereo vision for a servicerobot in a cluttered and dynamic office environment

presented in
October 2006

David Melnychuk

Carl-Petersen-Strasse 36
20535 Hamburg

9dannber@informatik.uni-hamburg.de
Matriculation number: 5201598

Primary supervisor: Prof. Dr. Jianwei Zhang
Secondary supervisor: Prof. Dr. Bernd Neumann

Contents

List of figures **III**

1. About this work **1**
- 1.1. Overview 1
- 1.2. Motivation 2
- 1.3. The Robot 2
- 1.4. Related work 5
  - 1.4.1. Planning methods 5
  - 1.4.2. Methods for three dimensional perception and mapping 6

2. Path planning **9**
- 2.1. Configuration space 9
  - 2.1.1. Basic notions 9
  - 2.1.2. Expansion of obstacles 10
  - 2.1.3. Properties of the employed configuration space 12
- 2.2. Visibility graph 14
  - 2.2.1. Description 14
  - 2.2.2. Comparison with other planning algorithms 15
- 2.3. The two stage approach 16
  - 2.3.1. Detection of narrow passages 18
  - 2.3.2. Planning in narrow passages 19
- 2.4. Computation of expansions 24
- 2.5. Experiments 27
  - 2.5.1. Testing different potential functions 28
  - 2.5.2. Testing different turn costs 29
- 2.6. Summary 32

3. Integrating sensory data **33**
- 3.1. Reacting to a changing environment 33
- 3.2. Integrating data from laser range finders 34
- 3.3. Obstacle memory 35

4. Using stereo cameras **41**
- 4.1. Natural stereo vision 41
- 4.2. The camera model 43
  - 4.2.1. The pinhole model 43

- 4.2.2. Internal parameters . . . 44
- 4.2.3. External parameters . . . 45
- 4.2.4. Radial distortion . . . 46

4.3. Camera calibration . . . 49
4.4. Geometry of two parallel cameras . . . 50
4.5. Rectification . . . 52
4.6. Stereo correspondence analysis . . . 53
4.7. Line segment extraction and processing . . . 55
- 4.7.1. Edge detection . . . 55
- 4.7.2. Hough transform . . . 58
- 4.7.3. Line segmentation . . . 59
- 4.7.4. Line segment matching . . . 60

4.8. Summary . . . 61

5. Results and conclusion 63
5.1. Planning and movement . . . 63
5.2. Obstacle sensing and stereometry . . . 64

Bibliography 67

A. Experimental planning tasks 73

B. Video recording 75
B.1. Scene 1 - The storage room . . . 75
B.2. Scene 2 - Passing a small passage . . . 75
B.3. Scene 3 - Obstacle avoidance . . . 75
B.4. Scene 4 - Dynamic pathplanning . . . 76
B.5. Scene 4 - Dynamic obstacle avoidance in a cluttered environment . . . 76

C. Erklärung / Statement 77
C.1. Deutsch . . . 77
C.2. English . . . 77

List of Figures

1.1. The robot TASER . . . 3
1.2. Scan areas (Top view) . . . 4

2.1. Shrinking of a rotationally symmetric robot . . . 11
2.2. Different shapes of expanded obstacles . . . 11
2.3. Ground and top plane of the relevant space . . . 12
2.4. A visibility graph . . . 14
2.5. A tangent graph . . . 15
2.6. The shortest path between p and q on the tangent graph . . . 16
2.7. Stretching a path to a wider expansion . . . 18
2.8. Overlapping expansions . . . 19
2.9. Possible orientations with different neighbor relations . . . 20
2.10. Cutting from a configuration space graph . . . 21
2.11. Planning results with different potential functions . . . 23
2.12. Computing the expansion by moving the robot around the obstacle . . . 24
2.13. The expansion of a point obstacle is the point reflected robot outline . . . 25
2.14. Moving the point expansion along a line . . . 26
2.15. The expansion as a convex hull . . . 26
2.16. Experimental results for planing with different potential functions . . . 30
2.17. Experimental results for planing with different αs . . . 31

3.1. Robot “trapped” between two obstacles . . . 37
3.2. Robot oscillates between two doors . . . 38
3.3. Robot oscillates between two rooms, because static map data is missing . . . 39

4.1. The basic pinhole camera model . . . 44
4.2. The basic pinhole camera model . . . 45
4.3. Radial distortion . . . 47
4.4. Undistortion performed with forward mapping . . . 48
4.5. Calibration scene . . . 50
4.6. Photographing a scene with two cameras . . . 51
4.7. Axis parallel stereo setup (top view) . . . 52
4.8. The Laplacian of a Gaussian . . . 56
4.9. The original chair scene . . . 57
4.10. The chair scene after edge detection . . . 57
4.11. The Hesse normal form . . . 59

4.12. The chair scene after Hough transform 60

5.1. A stereo image pair 65

A.1. Planning tasks 1, 2 and 3 74
A.2. Planning tasks 4, 5 and 6 74
A.3. Planning task 7 74

1 About this work

1.1. Overview

This diploma thesis describes an approach to create a robust navigational algorithm for a wheeled service robot working in an office environment. The approach taken in this work involves two major subproblems. The first one is the planning problem: How to create a safe path from the current position to the robot's destination efficiently, based on the current representation of the environment? The second one is the mapping problem: How can the robot's representation of the environment be modified, based on sensor data, so that it contains all information which is relevant for path planning?

Chapter 2 describes the planning algorithm used. It is based on a combination of the tangent graph method and exhaustive search of the configuration space. The tangent graph is used to quickly create a preliminary path, while the space search deals with narrow passages which require exact planning. The results from both approaches are united into one final path. This two-staged approach results in reliable planning results even in difficult planning situations, and yet it does not require much planning time.

As for the environment representation, the robot possesses a fixed inchoate map of the environment from the start. This circumvents the need for complex map building. In addition, sensory data from laser range finders and a stereo vision system is used to create an accurate representation of the robot's environment. Thus movable or transient obstacles and potential map inaccuracies can be accounted for. The problem of integrating sensory data into an existing representation is discussed in chapter 3.

A pair of stereo cameras is used to produce disparity images. From those, depth information about objects in front of the cameras can be obtained. With this information, additional obstacles like tables which cannot be detected reliably by the laser range finders can be found and considered in path planning. The process of generating depth information from stereo images is described in chapter 4.

The last chapter briefly states the results achieved. When using only laser range finder data, the aim of a reliable navigation algorithm was achieved. It can handle a changing environment, but there remains the constraint that there must not be any obstacles which appear much smaller in the scan than they really are. But the stereo vision approach failed to produce useable measurements. It demands a highly accurate camera calibration process, which was not reached during this work. Measurements therefore deviated up to 30% from the real values - too much to be useful for navigation purposes. This chapter will state possible reasons for this failure.

Furthermore, suggegtions will be given about what can be done in a future work to resolve this problem and ideas for a few other enhancements will be put forward.

1.2. Motivation

The TAMS[1] research group from the Department of Informatics at the University of Hamburg possesses a mobile service robot named TASER[2]. At the time the work for this thesis started, the path planning algorithm for this robot had some serious limitations, e.g. it could not pass doorways (except for very wide ones). Obviously a service robot which is not able to leave a room is of little use, hence the need for a better navigation algorithm.

Another limitation of the old algorithm was that the determined path could not be modified in the wake of new sensory data. Should there be a previously unknown obstacle in the robot's way, the robot would simply stop and wait until its path is clear again.

Thus the aim of this thesis was to create a path planning algorithm which would enable the robot to move safely from one room to another, even when there is relatively few space to manoeuvre somewhere along the path, as may happen in a doorway. The robot must not collide with any obstacles in its way. It should adapt its path to new obstacles which were not known to the robot at the beginning of the planning process, as well as to moveable obstacles like humans.

Planning on a static representation of the environment is clearly not enough to reach this aim. The robot must retrieve information from its environment directly via sensors and this information must be somehow incorporated into the planning process.

It is not intended that the robot manipulates its environment to achieve its goal. Thus if the only door to a room is closed, the robot would not try to open the door, but path planning would simply fail.

1.3. The Robot

The robot TASER (see figure 1.1) is propelled by a two wheel drive, the speed of each wheel can be set separately. While this would enable to robot the perform any curve on a plane ground, the control software is currently only able to move the robot forward or backward or to turn its orientation while not driving.

Two laser range finders[3] are attached to the robot some 20 cm above the ground, one at the front and one at the rear. Each scanner performs a 180° scan of the environment in a resolution of 0.5°. The space between the scanners exactly to the left and the right of the robot cannot be scanned, as it is outside of the range of both scanners (see figure 1.2). Apart from this blind area, the distances to all obstacles surrounding the robot on the plane 20 cm above the ground are known.

---

[1] Technical aspects of multimodal systems

[2] TAMS Service Robot

[3] SICK LMS 200

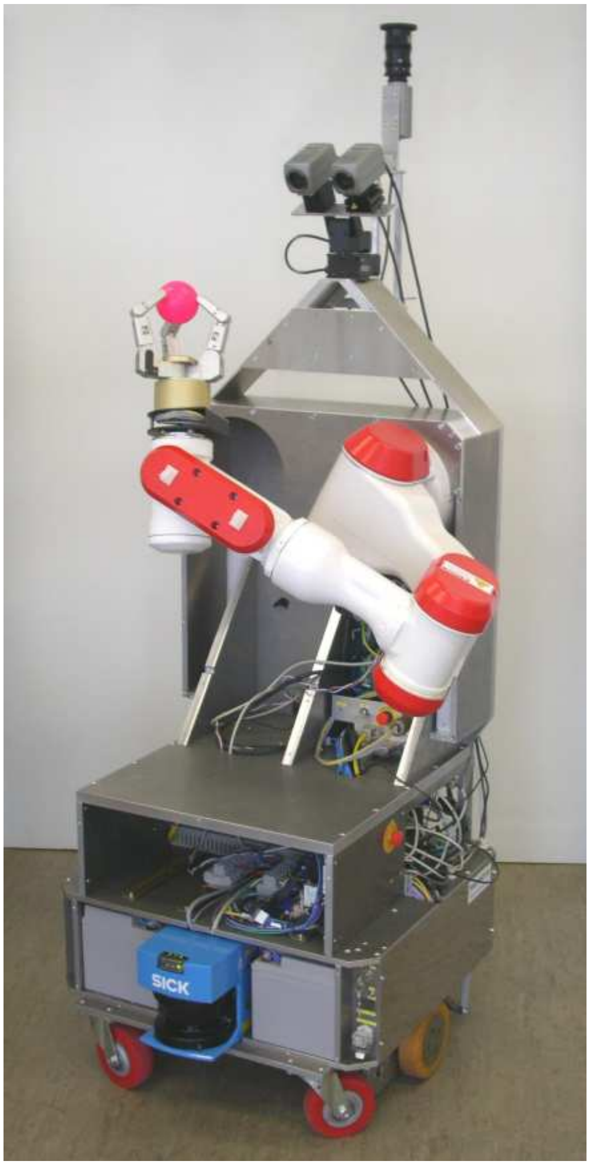

Figure 1.1.: The robot TASER
The laser range finder is the blue box at the bottom with “SICK” on it
The cameras can be seen at the top

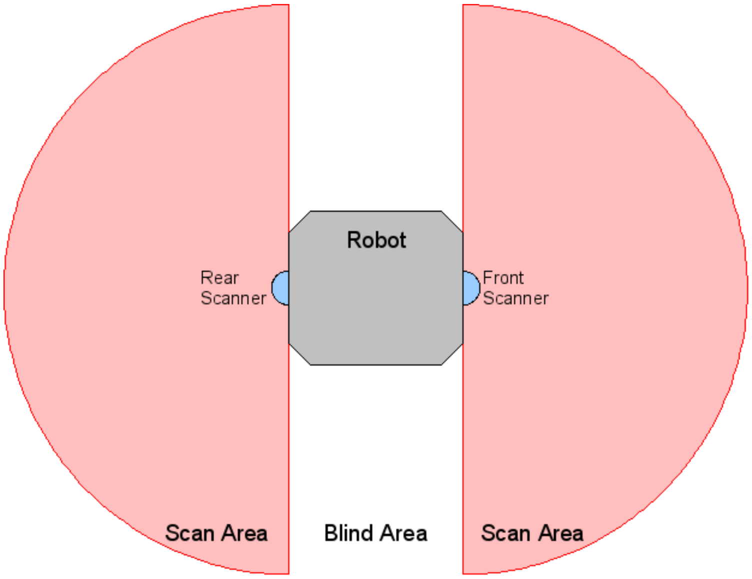

Figure 1.2.: Scan areas (Top view)

Two color cameras[4] are attached approximately in parallel to a moveable pan-tilt-unit at the top of the robot. They provide the optical data for the stereo vision system. Each camera produces frames with a resolution up to 640 × 480 pixel and up to 30 frames per second.

The other sensors and the robot's arms are of no concern to this thesis.

The robot has a PC-class computer on board to control all of its devices. It has a wireless network interface and uses the Roblet©-Framework [WSS+04] provided by the genRob-Project as control software. This software allows for control programs to be executed on remote computers. Time consuming tasks should be done on a remote computer, as not to overload the on-board computer and delay time critical tasks. On the other hand, sometimes tasks must be executed locally, since the network interface may be too slow for some applications. For instance, the stereo cameras create up to 150 Mbps of data, which would be too much for realtime analysis over current wireless networks.

A working localization algorithm has already been developed [SW02] for the robot. Therefore, in this thesis the position of the robot is assumed to be known (with an accuracy of about 3 cm) and this thesis is not about a SLAM[5] problem in the narrow sense of the word. The localization algorithm uses data from odometry for pose estimation. Odometry errors are corrected by measuring the distance to laser beacons on fixed known positions. Kalman filters [Kal60] are used to integrate the data.

---

[4] Sony DFW-VL500

[5] SLAM: Self localization and map building, a problem common to mobile robots with no prior information about their environment

1.4. Related work

Autonomous robot navigation has been extensively researched in recent years. Major interrelated subproblems to this task are processing of sensory data, map building and planning. In absence of any prior information of the environment an exploration strategy is also necessary. Some have been presented in [LGB02, OVFT04, OF05]. Furthemore, for robots with more complicated movement devices like walking or non holonomic robots, the execution of the planned path also may be a complex task.

In [SW02] a navigation system has been proposed for a similar robot. However, it has some limitations. The path planning algorithm uses a quite generous approximation of the robot as a circle, which makes the algorithm simpler but less reliable in narrow passages. Furthermore newly detected obstacles in the robot's path cause it simply to halt, instead of planning a new path around that obstacle. Finally, the only sensor used is a laser range finder scanning in a plane, with the same problems resulting from this fact as stated in the paragraph above. This work can be largely seen as a continuation of [SW02], with the focus on overcoming the limitations of the path planning algorithm proposed therein.

1.4.1. Planning methods

[Lat98] and [CLH$^+$05] provide an overview about established robot planning algorithms. The field of path planning is still under investigation, and new papers are published regularly. Some of these recent works will be presented briefly.

In [NN04], a sensor-based planning algorithm is presented for robots which cannot sense an obstacle at a distance, i.e. it detects obstacles only when it (almost) touches them. This would be the case if the robot has only haptic sensors. The classic algorithm for this situation is the *Bug*-Algorithm (see [Lat98, CLH$^+$05]). The focus of the cited work is the development of an algorithm which produces shorter paths on the average than algorithms known hitherto. These algorithms are not very useful if the robot has means to detect obstacles from a distance, e.g. by optical sensors or sonar scanners.

Path planning problems in general are known to be PSPACE hard [HW86]. Therefore complete planning algorithms suffer from excessively high computational cost and memory usage. Probabilistic planning methods can alleviate this problem, because they do not search the entire state space. Instead they probe some random samples of the state space, which *probably* contain a valid path.

An approach which combines cell decomposition with probabilistic sampling is presented in [Lin04]. The author states about his approach:

> “PCD [Probabilistic Cell Decomposition] is easily scalable and applicable to many different kinds of problems. Experimental results show that PCD performs well under various conditions.”

Another probabilistic approach was made in [KM04]. Here probabilistic roadmaps (PRM) are combined with a harmonic potential field. This approach is named “Harmonic Function Probabilistic Roadmap” (HFPRM). The author stresses, that whereas usual PRM methods perform poorly in the presence of narrow passages, HFPRM overcomes this weakness. This property perhaps would make HFPRM also a suitable method for the planning task for this thesis. However, the thorough investigation into alternative planning methods would be out of the scope of this thesis. Besides, the results of the planning algorithm used for this thesis are satisfactional.

An interesting approach using genetic algorithms for path planning was proposed in [HY04].

For this work, however, such sophisticated planning algorithms are not needed, because no high dimensional configuration spaces are involved and computation times resulting from traditional algorithms are acceptable.

1.4.2. Methods for three dimensional perception and mapping

To achieve a reliable navigation system in cluttered environments it is indispensable for the robot to percept obstacles in its environment in all three dimensions and not only in one plane, as many simple robots do. The reason for this is that there are certain obstacles which may have quite different appearances, depending of the plane the scan is conducted in. Overhanging obstacles and tables fall into this category. Such objects could block a much wider region than what can be seen in the plane scan.

This section will present a selection of recent publications that describe approaches to three dimensional environment perception and mapping. Many more approaches can be found in journals and conference reports on robotics, and the publications mentioned below also reference various further approaches.

In [BWW03] a tiltable laser range finder is used for SLAM on an autonomous outdoor vehicle. 3D perception is used to retrieve data from the environment, but the localization and planning algorithm work in a two dimensional space. This makes it possible to get a detailed view of the environment and still make use of the rich pool of established fast-performing 2D planning algorithms. Another work [WACW04] with partly the same authors deals with indoor mapping also using a tiltable laser range finder. In an approach similar to this work, 3D sensory data is projected into a plane before being passed to the planning stage.

Yet another work with many similarities to this one is [MJ97]. A robot named "Spinoza" uses depth images from a trinocular stereo vision system to explore and map its environment. The planning algorithm used is also very similar to the one used in this work (see section 2.3.2). It is a variant of the occupancy grid method, in which the robot's environment is represented as a regular grid. By using a potential field, the search for the a path tends to avoid grid nodes close to obstacles.

The navigation system for the humanoid robot "QRIO" [SFG$^+$04] also works in a similar way. A stereo vision system is used to gain information about its environment. The path planning system uses an occupancy grid with a potential field generating a repulsive force away from obstacles. This field is incorporated into an A^*-search on the grid.

A paper which addresses the need for mobile robots to detect specifically tables is [VVB05]. It spends a lot of effort to estimate the position and orientation of the ground plane (i.e. the plane the robot moves on). This is not necessary for this thesis, because the stereo cameras are at a fixed known height and angle above the ground, therefore the ground plane's position and orientation can be easily computed.

A stereo vision approach to the SLAM problem is presented in [SE04]. The approach tries to estimate the current robot movement by extracting features from 3D-point clouds and matching these features against features taken at a different pose. A matching score is calculated to filter out bad possible matches. From the remaining matches a movement esimation is generated. This is done by caluclating the 3D-point cloud that would be expected to be seen as a result of a particular movement. Various possible movements are considered and the movement that produces the cloud that matches the actual observed cloud best is selected as the movement estimate. The approach also generates a map of the environment using the point clouds and the pose estimates. Local mapping errors are eliminated by observing a consistency criterion and employing a map update strategy.

Path planning

2

This chapter addresses the central task of planning a path through the robot's environment. It will give a general overview of the concepts and methods for path planning which were employed in this work. There will be a detailed description of how these methods were used to solve this particular planning task. The chapter will explain the two stage approach, which is a combination of the tangent graph method and occupancy grid planning. By combining both methods it is possible to generate desireable paths in a short time.

One important aspect of planning is choosing the right representation of the robot and the obstacles. Because the planning algorithm should work in cluttered environments where narrow passages are commonplace, an exact representation is crucial. This topic explains in detail, how the representation of the environment is constructed.

Finally, experimental results for off-line planning[1] will be presented. The experiments show how the generated paths react to changes to adjustable parameters of the planning algorithm.

2.1. Configuration space

2.1.1. Basic notions

Most path planning algorithms operate on a configuration space. A good explanation of the concept of configuration space is given in [CLH+05]:

> "The *configuration* of a robot is a complete specification of every point of that system. The *configuration space* [...] of the robot system is the space of all possible configurations of the system. Thus a configuration is simply point in this abstract configuration space. [...] The number of *degrees of freedom* of a robot system is the dimension of the configuration space, or the minimum number of parameters needed to specify the configuration."

The degrees of freedom indicate in how many directions the robot can be moved or oriented. In a typical service robot scenario the robot can move freely in a plane (two directions) and can turn along the axis orthogonal to the plane. Thus there are three degrees of freedom and the configuration space is three dimensional. If the robot can be approximated as rotationally

---

[1] i.e. planning on a static map without sensor input

symmetric the orientation of the robot does not matter and its configuration space can be reduced to two dimensions.

Usually the robot's environment contains obstacles, i.e. objects which the robot must not collide with. Obstacles in the environment can be mapped onto configuration space obstacles. These are sets of points in the configuration space which correspondent with a configuration of the robot that would result in a collision.

Let C denote the configuration space and CO_i the configuration space obstacles. Consider the following definitions, which are made according to [CLH$^+$05]:

Definition 2.1.1 *The set* C_{occ} *of all points that are occupied by configuration space obstacles is* $C_{occ} = \bigcup_i CO_i$

Definition 2.1.2 *A path* p *in* C *is a continuous mapping* $p : [0,1] \to C$

Given these definitions the problem of path planning between the starting configuration $q_{start} \in C$ and the desired final configuration $q_{end} \in C$ can be described as finding a path p with $p(0) = q_{start}$, $p(1) = q_{end}$, and $\forall x \in [0,1] : p(x) \notin C_{occ}$. Such a path is collision free. Obviously there is not such path if either $q_{start} \in C_{occ}$ or $q_{end} \in C_{occ}$.

A path planning algorithm is an algorithm that produces a path given C, C_{occ}, q_{start} and q_{end}. It is called correct, if the path produced is always collision free. Alternatively the algorithm may terminate with an error indicating no path exists. The algorithm is called complete if it always returns a path if a path exists and always returns an error otherwise. The algorithm is called optimal, if the returned path fulfills some optimality criterion, for example that the path returned has the shortest length of all existing paths.

For the service robot scenario the most important property of the algorithm is correctness, because collisions simply must not occur, since collisions could damage or even destroy the robot or something valuable in its environment. The properties of completeness and optimality do not have to be met strictly. In practice it is acceptable for the algorithms to fail in some awkward situations that occur rarely. Likewise it is acceptable that the path found is less than optimal, as long as it does not get too bad in respect of the optimality criterion.

2.1.2. Expansion of obstacles

Path planning algorithms can be made simpler if the robot is represented as only one point in the configuration space. This point will be called *reduction point*. To still create collision free paths, the configuration space obstacles must be expanded by the same amount of space the robot has been shrunk. The simple case of a two dimensional configuration space and a rotationally symmetric robot shrunk to a point is shown in figure 2.1. Section 2.4 will cover in detail how the expansion is computed.

If the robot is not rotationally symmetric, the configuration space requires one more dimension for the orientation of the robot. The expanded obstacles have a different shape for each possible orientation of the robot (see figure 2.2).

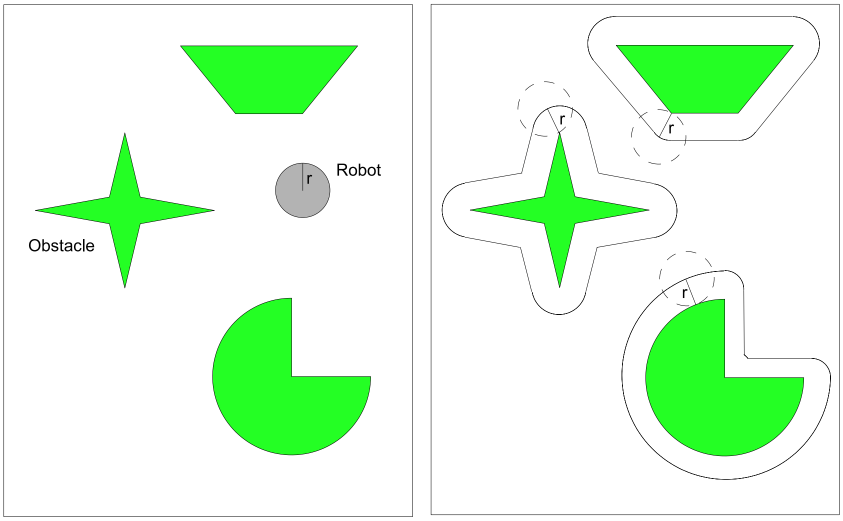

Figure 2.1.: Shrinking of a rotationally symmetric robot to a point in the configuration space

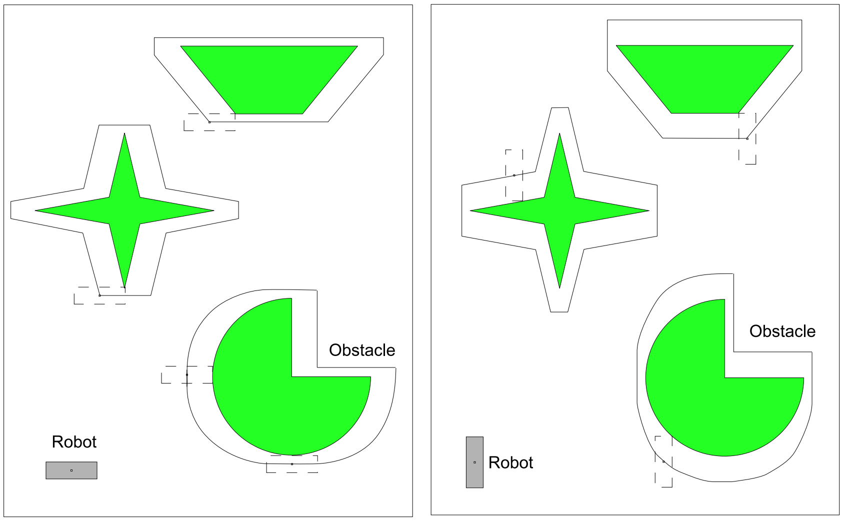

Figure 2.2.: Different shapes of expanded obstacles depending on the robot's orientation

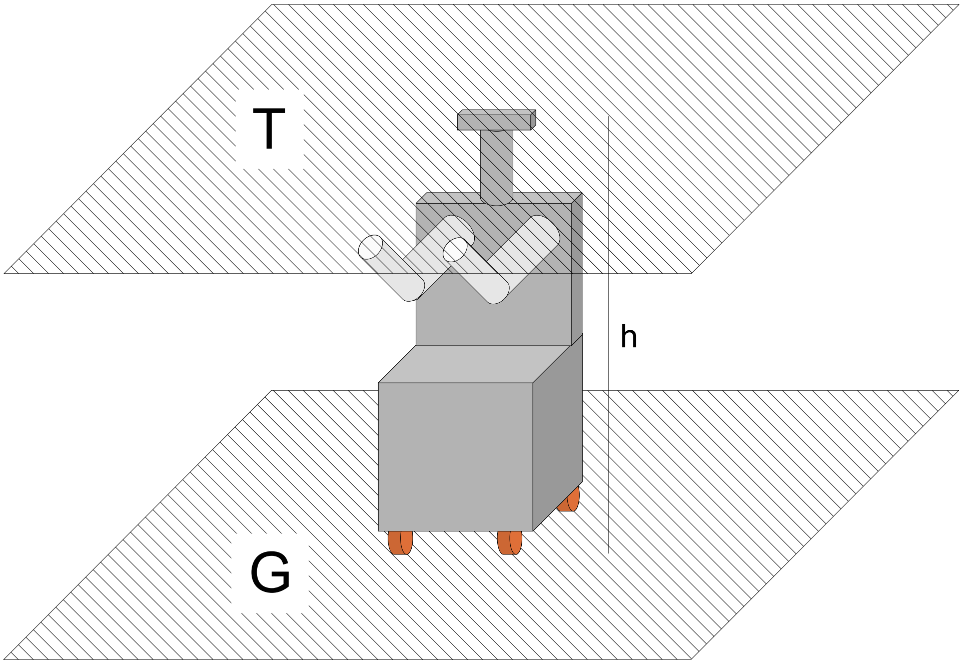

Figure 2.3.: Ground and top plane of the relevant space

Thus there are as many different planar configuration spaces as there are possible robot orientations. If the robot is able to perform arbitrarily small changes of orientation, this would result in an infinite number of planar configuration spaces. All these planar configuration spaces can be assumed as stacked one on another and thus composing a single three dimensional configuration space.

2.1.3. Properties of the employed configuration space

The robot has three degrees of freedom (movement in the plane and orientation), thus the configuration space should be three dimensional. The robot's outline is approximately rectangular, with the longest diameter being approximately 20 cm longer than the shortest. If the robot was modeled as a circle with the long diameter, a path through a narrow doorway would appear to be blocked, even if in reality there is enough space left to navigate. Unfortunately, doorways with such a diameter are common in the robot's environment, thus such a solution is not satisfactory.

If a shorter diameter is chosen, a path with a collision might be generated. For example, the algorithm could bring the robot into a doorway and make it turn there. If the doorway is small, the robot might fit just into it, but a collision would occur upon turning.

As a result, the robot cannot be assumed as rotationally symmetric, and the configuration space must remain three dimensional if we want to robot to navigate trough narrow passages.

Until now, no mention was made of the robot's height nor of the obstacles' heights. When the robot's representation is being shrunk to a point, one also must account for its height. The solution to this is simple. Consider the plane G, which is the ground plane the robot moves at, and the plane T above G, as in figure 2.3. G and T are parallel and the distance between them is the same as the robot's height. Obviously only obstacles between these planes are of interest, and any obstacle within this room must have a counterpart in the configuration space. If there is an obstacle anywhere between those planes, the outline of this obstacle is marked as occupied in the configuration space.

More formally, G is described by the equation 2.1 and T by equation 2.2.

$$\begin{aligned} 0\cdot x+0\cdot y+z+0 &= 0 && (2.1)\\ 0\cdot x+0\cdot y+z-h &= 0 \quad \text{where } h \text{ is the robots height} && (2.2) \end{aligned}$$

G and H can be seen as sets of points which fulfill the equations. Thus $G=\{\vec{p}|\vec{p}=(x_0,x_1,0)^T\}$ and $T=\{\vec{p}|\vec{p}=(x_0,x_1,h)^T\}$ with arbitrary x_i.

An obstacle O_i in the three dimensional environment can be represented as the set of points it occupies. The relevant environment RE is the space between the planes G and T, that is $RE=\{(x_0,x_1,x_2)^T|0\leq x_2\leq h\}$. Now the occupied relevant environment RE_{occ} can be stated as in equation 2.3.

$$RE_{occ}=\bigcup_i O_i\cap RE \tag{2.3}$$

The next thing to determine is how the relevant occupied environment RE_{occ} is to be transformed into obstacles of the three dimensional configuration space ${C^3}_{occ}$. The three dimensional configuration space shall be denoted as C^3 and the configuration space without orientation as C^2.

It suffices to have at least one point obstructed by an obstacle along the line between $(x,y,0)^T$ and $(x,y,h)^T$ to mark the position $(x,y)^T$ as occupied in C^2. Thus the first step of the transformation is a projection of the obstacle points into C^2 as described by the mapping $proj$ in Definition 2.1.3. Equation 2.4 explains the relation between the occupied space in the environment and the occupied space ${C^2}_{occ}$ in C^2.

Definition 2.1.3 $proj:\mathbb{R}^3\rightarrow C^2, proj\begin{pmatrix}x_0\\x_1\\x_2\end{pmatrix}=\begin{pmatrix}x_0\\x_1\end{pmatrix}$

$${C^2}_{occ}=\{\vec{p}|\vec{p}=proj(\vec{q}),\vec{q}\in RE_{occ}\}\subseteq C^2 \tag{2.4}$$

The next step is the expansion of the obstacles to compansate for the shrinking of the robot. Definition 2.1.4 establishes a mapping exp which expands the occupied point in C^2 into a set of points in C^3 to this end. How the expansion must be facilitated has been described above in this section and will not be formalized in detail.

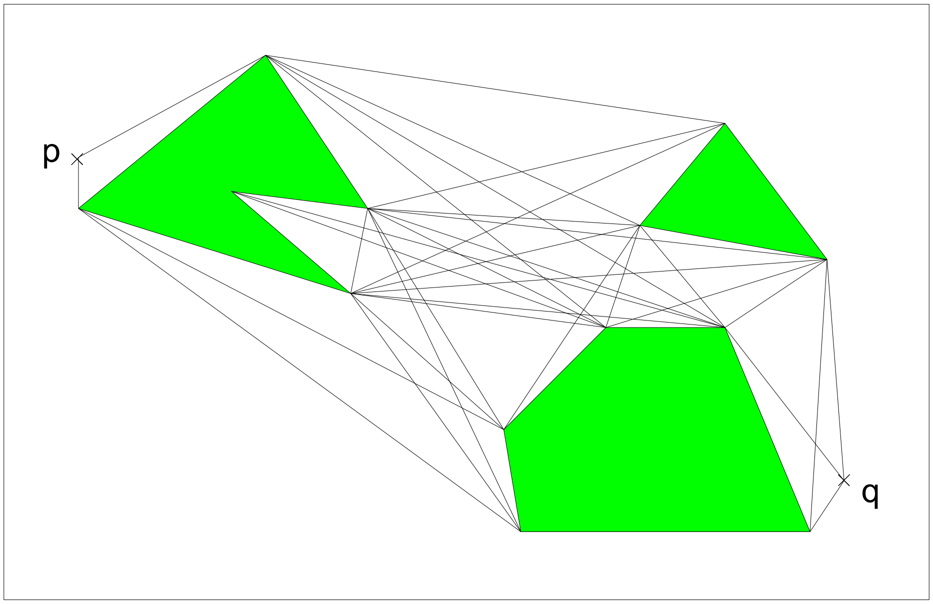

Figure 2.4.: A visibility graph

Definition 2.1.4 $exp : C^2 \rightarrow \mathcal{P}(C^3)$

Finally the occupied space in C^3 can be described by equation 2.5.

$$C^3{}_{occ} = \bigcup_i exp(\vec{p}_i) \quad \text{with} \quad \vec{p}_i \in C^2{}_{occ} \tag{2.5}$$

2.2. Visibility graph

The planning algorithm used by the navigation system is based on the well established visibility graph method [LPW79, Lat98]. The reasons why this method was chosen will be stated in section 2.2.2. But first, the algorithm will be introduced in the following section. To avoid unnecessary complexity, it will be presented for the two dimensional case. Furthermore, the algorithm is used in this work only in a two dimensional context. However, the visibility graph algorithm can easily be transfered for use with higher dimensional configuration spaces.

2.2.1. Description

The *visibility graph* method starts out with a map with polygonal obstacles and free space between them. Curved obstacle borders must be approximated by multiple lines. From this map a graph is constructed. For all points defining the polygons' borders a node is added to the

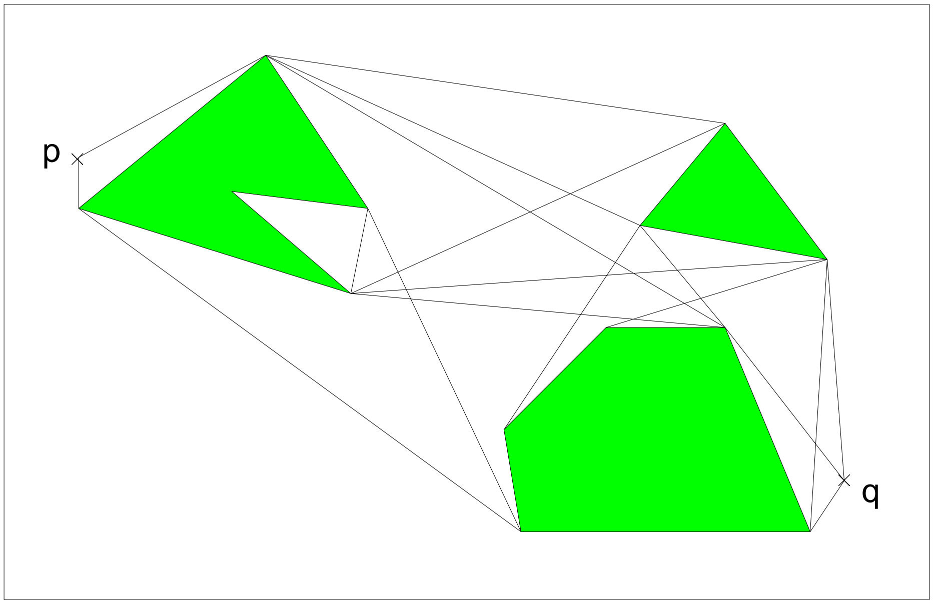

Figure 2.5.: A tangent graph

graph. Additionally, there is one node for the robots starting position p and one for the desired position q (see figure 2.4).

The graph contains all *free edges* between any two nodes. A *free edge* is an edge which does not pass through an obstacle. Thus all edges in the graph are possible routes for the robot, i.e. movements along these edges do not result in collisions. The name visibility graph is derived from the fact that a node is connected to all other visible nodes, i.e. all nodes which are not obstructed by an obstacle in the direct line of view.

Now it is possible to use a graph searching algorithm, like A^* [HNR68], to determine a (shortest) path between p and q. The cost of an edge can be set to be the distance between its nodes.

Some of the edges in the visibility graph are actually useless and can be omitted from the graph [LA92], thus reducing the number of operations needed during the search. An edge is unnecessary, if any arbitrary small expansion of the edge would intersect with an obstacle. The graph without such edges is called *reduced visibility graph* or *tangent graph*[2] (see figure 2.5).

2.2.2. Comparison with other planning algorithms

The visibility graph method was chosen because of its simplicity and small computational complexity. Furthermore the path it produces consists of straight lines with only few changes of

---

[2] This designation stems from the fact that all expansions of the edges in this graph are tangential to the obstacles they connect

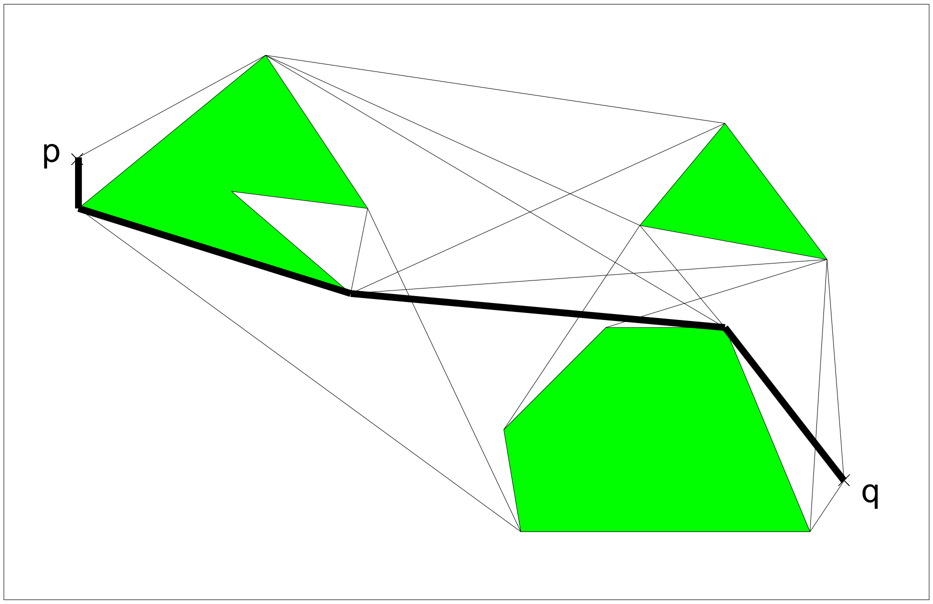

Figure 2.6.: The shortest path between p and q on the tangent graph

orientation. This is favorable because the robot is only able to move in straight lines (see section 1.3). If an algorithm had been used that produces curves or even erratic paths this would require the resulting paths to be straightend afterwards. This in turn would further increase complexity and could worsen the results.

The visibility graph method also has some weaknesses compared to other methods. But at the time the planning algorithm was drafted it seemed that these weaknesses would not be grave for this particular application or they could be overcome. The most serious weakness is that the resulting path comes as close to an obstacle as possible. In the area of robotics, where there always are inaccuracies in the representation of the environment as well as in the execution of a planned path, such a property is not very desirable.

All in all it seemed that the visibility graph may provide a good base for the planning algorithm. This assumption was proven to be correct by the achieved results.

2.3. The two stage approach

As already stated in the previous section, the main drawback of the visibility graph method is that it brings the robot very close to the obstacles. The straightforward usual solution to this is to expand the obstacles gratuitously so that there is enough space left to navigate the robot safely. The amount of the additional expansion depends on the accuracy of the map, localization, and locomotion. This is the approach taken by [SW02] in the previous navigation algorithm for TASER (see section 1.4).

Furthermore, planning in a two dimensional configuration space requires modeling the robot as a circle with a diameter equal to the longest diameter of the actual robot (see section 2.1.2). But this means that the expansion of an obstacle may be larger than needed, depending on the orientation of the robot. In the case of TASER, there are approximately 20 cm wasted. This may be acceptable if obstacles are scarce and there is enough space between them. But this is not how a typical indoor office environment looks like.

The problem is even more pointed with passages like doorways, because the wasted space occurs on both sides of a doorway. Thus with the visibility graph approach the robot could not pass a doorway, even if it is 40 cm wider than the robot.

The solution taken to overcome this weakness is the *two stage approach*. Planing is first conducted by the visibility graph method with minimal obstacle expansion. To ensure that there are no collisions, the path is modified in the second stage. To this end the path is divided into sections which are far away from obstacles and sections close to obstacles. Section 2.3.1 describes how the path is divided into sections and how sections are classified.

The sections of the path close to obstacles are re-planned with another algorithm, which is explained in section 2.3.2. This second algorithm is specialized to plan a path for sections with little free space left.

This two stage approach has been initially examined in a practical course by three students in 2005, one of them is the author of this thesis. During the work for this thesis, this approach was refined and completed.

The main advantage of the two stage approach is that a computational costly algorithm can be used for planning in narrow passages, but the overall computational cost of planning remains relatively small, because the costly algorithm is invoked only for a small area and not for entire map.

The drawbacks of the approach are in terms of completeness and optimality. The first stage (visibility graph) cannot know for sure whether there is a way through a narrow passage or not. It may turn out at the second stage, that a path cannot be found for the designated passage. As a consequence, the algorithm terminates with no path found, although there could be a path via a different passage. This situation will be called case “A” in the following.

Furthermore, there is the opposite case; this is when the first stage determines there is no passage between two obstacles, but the second actually may have found one if it had been invoked for this passage. If there is no other possible path apart from going through this passage, the result will be that the two stage approach finds no path altough one exists (case “B”).

It could be possible to remedy this situation, but it would involve much more computational cost and these situations occur in practice very rarely, so this effort was not undertaken. Still, a possible strategy will be described in short.

When the obstacle expansion at the first stage is chosen small enough, it is always the second stage which determines that a passage would be impassable and not the first, thus case “B” would not occur anymore. Instead, the frequency of the occurance of case “A” would increase. The associated problems could be resolved with some kind of feedback to the first stage. The second stage would notify the first stage that the designated passage is blocked. Then, the first

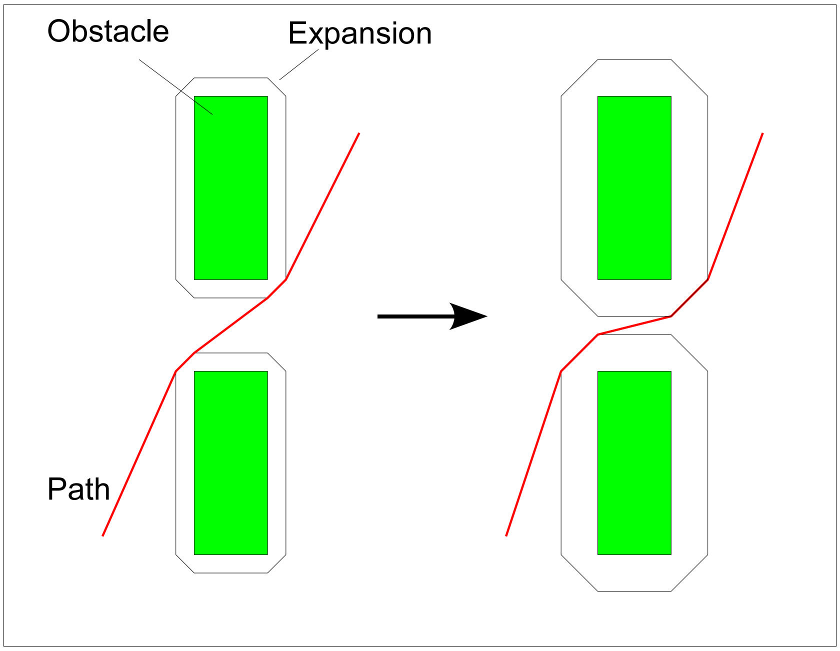

Figure 2.7.: Stretching a path to a wider expansion

stage could mark this passage as impassable in its map and start over again, thus generating a different route if possible.

Another way to handle this situation would be to simply extend the second stage to the entire map, but this would drive computational costs up and render the two stage approach absurd.

Optimality of the computed paths cannot be guaranteed for similar reasons. Still the resulting path cannot deviate too far from the optimal path, because the changes to the path resulting from the first stage, which is optimal in the first stage model, are restricted to a local area. So the resulting path may be less than optimal, but it will not be a very bad path.

2.3.1. Detection of narrow passages

After the first stage of planing determined a path using the visibility graph method with a small obstacle expansion, this path needs to be modified so that no collisions occur. This is done by using a wider obstacle expansion and stretching the path to fit with the new expansion. This can be done when there is enough free space between the obstacles (see figure 2.7).

But in the presence of another nearby obstacle, the result of stretching the path can be that the path lies within the expansion of the other obstacle. It is possible that there is no way to pass between both obstacles without intersecting either expansion, because both the expansions overlap (see figure 2.8).

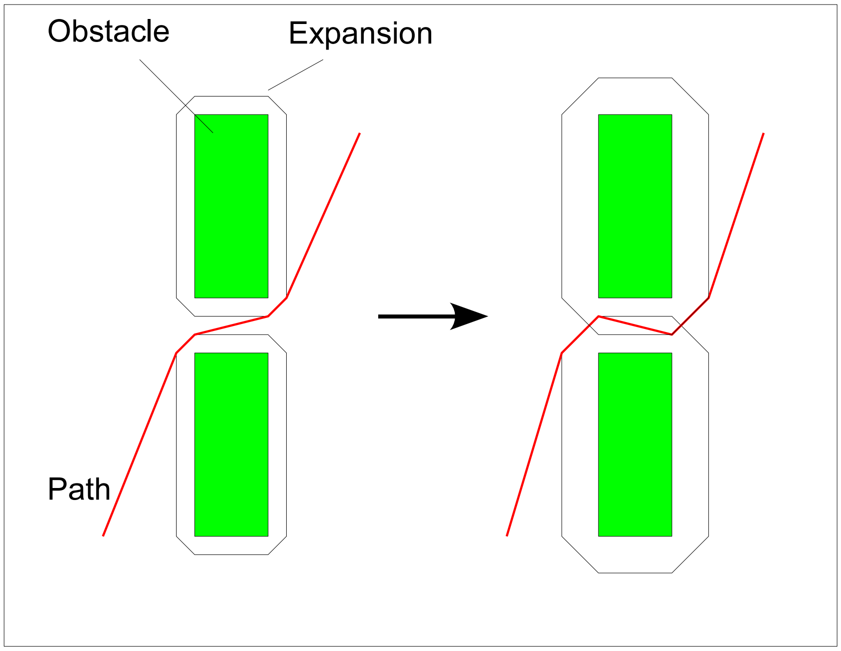

Figure 2.8.: Overlapping expansions

Sections of the map with intersections of multiple wide expansions of obstacles are called *narrow passages*. Sections of the path that lie within a narrow passage can be easily identified. All that is needed is to check whether the section intersects with some expansion after the path has been stretched to the wide expansions.

These sections of the path are passed to the second planning stage. The task of the second stage is to find a path from the entry point into the narrow passage to the exit point, where the path leaves the narrow passage. After the second stage found a path through the narrow passage, this path replaces the section of the original path.

2.3.2. Planning in narrow passages

The two stage approach allows the usage of a planning algorithms with quite high computational cost at the second stage without making the overall planning too much computationally expensive. One quite simple, yet powerful planning algorithm is the state space search.

It involves the creation of a graph with all possible configurations of the robot as the graph's nodes. The edges represent transitions from one configuration to another. Transitions are allowed only between neighboring configurations. The two neighboring configurations must not differ too much, so that a transition between these two can be done safely without further planning. A graph search, such as the A^*-Algorithm, can be used to find a path between the start and the end configuration.

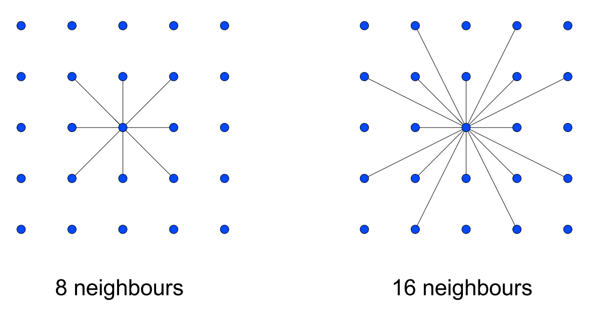

Figure 2.9.: Possible orientations with different neighbor relations

The main drawback of this algorithm is the high cost in terms of numbers of operations and memory used. But, as already mentioned, the impact of this is not very grave, because only a relatively small configuration space is searched.

The configuration space graph

Every node in the graph represents exactly one point in the configuration space. Thus the configuration space cannot be represented as a graph in continuous form, because this would require an unlimited number of nodes. Therefore a discretization of the configuration space is needed.

The graph resembles a grid with regular intervals between the nodes. The actual size of the interval can be set to different values, and intervals between 20 and 60 mm proved to be good tradeoffs between size and exactness of the representation. Nodes at occupied positions, i.e. positions that would result in an collision with an obstacle, are not included in the graph.

This approach to planning can be seen as a variant of the occupancy grid mapping method, which has been first used for mobile robots by Moravec and Elfes in [ME85] and [Elf89].

To account for effects of turning the robot as laid out in section 2.1.2, the configuration space must be three dimensional, with two dimensions for the position in the plane and one for the orientation. Each node of the graph has an associated configuration, which consists of a x and a y coordinate, and the robot's orientations. Nodes at blocked positions, i.e. positions that would result in an collision with an obstacle, are not included in the graph.

Within one plane of the graph's grid, all nodes have a configuration with the same orientation, thus they differ only in the position. The planes can be assumed as stacked on one atop the other, one plane for each possible orientation. Each node is connected with the node that lies next to it in the direction of the plane's orientation, representing the possible straight forward movement without changing the orientation. It is also connected to the two nodes representing the adjacent orientations at the same position. Thus the transitions between the graph's nodes represent the two basic motions the robot may execute, straight forward motion and rotation.

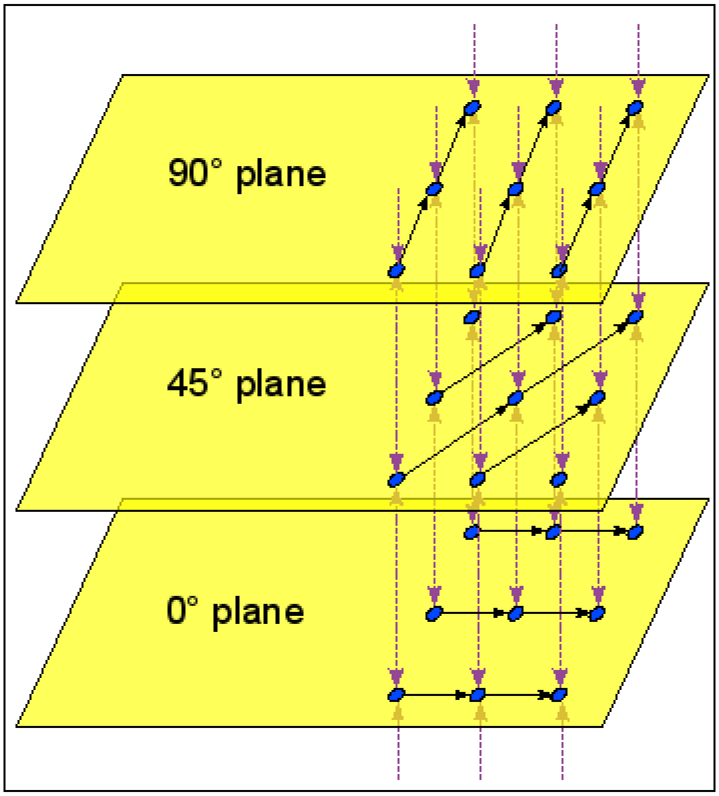

Figure 2.10.: Cutting from a configuration space graph

Due to the grid shaped graph, only selected orientations can be represented. Eight orientations are possible if position changes are restricted to immediate neighbors of the node. The other orientations would result in movements that end somewhere "between" the grid nodes. The current implementation uses transitions to immediate neighbors and to neighbors two steps away, thereby allowing for sixteen orientations (see figure 2.9).

The intention for the use of many orientations is to reduce the occurance of erratic movements. The path in a passage that requires a movment in a direction, which is not available to the planner, would require a discrete approximation. The result would be many small steps, which are interrupted by rotational movements, one to the left, the next to the right.

A cutting from a simplified configuration space graph with eight possible orientations is shown in figure 2.10. Note that the planes are connected in a circular way, as it is possible to move from the last configuration (at 315°) to the first (at 0°) in one step.

Backward motion would be possible, but is not implemented, because the stereo vision system for obstacle detection looks only in forward direction, so a backward motion would pose the risk of collisions.

Edge costs

The planing algorithm for narrow passages presented so far still lacks any provision to avoid paths which come close to obstacles. The search in the configuration space for the shortest path in fact tends to produce paths with little distance to obstacles, because going along the shortest path usually means to pass an obstacle with the shortest possible distance to it.

Here the flexibility of the graph search method makes it possible to change the semantics of what the best path actually is. The edges of the configuration space graph can be associated with costs, and as the search finds the path with the shortest *total* cost, it avoids using costly edges when there are cheaper alternative routes.

By associating high costs with undesirable edges, these edges will be less likely be part of the resulting path. Hence paths which come close to obstacles will be avoided, if edges close to obstacles have high costs.

This approach presented in this work has been inspired by the potential field planning method, which was pioneered by [AH83] and [Kha86]. The potential field method is a navigation algorithm based on a virtual field, which pulls the robot towards the destination and repels it away from obstacles. This method, however, has the drawback that the path may end at at a local minimum, thus it can fail under certain circumstances (e.g. in the presence of concave obstacles).

The present approach does not use potentials to determine the direction of a motion. Instead, a potential is used to determine the cost of an edge in the graph. A similar approach was taken in [MJ97] and in [SFG⁺04].

The cost of an edge E is set to be proportional to the length of the edge. Additionally, the cost is multiplied by a proximity factor, which is high when the edge is close to an obstacle. When the edge is far away from any obstacle, the proximity factor approaches 1.

So if the edge E_{AB} connects the nodes A and B, then its cost is determined by equation 2.6. The function $dist_p$ in this equation denotes the euclidean distance between the location of a node, while $dist_\phi$ provides the difference of the orientations of the two nodes. The function Pot is the potential that maps the location of a node to the interval $[1, \infty]$.

$$cost(E_{AB}) = \frac{1}{2}(Pot(A) + Pot(B)) \cdot (dist_p(A,B) + \alpha \cdot dist_\phi(A,B)) \tag{2.6}$$

The term $\alpha \cdot dist_\phi(A,B)$ on the right side of equation 2.6 accounts for the costs of rotational movements. Because the graph contains no edges that would connect nodes with different positions *and* different orientations, either $dist_p$ or $dist_\phi$ is zero. The factor α is adjustable and determines how much rotational movements are penalized compared to translational movements. Experiments were made to determine the influence of α on the planning results. The experiments and their evaluation will be presented in section 2.5.2.

Many potential functions are possible, although they should be continuous and increase monotonically as the distance to an obstacle decreases. Figure 2.11 shows how different planning results, which are the outcome from the use of different potential functions. Experiments with various potential functions and their results will be presented in more detail in section 2.5.1.

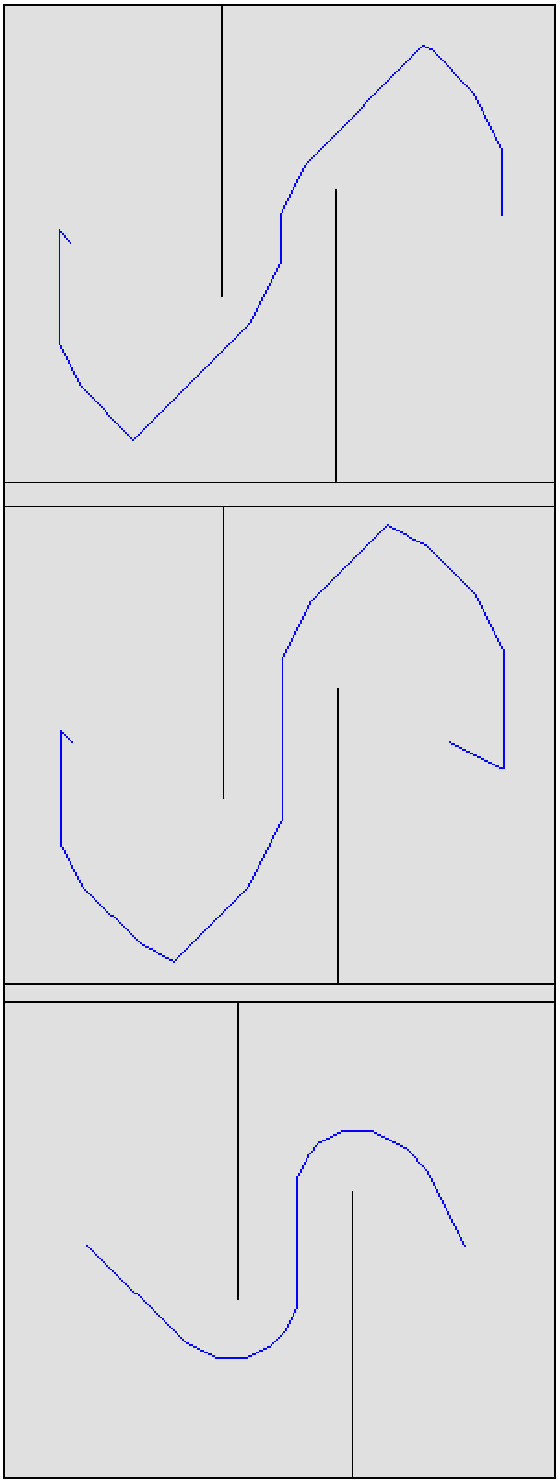

Figure 2.11.: Planning results with different potential functions

2.4. Computation of expansions

After the principles of the planning algorithm have been laid out, this secion will highlight some aspects of implementing the algorithm.

The planning algorithm was said to work on polygonal obstacles. However, computing the expansion of polygons proves to be a little tricky. The problems are described in [SW02]. The same paper proposes a solution by reducing the polygons to only their outline and using an expansion of their bordering lines. This expansion can be computed easily (see below).

The expansions of lines are polygons of themselves or they form curved areas, which must be approximated by polygons. Some expansions of lines may overlap as a result, but this is not a problem at all. Furthermore, the inside of a sizeable obstacle now can be represented as free space, but this does not matter neither, since the free space is all bounded by the expanded bordering lines. Therefore, this inside space cannot be reached from the outside. Thus the map contains only lines expanded to polygons and it can be used by the planner without further modifications.

A simple approach to compute the expansion of a line would be to lay the outline of the robot and moving the robot's outline along the line in a manner that it always touches the line, but never intersects with it. The reduction point also moves together with the robot's outline. Its movement will mark a line that is the border of the basic line's expansion (see figure 2.12). This procedure must be repeated for every orientation of the robot that is to be used by the planner.

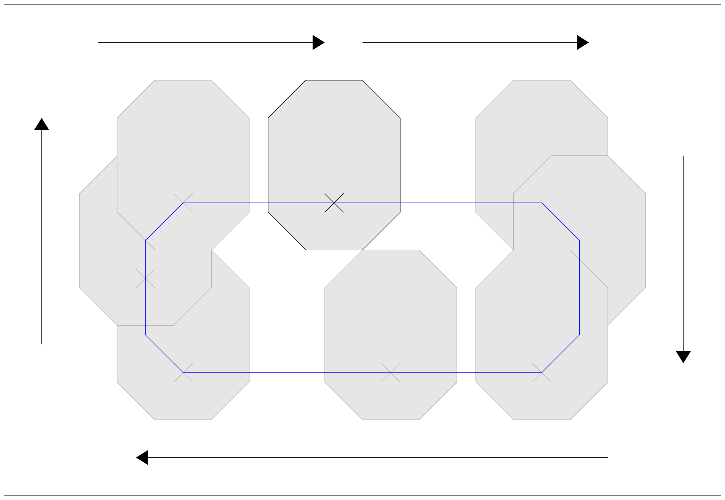

Figure 2.12.: Computing the expansion by moving the robot around the obstacle

This approach is also computationally complex, since it involves a lot of points being computed as the reduction point moves along the basic line. The same result can be achieved using a less complex method. To understand how this method works, the expansion of a single point will be explained first. This means that the following question must be answered: Which areas of the map are blocked, i.e. may not be entered by the reduction point of the robot, if the map is entirely free, apart from a single very small sized obstacle e.g. a vertical bar?

As the robot approaches the obstacle, a collision would occur if the distance between the reduction point and the obstacle is the same as the distance between the reduction point and the outline of the robot. From the obstacle's point of view the reduction point must not come closer to it, than the distance from the reduction point to the robot's outline.

Because the robot is not rotationally symmetric, there are different distances from the reduction point to the outline. Here the one must be considered, which lies on the line between the reduction point and the obstacle.

Therefore, the distance between the obstacle and the expansion at any given angle is the same size as the distance between the reduction point and the robot's outline but at the exact *opposite* angle. As a result the shape of the expansion of a point obstacle is the point-reflected robot's outline, reflected on the reduction point (see figure 2.13).

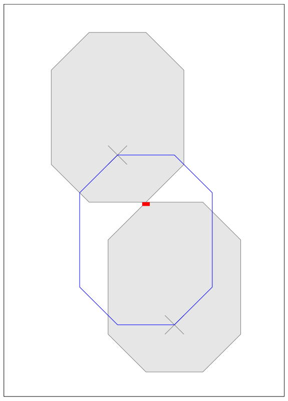

Figure 2.13.: The expansion of a point obstacle is the point reflected robot outline

Now that the expansion of a point has been determined, this approach can be extended to lines as well. All that is needed is to take the reflected shape of the robot and move it along the obstacle line. All points marked by the shape are blocked and part of the expansion of the line (see figure 2.14).

The computation of this area can be done very simple. It can be seen as the convex hull of the defining points of the robot's shape put at the start and the end of the line (see figure 2.15). There are many algorithms to find the convex hull of a graph. The one used in this work was the algorithm called “Jarvis march” [Jar73] (sometimes also named “Gift wrapping”). It was used because of its simplicity, altough there are algorithms performing better. Yet in the presence of a small number of points, as it is in this case, the algorithm's performance is only of small significance.

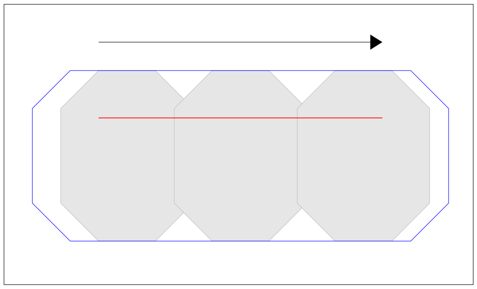

Figure 2.14.: Moving the point expansion along a line

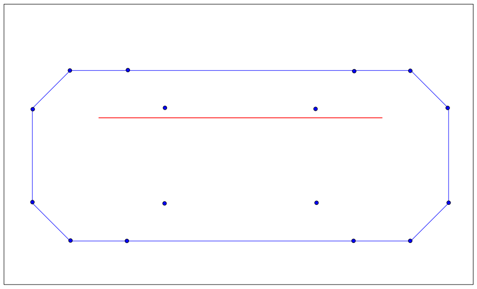

Figure 2.15.: The expansion as a convex hull
The nodes are the points defining the robot's outline

2.5. Experiments

It has been stated in section 2.3.2 that it is possible to exert influence on the paths generated by the graph search by modifying the costs of the edges in the configuration space graph. The costs are determined by equation 2.6, which has two adjustable components: The potential function Pot and the turn cost factor α.

Experiments have been conducted to determine the impact of these variables on various properties of the generated paths. To this end, seven different planning tasks were devised, which are shown in appendix A. The planning algorithm was used to generate paths for each of the planning tasks. This has been repeated with different settings for the potential function and the turn cost factor.

Every path generated was analyzed for several measures which are significant for the judgment of the quality of the path. These measures are:

- The overall length of the path l
- The number of rotational movements (changes in orientation) along the path n_r
- The sum of the amount of all rotational movements $\widehat{\phi}$
- The average distance from the path to the closest obstacle $\bar{d}$
- The average distance at a turning point to the closest obstacle $\bar{d_t}$
- The closest distance from the path to an obstacle d_{min}
- The closest distance from a turning point to an obstacle $d_{t,min}$

Additionally the planning duration has been recorded. While this is not a property of the path, it also should be considered, because it might be undesirable to gain a small quality improvement in exchange for much longer planning duration.

The experimental values of the measures are normalized to make them comparable, thus the values presented here are not absolute. Furthermore, they are averaged over the different planning tasks. The index base is set to 100. So if a measure has the value 150 in a certain setting, this means this measure was in average 1.5 times greater than in the reference setting.

It should be noted that the safety of the path in general does not rely on high values for $d_{t,min}$ or other distances. If the localization assumption of the robot is correct and the environment map is exact, the generated path will be collision free. This is asserted by the occupancy grid method. The grid has no nodes at blocked positions, therefore no paths through blocked positions can occur.

As a result, basic safety is already provided by the underlying algorithm. Yet the additional distance to obstacles gained by modifying α or by choosing the right potential function can provide some additional safety, just in any case. Since measurements, either of the own position or of the obstacles, are never exact, this additional safety is useful.

2.5.1. Testing different potential functions

In the potential field method, the repellent forces from different obstacles are usually summed up to determine the resulting force at any location. However, with this approach it is also possible to use only the proximity factor of the closest obstacle.

To be more precise, assume that $rep(d_i)$ is the magnitude of the repellent force at a distance d_i to an obstacle i. Furthermore, $Pos(i)$ gives the position of obstacle i and $Pot(p)$ is the combined magnitude of all obstacles at position p. Then whereas in typical potential functions equation 2.7 is valid, in this planing algorithm a function according to equation 2.8 could also be used. Since the potential function is required to be monotonic, choosing the closest obstacle is identical to choosing the obstacle with the highest potential at a given point. This section will present the different planning outcomes that result from using either of these possibilities.

$$Pot_s(p) = \sum_i rep(\|p - Pos(i)\|) \tag{2.7}$$

$$Pot_m(p) = \max_i(rep(\|p - Pos(i)\|)) \tag{2.8}$$

In the following, the function $rep(d)$, which determines the magnitude of the repellent force in dependence of the distance to an obstacle, will be called repulsion function, whereas the function $Pot(p)$, which determines the combinded magnitude of the repellent forces of *all* obstacles will be called potential function.

In the experiment, three different repulsion functions were used. For each repulsion function, two different potential functions were designed. The first potential function considers only the cost of the closest obstacle (according to equation 2.8), and the second one sums up costs from all obstacles (according to equation 2.7). This yields a total of six different potential functions. For each of them, one experimental run has been carried out.

Any repulsion function should be monotonic. It maps a distance d_i (from a certain point to obstacle i) to a positive value. At big distances it should have a value close to 0, and at close distances it should have very high values. All field functions tested fulfill these requirements.

In the experiments the distance unit was one millimeter. Every run has been carried out with $\alpha = 1$.

The repulsion function given in equation 2.9 increases linearly as the distance between one point and an obstacle decreases. u and a are parameters to the function. The parameter u sets the value of the function at the distance 0, and a sets how fast the function decreases. In the experiment, $u = 10000$ and $a = 10$ was chosen. To be exact, this function is only used when $a \cdot d < u$, for other d the function has a 0 value. Otherwise negative values would occur.

The repulsion function in equation 2.10 has a reciprocal relationship between the distance and the resulting value. The parmeter u sets the distance at which the function assumes the value 1, and the parameter a determines how fast the function grows when the distance decreases. This repulsion function has been used in the experiment with two different sets of parameters. In the first instance $a = 1$, $u = 1000$ and in the second $a = 0.25$, $u = 1000$ was chosen.

Function	$\max_i(\sqrt[4]{\frac{1}{d_i}})$	$\sum_i(\sqrt[4]{\frac{1}{d_i}})$	$\max_i(u-d_i)$	$\sum_i(u-d_i)$	$\max_i(\frac{1}{d_i})$	$\sum_i(\frac{1}{d_i})$
l	100	96.9	121	121.5	133.5	114.5
n_r	100	71.2	85.9	86.2	252.4	166.9
$\widehat{\phi}$	100	73.1	166.7	164.5	362.6	155.5
$\overline{d}$	100	95.2	123.6	117.6	126.9	108.3
$\overline{d_t}$	100	88.6	143.7	129.3	131.2	104.9
d_{min}	100	94.5	94.2	81.4	108.3	106
$d_{t,min}$	100	94.2	93.6	85.9	96.2	104.4
plan. duration	100	112.6	123.6	118.4	145.9	126.1

Table 2.1.: Experimental results for planing with different potential functions

$$rep(d_i) = u - a \cdot d_i \tag{2.9}$$

$$rep(d_i) = \left(\frac{u}{d_i}\right)^a \tag{2.10}$$

The results of the experiments are given in table 2.1 and in figure 2.16. The values of the measures are normalized, so they equal 100 for the first potential function. Attention should be paid to the diagram, where in the instance of the second last function two measures are off the scale. Their actual values can be read from the tabular view.

The results show that the first two potential functions (in the same order as they occur in the diagram) yield much smoother paths, with less rotation and less overall length than the other functions. They also generate paths which have on the average the highest shortest distance at a rotation $d_{t,min}$, with the exception of function six, which has a 4% higher value for $d_{t,min}$. Thus function one and two both seem to be good choices for the navigational system as they do not differ much. Furthermore, with these two functions the planning duration is the shortest, although with the other functions it was no more than 46% higher. Function one has a slightly higher $d_{t,min}$ and the shortest planning time, while function two produces shorter paths with less rotational movements.

2.5.2. Testing different turn costs

In this setting the impact of changing the turn cost factor α was analyzed. The planning algorithm was executed with different values for α. The resulting values of the path measures have been normalized so they equal 100 for planning with $\alpha = 0$ Setting α to 0 means that the planning algorithm considers changes of orientation as free of costs. These values are given in table 2.2 and figure 2.17 as a diagram. The potential function used was 2.10, with $u = 1000$ and $a = 0.25$, and only the closest obstacle counted (this is the first function in diagram 2.16).

The results show that the turn factor may help to generate path with less rotational movements. Furthermore the rotational movements are performed with more distance to obstacles (measure

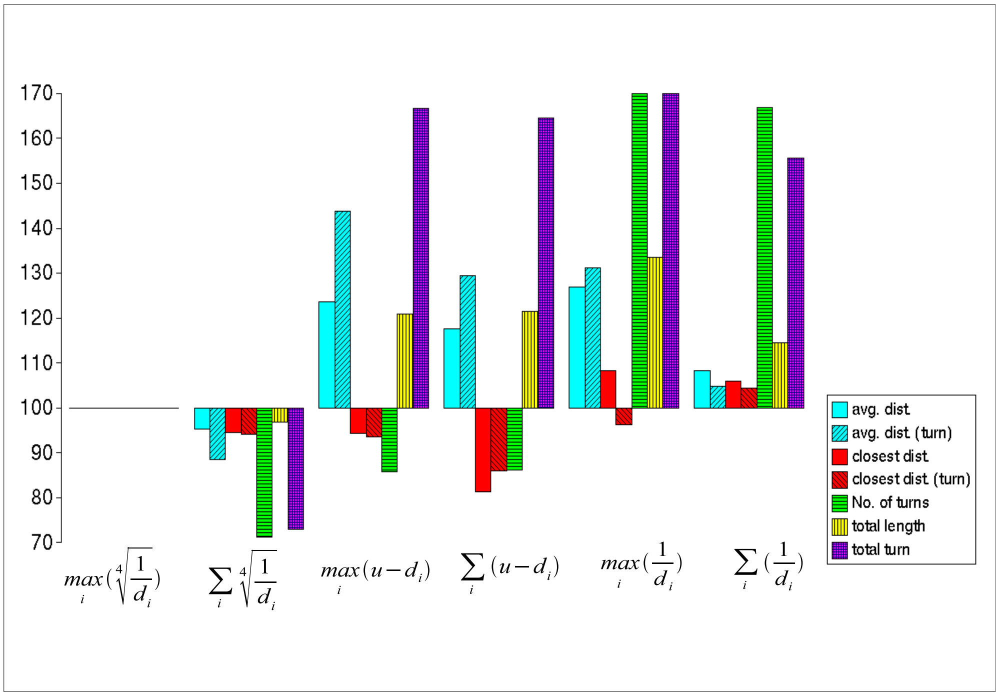

Figure 2.16.: Experimental results for planing with different potential functions

$\overline{d_t}$), thus making the path more safe. Especially the rotation with the closest distance to an obstacle (measure $d_{t,min}$) is performed at a about 20% larger distance, when α has a value of 4 to 8. This is a very good sign, because it means that the most critical situation is alleviated.

However, increasing α also seems to result in a path which may lead a little closer to an obstacle during translational movements (measure d_{min}). Thus adjusting this parameter allows a trade off between rotational movements performed further away from obstacles and translational ones. With higher α, the overall path length also increases slightly.

It is not quite clear why the sum of all rotational movements $\widehat{\phi}$ has a minimum at $\alpha = 2$ and then starts to grow for higher αs, especially since the total number of turning points decreases continuously. Perhaps this experimental result is caused by the some characteristic of the planning tasks used in the experiment and this feature would not occur if more planning tasks would be evaluated.

The value of α has a heavy impact on planning duration, as can be seen in table 2.2. This is probably caused by the higher edge costs, which make the cost estimation function[3] in the A^*-search less informed. Thus more nodes are visited during the search before the path is found. This should be kept in mind before choosing excessively high values for α.

As a result from this experiment, a value of 1.5 for α was used for the navigation system of the robot.

---

[3] The cost estimation function used for the A^*-algorithm is the simple euclidean distance from the node to the final node, plus the minimum rotational cost needed

α	0	0.5	1	2	4	8
l	100	100.4	99.6	100.2	100.9	104.5
n_r	100	82.3	69	62.2	56.4	46.1
$\underline{\widehat{\phi}}$	100	80.7	68.3	61.2	63.9	74.7
$\underline{d}$	100	99.6	99.1	98.8	97.4	99.8
d_t	100	107.3	113.7	108.4	109.4	120
d_{min}	100	98.5	98.9	98.4	94.8	94.8
$d_{t,min}$	100	104.1	112	114	117.8	121.7
planning duration	100	215	305.4	498.7	735.3	951.4

Table 2.2.: Experimental results for planing with different αs

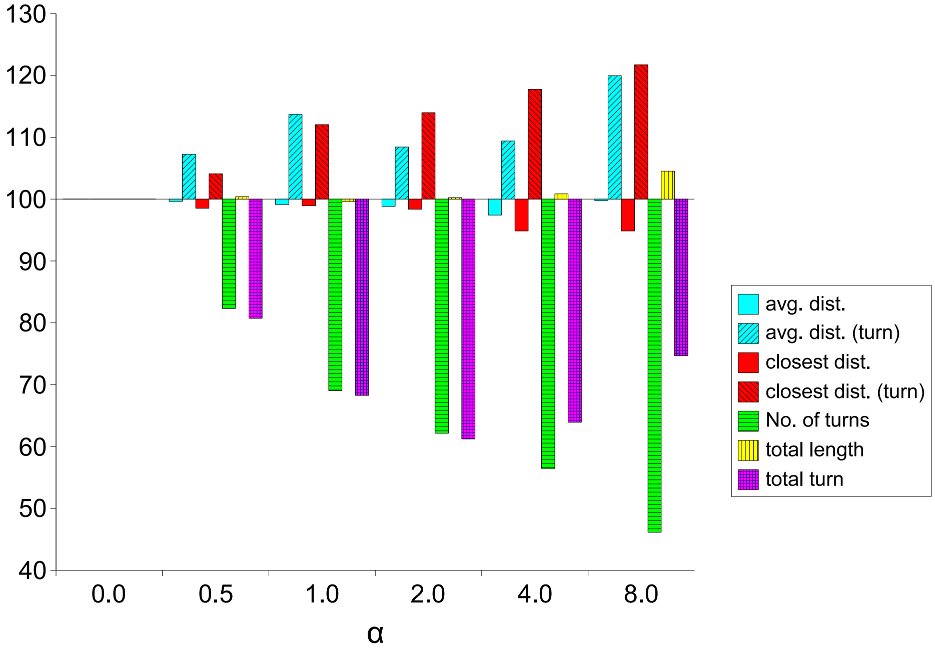

Figure 2.17.: Experimental results for planing with different αs

2.6. Summary

In this chapter the path planning part of the navigation algorithm was introduced. There were three central topics. At first, there was a description how obstacles have to be expanded if the robot is shrunk to a point in the configuration space. Then there was a presentation of the basic navigation algorithm used, namely the visibility graph method. At last, the algorithm for navigating through a narrow passage was described. It is based on a search in a space grid with a potential field. The field causes the search to prefer locations further away from obstacles. Both planning methods are combined to generate well suited paths quickly. Another short section showed how the expansion of obstacles is computed.

After all principles used for path planning have been laid out, there was a presentation of experiments which have been conducted in order to decide which parameter settings suit the problem best. Offline runs of the planning algorithm show that the generated paths are sensible and the planning duration is within acceptable bounds. Thus the planning part of the navigational algorithm seems to provide a good base for the further tasks remaining to be done.

The planning algorithms presented so far works on static data. The next step to do is to acquire data from sensors about obstacle positions and to integrate this data into the planning algorithm. Only with this live data a reliable navigation algorithm can be achieved when moveable, transient obstacles or map inaccuracies are present.

Integrating sensory data 3

The path planning algorithm presented in the previous chapter works on static data. As already stated before, the robot cannot rely only on static environment data. The reason is obvious. The office environment is dynamic, and changes in the environment, like a moving human or a moved object, must be reflected in the robot's representation immediately. Therefore the robot must use sensors to gather information from its environment and update its representation accordingly.

However, as will be pointed out in more detail in section 3.3, a static map of immobile objects (like walls) is not useless at all. Therefore the planning system uses a mixture of static map data and sensory data for the environment representation. The static map must be once created by an operator an can be stored in a file for reuse. It contains only permanent obstacles, i.e. obstacles which will never move or dissapear. A door is considered to be a moveable obstacle, which can block the doorway. Therefore doors are not entered into the static map, only open doorways are inserted instead. The sensors will sense the position of the door and the dynamic environment representation will be updated accordingly.

The navigation system can make use of any sensory data, as long as there is a transformation which maps the sensory data into a map of occupied and free space. In this work, laser range finders and a pair of cameras were used as sensors. The main parameters for these sensors have been presented in section 1.3.

This chapter explains the details that have to be considered when integrating sensory data into the environment representation, with the main focus on laser range finder data.

3.1. Reacting to a changing environment

The basic approach how to handle a dynamic environment is the following: After a path to the destination has been planned, the execution of this path starts. During the execution, the data from sensors updates the environment representation. At the same time, the navigational system periodically checks whether the planned path intersects with an obstacle. Of course the path initially did not intersect with any obstacles. If it does at a later moment, this is caused by some new or moved obstacle, which came into the planned path.

If such a situation is detected, the path execution stops immediately, and a new path is generated from the current position to the destination, using the new environment data. Then the execution of the new path begins, which is known not to intersect with any obstacles known at the time of the planning. Thus the new path circumvents the new obstacle. Such an approach was pioneered by [Zel92].

There are more sophisticated approaches to the problem of replanning in the presence of a dynamic environment, like the D^*-algorithm [Ste94, Ste95]. Their main advantages are less computational costs of replanning. This is accomplished by not performing the entire planning phase again. Instead, only small changes to the last planning result are made.

However, in the scenario evaluated by this work, the replanning phase is often hardly noticeable, and only in awkward difficult situations the replanning phase takes longer than a few seconds. This is in large part caused by the fast performing two stage approach. So there is not much need to implement the sophisticated replanning methods, when the simple approach of starting all over yields satisfactional results.

3.2. Integrating data from laser range finders

The main advantages of laser range finders are that they provide reliably accurate measurements of the distances to obstacles surrounding the scanner. The measurements are done quickly and can be repeated several times per second. They produce a comparatively small amount of data, so not much filtering or postprocessing is needed. The data from one scan simply consists of 361 distance values.

All these properties would make the laser range finder an ideal sensor for obstacle detection, if not for one decisive drawback. It performs the scan only in one plane, thus it inherently cannot detect hanging objects and it has only a bad perception of certain objects like chairs or tables, as it senses only the legs, but not its far more sizeable surface.

Although there are tiltable laser range finders which scan different planes subsequentially (as used in [BWW03]), such a scanner is not available on the robot. Besides, this type of scan takes much longer and generates much more data, which makes complex postprocessing necessary.

Still, the laser range finder can detect a lot of obstacles reliably, so it would be a waste not to use this valuable data. The intention is to detect the remaining obstacles by the stereo cameras (see chapter 4).

One scan can be seen as a set of points surrounding the scanner. For each point, the distance to the scanner and the angle relative to the scanner are known. So the scan points are given in polar coordinates relative to the scanner. Because the position of the robot in the world coordinate system can be retrieved from the localization system, and the position of the scanner relative to the robot is fixed and known, the coordinates of the scan points can be easily transformed to world coordinates. This is necessary, because the map uses the world coordinate system.

It is worth to remind that the robot's map is based on lines (see section 2.4). Although it is technically possible to insert each point individually into the map as a zero sized line, the computational cost of the planning algorithm depends on the number of obstacles. Thus it is favorable to group a set of points together and insert them as one line. In fact, in a typical scan there are a lot of points which form a line (like walls, doors, etc.). Thus the next transformation step from the raw data is to perform a line extraction on the points from the scan.

There are a lot of different line extraction algorithms. A comparative overview is given in [NMTS05]. One of the most popular line extraction algorithms is called *split and merge*. This algorithm originates from the field of computer vision [PH74], and was then adopted to mobile robotics [BA04]. The algorithm used for line extraction in this work is a variation of the *split and merge* algorithm. Algorithm 1 outlines how it works.

First the set of points is approximated by a line stretching from the first point to the last. A line error function calculates the total approximation error of the set of points being approximated by the line. If the error is below the threshold σ_{max}, the line is accepted as the approximation of the set. In the opposite case, the set is split into two subsets at the point that has the highest distance to the approximating line. Then the split function is called recursively on both subsets. The union of the results of both recursive calls is then returned as the result of the line extraction.

Algorithm 1: The *split* line extraction algorithm

```
Constant: $\sigma_{max}$: Maximum line error
Function: split($\mathcal{P}$)
Input: $\mathcal{P}$ List of scanned points ordered by angle to laser range finder
Output: $\mathcal{L}$: A set of lines

begin
    $start \longleftarrow 0$
    $end \longleftarrow \mathcal{P}.size()$
    $line \longleftarrow new\ Line(\mathcal{P}[start], \mathcal{P}[end])$
    if $line.error() < \sigma_{max}$ then
        $\mathcal{L} \longleftarrow \{line\}$
    else
        $P_{split} \longleftarrow find\_most\_distant\_point(line, \mathcal{P})$
        $i_{split} \longleftarrow \mathcal{P}.indexof(P_{split})$
        $\mathcal{P}_{left} \longleftarrow \mathcal{P}.sublist(start, i_{split})$
        $\mathcal{P}_{right} \longleftarrow \mathcal{P}.sublist(i_{split}, end)$
        $\mathcal{L} \longleftarrow split(\mathcal{P}_{left}) \cup split(\mathcal{P}_{right})$
    endif
    return $\mathcal{L}$
end
```

σ_{max} is an adjustable parameter. Set to high values, the algorithm will produce fewer and longer lines, but the lines may deviate a lot from the original scan points. If the parameter is set to a low value, the generated lines will be close to the original scan points, but they tend to be short and many.

The extracted lines, given in world coordinates, can be inserted into the map without further transformation. These lines are interpreted as new obstacles.

3.3. Obstacle memory

It is not enough only to add obstacles to the map. At some point they must be removed again. Otherwise, more and more obstacles would be added to the map, thus slowing down the planner.

Furthermore these obstacles would block positions which in fact are freed meanwhile. This would happen when a moving obstacle is observed. All of its old positions would remain as obstacles in the map. Finally, without clearing old obstacles, a memory overflow could occur.

Even in the case of a static environment and the robot not moving, more and more obstacles could clutter up the map, if there was no provision to remove obstacles after some time. The reason is, that there is no obstacle matching included in the mapping algorithm. Because of sensor noise, the same obstacle might have slightly different appearances in subsequent scans. Without scan matching, which could try to match the scanned obstacle with an obstacle from previous scans, each differing scan is treated as a new obstacle. Thus one single obstacle in the environment could produce a lot of obstacles in the representation.

As the approach presented in this works performs no kind of object recognition, there is no way for the robot to distinguish temporary, movable obstacles (like humans or doors) from permanent ones (like walls). This requires all sensed obstacles to be treated as temporary. Only the obstacles entered into the static map are treated as permanent.

Several ways of removing obstacles are possible. One approach could be to remove all obstacles, which according to the most recent scan cannot exist anymore, because the scan sees an object behind the obstacle. Yet this approach doest not guarantee that all old obstacles are eventually removed. An obstacle that was once detected, but cannot be seen from another position, could remain in the map forever.

One method which guarantees that an obstacle is eventually cleared is to tag any new obstacle with a timestamp, and remove any obstacle from the map which has a timestamp which is older than a certain threshold.

This raises the question, which threshold to use. One could be tempted to use a very small threshold, thus removing obstacles immediately upon the next scan. But during tests of the navigational system, certain situations emerged, which suggested that it might be beneficial to the system behavior if not only immediate sensor data was used for planning, but obstacles from earlier scans were also considered.

The first such situation is, when a certain pattern in the scan repeats itself over a short time. This might be caused by a periodically moving object or by sensor noise, which makes the same obstacle appear at slightly different positions every time. In this situation it is sensible, if the planning algorithm not only “knows” the most recent scan, but is also aware of obstacles identified by previous scans. To achieve this, a threshold of a few seconds could be used. In effect, the entire area where the mentioned movement takes place appears as blocked to the planner.

While this situation does not occur often and might be glossed over, in another situation the correct threshold is much more important. This is the situation when an obstacle that was visible in a certain scan is not visible in subsequent scans, although the obstacle has not moved. This might be caused by two reasons.

Firstly, as stated in section 1.3, there are two “blind areas”, where no scanning is possible due to the characteristic of the scanners and their positions on the robot. If an medium sized obstacle is inside the scan area, thus visible in the scan, and the robot performs a movement, the obstacle might end up being inside the blind area and not visible any longer.

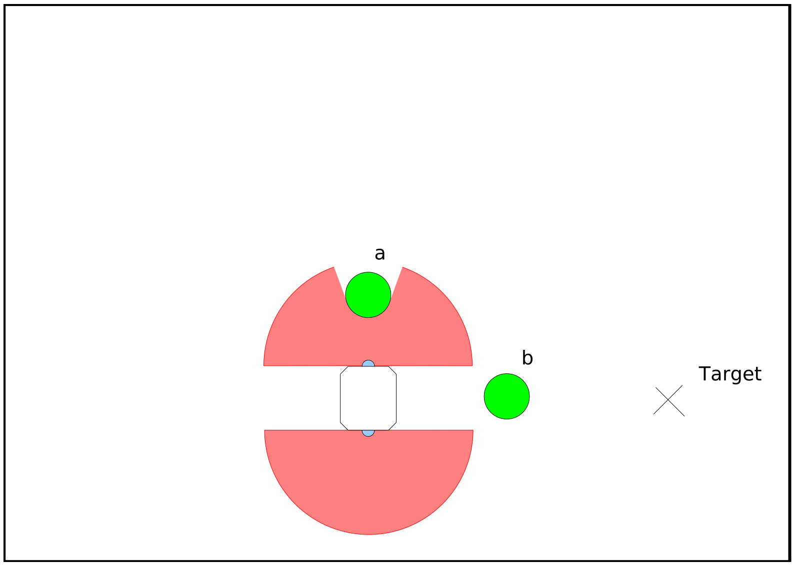

Figure 3.1.: Robot "trapped" between two obstacles

Secondly, in the case of two (or more) visible obstacles, one obstacle might "disappear" because it is hidden behind the other obstacle after the robot performed some translational movement.

Both cases can lead to oscillations in behavior, when only immediate scan results are considered. The robot might get caught trapped between two obstacles; figure 3.1 illustrates this situation. The robot does not see obstacle *b*, therefore it plans a path trough the position of *b*. If it turns to the right to execute the path, *b* becomes visible, but *a* dissappears. The planner detects the potential collision with *b*, thus a new path is planned. Because now *a* is not visible anymore, a path through *a*'s position is planned. The robot turns back left. Then the same situation as in the beginning is reached, and the cycle can repeat again.

For the second case, consider figure 3.2. There are two doors to a room, but both of them are closed. The robot has the task to enter the room. A wall between both doors blocks the view, such that both doors cannot be observed at the same time. In this case, the robot would endlessly move from one door to the other, always anticipating one of the doors is open, only to detect it is closed when it stands in front of it. Yet at this position it cannot see the other door, thus the other one is assumed to be open. When the robot reaches it, the cycle starts over again.

This problem can be addressed to a certain extent by setting a higher threshold for removal of obstacles which are not visible from the current position. The planning system checks if a previously scanned obstacle is either within the blind spot or behind some other obstacle. If this is the case, it will be removed only when a longer timespan has passed since last sensing the obstacle (this timespan is presently set to 60 seconds).

This kind of remembering of obstacles which are not visible directly could be called short term memory. While this technique breaks most of the usual oscillation situations, there are some

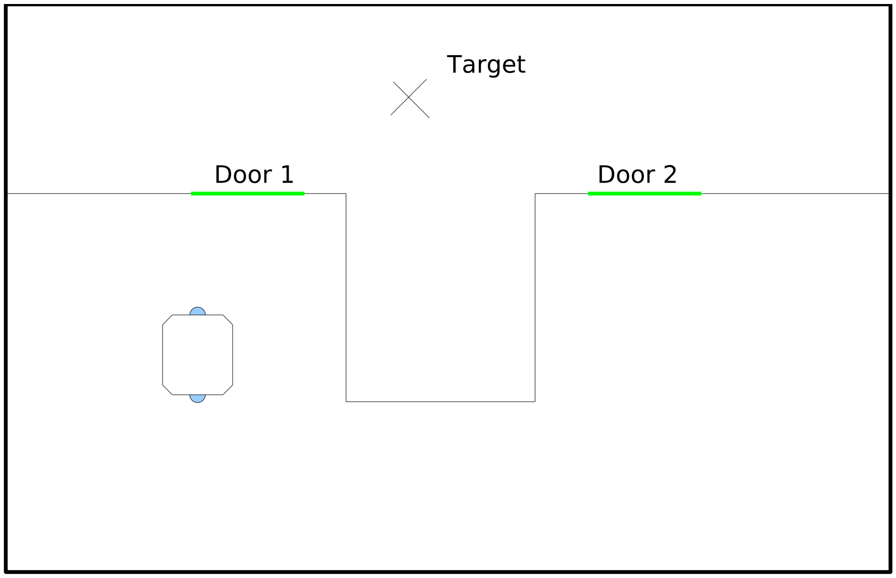

Figure 3.2.: Robot oscillates between two doors

limitations. Consider the before mentioned examples of the room with two doors. It is conceivable to create a situation where the distance between the two doors is long enough for the robot to "forget" the closed state of the first door before it arrives at the second. Of course, the threshold to forget an obstacle could be adjusted to fit the current environment, so that this situation does not occur. But the threshold cannot be increased indefinitely, due to the reasons laid out at the beginning of this section.

To handle such situations properly, other planning approaches must be used. Planners that make use of learning or behavioral techniques could be used here. Higher level planners with different types of memory and world knowledge could use object recognition to detect obstacles that are crucial for path planning (like doors or elevators). In combination with a set of behaviors, either pre-programmed or learned, the robot could start actions like opening doors or moving away obstacles to reach its goals. These techniques are however still being research and at an early stage of development. They are out of scope of this diploma thesis, but could be used in further work.

With the problem of oscillations in mind, it becomes obvious why a static map of permanent obstacles is a necessity. Consider an environment representation that does contain only obstacles sensed by the sensors and does not contain any permanent data. Because previously sensed obstacles that cannot be sensed anymore have to be removed eventually from the map, similar situations like the before mentioned ones could occur. Such a situation is shown in figure 3.3. The robot could move between rooms A and B, trying in vain to reach a third room C. Each time the robot enters room A it would "forget" that there is a wall separating B from C, because it can see only the walls colored red in the figure. Upon detection of the wall between A and C it would go back to B, because it supposes that the shortest path to C is via B. But when it arrives at B, the same problem occurs and the robot moves back to A.

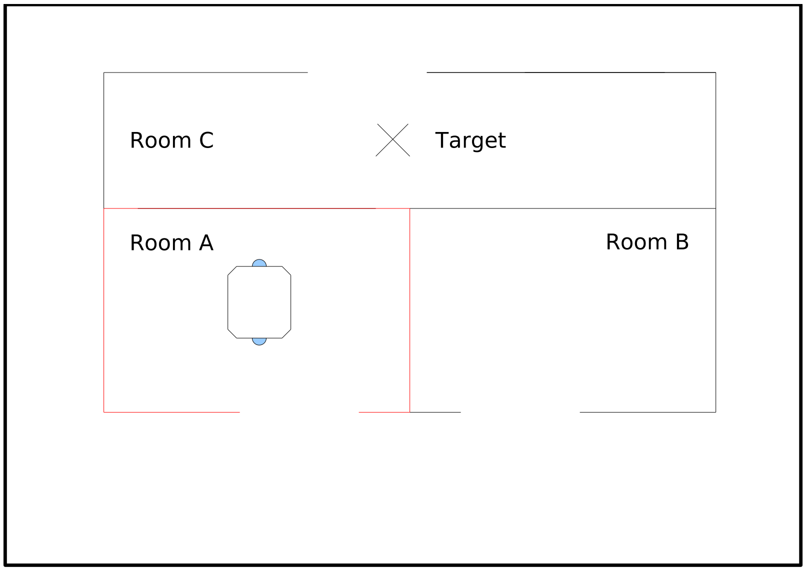

Figure 3.3.: Robot oscillates between two rooms, because static map data is missing

With static map data, this type of oscillation occurs rarely. In contrast to the short term memory, which contains recently sensed obstacles, this static information could be seen as a simple long term memory.

This problem could also be solved by using a good mapping algorithm which remembers the obstacles that the robot scanned over a long time. However, a good mapping algorithm would require some kind of obstacle classification, because moveable obstacles need to be removed quickly, and permanent obstacles should remain in the map over a long time. Because the discrimination between temporary and permanent obstacles is impossible without some kind of object recognition, which this approach lacks, such a solution is not appropriate for this work.

4

Using stereo cameras

As has been already stated in section 3.2, the laser range finders are inherently incapable of detecting certain obstacles. The question arises: Can another modality compensate for this weakness? If so, both modalities combined could produce a more reliable representation of the environment. As the robot possesses multiple cameras, including a pair of stereo cameras, these might be used as the second modality. Stereo computer vision is nowadays a well established field of research with many working applications. So the question to be examined for this part of this thesis is: How can stereo vision be applied to gain additional information about the environment to make the navigation system more robust and safe? To answer this question, the process of generating depth information from stereo images must be understood.

This chapter will describe the process of computer stereo vision. At first, there will be an introduction into stereo vision in humans and animals, which acts as inspiration for computer stereo vision. Then the camera model is presented, which is a formalizing abstraction of the image taking process. This model allows to make geometrical computations with the scene shown in the picture. The camera model is closely bound up with the process of calibration, which estimates unknown parameters of the camera model. This process will be explained in one section of this chapter. After that, the geometrical coherence of a scene which is photographed by two cameras will be explained. This allows to formulate equations, from which the depth of a scene feature can be calculated. These calculations are based on *disparity*, the difference in position of the same feature in the left and the right picture. The subsequent section will cover how to detect disparities computationally. Finally, it will be shown why line segments in the picture are chosen to find disparities, and how they can be extracted from images.

4.1. Natural stereo vision

The vision system in humans and many animals allows for a three dimensional perception of the environment. The bearer of such a system sees the world not in a plane, but he can also percept the approximate distance to an object. This capability is a considerable advantage for predators, since they can attack their prey much more accurately, thus more successfully, if they know the exact position in space of their prey relative to themselves. This advantage has led to the gradual development of three dimensional vision in various species during the millions of years of evolution, until the highly sophisticated vision systems of humans and some other animals were reached.

Humans, and many animals, are capable of three dimensional vision in spite of the fact that all visual information is gathered from a two dimensional retina. This means, that the information about the depth of an object must be somehow reconstructed by the visual system after capturing the scene in the eye. A good reference for current knowlegde about the workings of natural vision systems is [PCGK95].

There are varying degrees in the capability of depth perception in different species. Less developed animals, like insects, reptiles or fish, have little or no depth perception at all. Higher developed animals have a certain degree of depth perception, but only a few species like cats, some birds and primates seem to have visual capabilities similar to humans [HS01, PG75]. The ability of three dimensional vision seems to have developed independently in birds and primates.

There are many ways how depth information can be acquired from flat images, including depth perception by relative size of objects, overlay and shadowing. Animals which have a vision system that uses this information have some depth perception. But what is unique to species with best depth perception capabilities is the ability to generate depth information from the differing images in the eyes. Because of the different positions of the eyes, light rays coming from the same object do not hit the retinas in the same point. The difference is directly related to the distance from the eyes to the object. The process of generating depth information from two images taken at differing positions and orientations is called stereopsis, binocular vision or stereo vision. Its geometry will be explained more detailed in section 4.4.

Binocular vision in nature requires eyes with parallel vision axes and a visual cortex able to process the information. It seems that the development of binocular vision contributed heavily to the growth of the visual cortex and the brain in general in primates [Bar98, Bar04]. This fact, the late stage in evolution at which binocular vision was developed and the rather small number of species who developed it at all point to the complexity of this capability. Of course some species, e.g. herbivorous animals, have not much use for stereo vision. For humans however it seems to have been one key ability necessary for the development of civilization because tool making requires a good depth perception.

If the principles of the natural vision system are understood, the natural process of retrieving depth information from planar images can be emulated by technology. Although stereopsis is not the only means how the human vision system generates depth information, most computer vision systems use it exclusively. This is true for all the works about robotic vision stated in section 1.4.2. The reason is that the process of stereo vision is best understood and easiest to implement compared to the other sources of depth information. Furthermore, this method provides quantitative results while others often can only provide qualitative results (e.g. object A is behind object B).

To state an example for another possible information source consider that humans know that objects usually do not shrink or grow when they move. So when they see a small image of an object, which is known to be otherwise much bigger, they will assume it is far away. Such an approach however would be much more difficult to implement. It would require knowledge or assumptions about the world and object recognition. Then different hypothesises about the position of objects could be tested whether they are compatible with the world knowledge or assumptions. In contrast, the principle behind stereo vision is only basic euclidean geometry. For these reasons and because this approach has proved workable in many systems, preference was given in this work for stereo vision above other visual methods.

4.2. The camera model

A camera is a device which somehow controls the flow of light rays from a scene and guides the light rays onto a light sensitive two dimensional medium, which can measure or record the flux. Eyes would also be cameras by this definition, but in the following sections only artificial cameras will be considered. In such cameras the flux measuring medium is usually flat, and is either a light sensitive chemical substance (in traditional analog cameras) or a CCD[1]. To enhance its image capturing capabilities, cameras also have lenses and mirrors, which refract and reflect the light rays.

A mathematical model for a camera which contains all the effects of refraction, diffraction and reflection in multiple lenses would be very complex and inconvenient to handle. Fortunately, for most computer vision applications a simpler model is accurate enough. This is the so called pinhole camera model, which can be extended by an additional model of radial distortion occurring in a lens. This model will be explained in the following.

This section will describe the camera model and the projection geometry. These two items are indispensable for a good understanding of how an image is formed and what relationship exists between the coordinates of the photographed scene and the coordinates of the image of the scene. The notions in this chapter are chosen to be similar to those in the literature. For further reading on this topic, consult [MSKS04] or [Sch05].

4.2.1. The pinhole model

The pinhole camera model consists of a point (the optical center) and an image plane. The image plane is located behind the optical center in respect to the scene which is being photographed. The photographic medium is located on the image plane. The camera is sealed, so that no light rays, apart from those which pass through the optical center, can fall on the photographic medium. The distance between the optical center and the image plane is the focal length f. The line which is perpendicular to the image plane and passes through the optical center is called optical axis. The origin of the image coordinate system is the intersection of the optical axis and the image plane. This model is shown in figure 4.1.

Because we assume the light rays travel linear, simple euclidean geometry provides the properties of the perspective transformation within this model. Equations 4.1 and 4.2 show the relation between a point $M = (x,y,z)^T$ in the scene, and the corresponding projected point $m = (v,u)^T$ in the photographic medium. The model can be seen as a linear mapping, or projection, from the three dimensional scene space to the two dimensional image space. The mapping can also be described with homogeneous coordinates ($\widetilde{m}$, $\widetilde{M}$) and a projection matrix $\mathbf{P'}$ as in equation 4.3.

$$u = (-f \cdot x/z) \tag{4.1}$$

$$v = (-f \cdot y/z) \tag{4.2}$$

---

[1] A CCD is an array of light sensitive capacitors. The abbreviation stands for “Charge-coupled device”

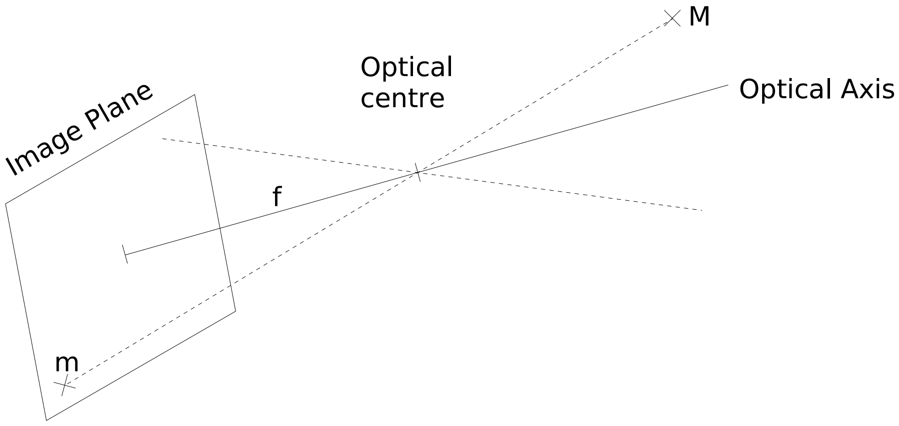

Figure 4.1.: The basic pinhole camera model

$$\widetilde{m} = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \widetilde{M} = \mathbf{P}'\widetilde{M} \tag{4.3}$$

4.2.2. Internal parameters

This very simple model can be extended to provide some more flexibility, more convenience and more accuracy. First of all, with the basic pinhole model images are upside down, and the left and the right sides are switched. It is more convenient to work with an equivalent model, where the image plane is at the distance $-f$, i.e. the image plane is *before* the optical center. In this case, the image is captured in the usual top-down and left-right orientation. This setup will be used from now on. Equation 4.4 shows the corresponding projection matrix.

$$\mathbf{P}' = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.4}$$

The coordinate system of the image has its origin usually in the top left corner, and the x coordinates increase to the right, while the y coordinates increase in downward direction. The photographic medium is not necessarily perfectly aligned, so the intersection of the optical axis and the photographic medium can can lay anywhere in the the image plane. This intersection is called the principal point. The coordinates of the principal point are $C = (u_0, v_0)^T$.

The coordinates in the scene are usually measured in a metric unit (like millimeters), whereas the coordinates in the image are measured in pixel. We can use another transformation matrix

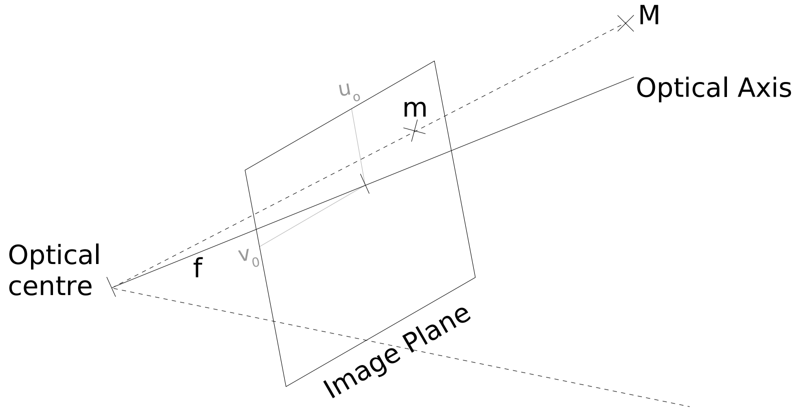

Figure 4.2.: The basic pinhole camera model

H to perform unit scaling and the translation of the principal point. **H** is shown in equation 4.5 with the unit scale factors being k_u and k_v.

$$\mathbf{H} = \begin{pmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

This matrix is known as the internal transformation matrix. The model that combines the pinhole camera model with a frontal image plane and internal unit transformation is shown in figure 4.2. Equation 4.6 shows the resulting coordinate transformation.

$$s\widetilde{m} = \begin{pmatrix} fk_u & 0 & u_0 & 0 \\ 0 & fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \widetilde{M} = \mathbf{H}\mathbf{P}'\widetilde{M} = \mathbf{A}\widetilde{M} \tag{4.6}$$

4.2.3. External parameters

Until now it has been assumed that the coordinate system of the scene and the coordinate system of the image are aligned, i.e. orientated in the same direction and that they have a common origin. It is much more practical to use different coordinate systems, especially if the camera can be moved or oriented freely. What is needed then is a euclidean transformation which converts the scene coordinate system into the image coordinate system, and its inverse transfromation converts in the opposite direction. Such a transformation consists of a rotation and a translation. The matrix **D** in equation 4.7 describes such a transformation. It converts a vector $\widetilde{M}_w$ of homogeneous scene coordinates into a vector $\widetilde{M}_c$ of homogeneous image coordinates (equation 4.8). **D** is called external transformation matrix. It should be noted that it has only six degrees of freedom, because the r_{ij} cannot be chosen arbitrarily, but they depend on three rotation angles [GL96].

$$\mathbf{D} = \mathbf{TR} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

$$\widetilde{M}_c = \mathbf{D}\widetilde{M_w} = [\mathbf{R}^* \ t]\widetilde{M_w} \tag{4.8}$$

$$\text{with } \mathbf{R}^* = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \\ 0 & 0 & 1 \end{pmatrix}, \ t = \begin{pmatrix} t_x \\ t_y \\ t_z \\ 1 \end{pmatrix}$$

Now all of these transformations can be combined into one, as in equation 4.9, where $\widetilde{m}_s$ is the image point in sensor coordinates (pixels) [Sch05]. s is a scale factor which has not much meaning because the coordinate vector in homogeneous representations can be multiplied arbitrarily, but it still represents the same coordinate.

$$s\widetilde{m}_s = \mathbf{HP}'[\mathbf{R}^* \ t]\widetilde{M}_w = \mathbf{AD}\widetilde{M}_w = \mathbf{P}\widetilde{M}_w \tag{4.9}$$

Here **P** is the general projection matrix. It has ten degrees of freedom, six for the orientation and position of the camera coordinate system, two for vertical and horizontal scaling (includes the focal length), and two for the position of the principal point.

4.2.4. Radial distortion

The last extension to the basic pinhole camera model will be radial lens distortion. Optical lenses often distort an image radially. This means the image magnification is not constant over the image, but depends on the distance to the optical axis. As a result, straight lines do not appear straight in the image (see figure 4.3).

The distortion function can be approximated by a polynomial (equation 4.10) [Tsa87], where the κ_i are distortion parameters, m_d are the distorted and m_u the undistorted image coordinates, and $r_d = \| m_d \|$. The approximation is usually good enough if only one second degree term is used. This means that only κ_1 will be used as distortion parameter. All other coefficients (κ_i, with $i \geq 2$) are set to 0. This restriction makes the model, especially the inversion of the distortion function, much simpler.

$$m_u = m_d(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4 + \ldots) \tag{4.10}$$

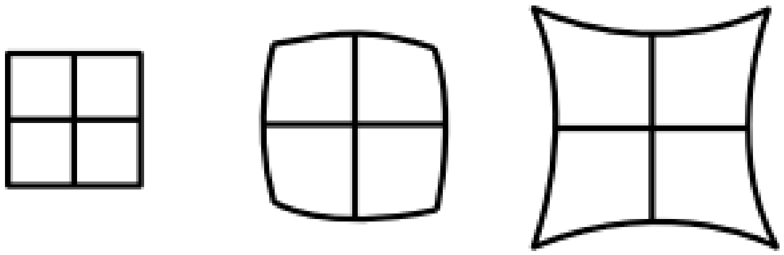

Figure 4.3.: Radial distortion
Undistorted left, barrel distortion middle, pincushion distortion right.
Figure taken from the Encyclopaedia Britannica, 1911 edition.

A more accurate model for radial distortion allows for the center of distortion to be anywhere on the image plane [DF95]. It is shown in equation 4.11, with c_d being the center of distortion. The center of distortion is often the principal point. The same rationale for why the principal point may be not in the exact center of the photographic medium, applies to the center of distortion.

$$m_u = m_d + (m_d - c_d)(\kappa_1 r_d^2) \tag{4.11}$$

Since radial lens distortion is a nonlinear process, it cannot be modelled with a matrix. Therefore it cannot be integrated easily into the existing mathematical model. As a result, any image obtained from the cameras must first be undistorted, i.e. another image must be computed from the source image which “undoes” the distortion, before further image processing can proceed.

A simple way of undistorting an image could be done by calculating the undistorted coordinates for each pixel in the distorted image using equation 4.10 (see algorithm 2). This approach is called *forward-mapping*.

However, with this approach a problem arises due to the discrete character of the image. It may happen that two neighboring pixels in the distorted image are transformed into pixels in the undistorted image which are no neighbours anymore. Instead, there is then a small gap between them, which will not be filled. These gaps will be present throughout the image, as can be seen in figure 4.4, and they will severely hinder further image analysis.

To circumvent this a slightly modified approach can be employed [Sch05]. Instead of taking every pixel position from the *distorted* image to compute the undistorted position, it is possible to consider every pixel in the *undistorted* image, and compute the coordinates of its corresponding pixel in the distorted image. This pixel is taken to fill the pixel in the undistorted image. This makes sure that every pixel in the undistorted image is filled, as long as there is a corresponding pixel within the distorted image’s bounds. Because the transformation follows the opposite direction in this approach, it is called *backward-mapping*.

Algorithm 2: Simple image undistortion algorithm

```
Function: undistortImage(Dist_Image)
Input: Dist_Image: The distorted Image
Output: Undist_Image: The undistorted image

begin
    Undist_Image ⟵ new Image
    foreach (x,y) from Dist_Image do
        $(x_u,y_u) \longleftarrow undistort(x,y)$ // Equation 4.10
        $p \longleftarrow Dist\_Image.getPixel(x,y)$
        $Undist\_Image.setPixel(x_u,y_u,p)$
    endfch
    return Undist_Image
end
```

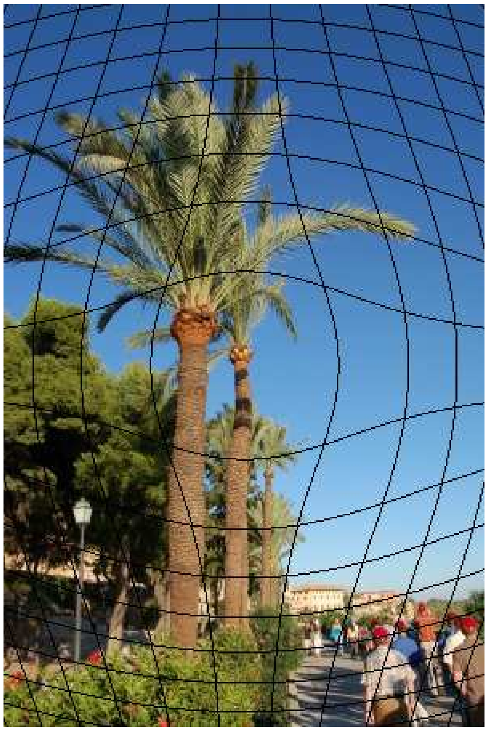

Figure 4.4.: Undistortion performed with forward mapping

Now the coordinates of the corresponding distorted pixel may have a fractional part, meaning that the source from the distorted image lies actually between two discrete pixels. This can be addressed by bilinear interpolation of the data from the source's four surrounding neighbours.

To calculate the distorted pixel positions it is necessary to invert equation 4.10, which is not trivial since it involves the solving of a cubic [DF95].

This approach of backward-mapping can in fact be applied to any image transformation which involves only a transformation of pixel coordinates. It will be used again for rectification in section 4.5.

4.3. Camera calibration

In the previous section the pinhole camera model was introduced. Central to the model is the projection matrix **P** (equation 4.9), which has ten degrees of freedom. Furthermore, the model can be extended by a parameter for radial distortion, adding one additional degree of freedom.

For further scene analysis, especially for photogrammetry, it is necessary to determine all these parameters. Only when the projection matrix is known, it is possible to calculate where a scene point will appear on the picture. The same is true for the other direction, i.e. the question "Given a pixel in the picture, what is the position and orientation of the light ray that produced this pixel?" can only be answered if the projection matrix is known.

The process of determination of these parameters is called *camera calibration*. The Tsai-Method [Tsa87] is one widespread method for calibration, although there are others [CT90, Rob96]. It requires a number of points in the scene whose coordinates are known with respect to the scene coordinate frame and the positions of the projections of these calibration points into the image. One of the advantages of the Tsai-Method is that it allows for the calibration points to be coplanar.

The Tsai-Method permforms an iterative error reduction to determine a set of parameters that fits the data best. This is done in two steps. First, estimates of the parameters are obtained by the linear least-squares method [LH95]. These estimates are used as staring values for a non linear optimization which enforces model restrictions in the second step.

Usually calibration is performed on a calibration rig with a checkerboard pattern. For this work however, a set of points in the scene has been marked with strips, with their coordinates known in the world coordinate system (see figure 4.5). When using a calibration rig, the camera parameters are obtained with respect to the rig's coordinate frame. With the approach used, the parameters are obtained with respect to the world coordinate frame, thus eliminating one potential source of inaccuracy, namely the transformation of the rig's coordinate frame into the world coordinate frame.

These calibration points are identified in the image manually (i.e. by human action). This seems acceptable, since calibration does not need to be performed repeatedly, but only when the camera setup has been modified.

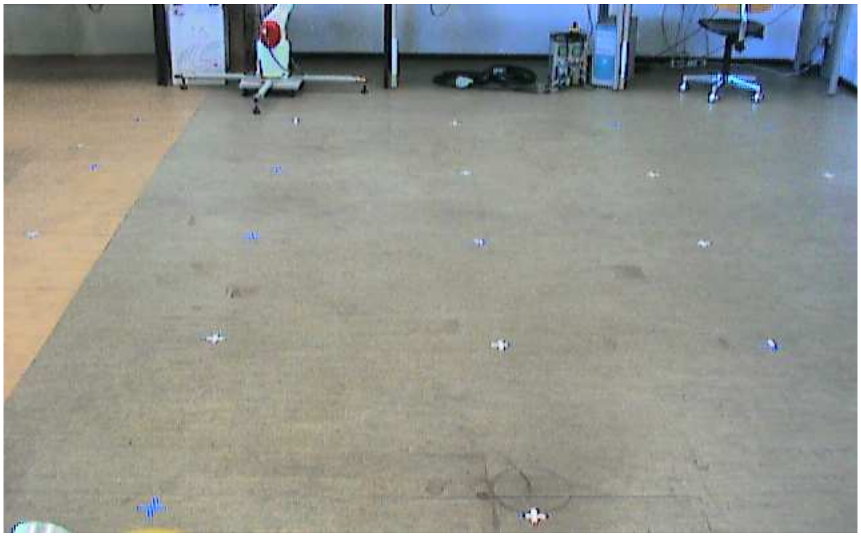

Figure 4.5.: Calibration scene

4.4. Geometry of two parallel cameras

When a scene is photographed from two different positions, the differences between the images can be used to infer information about the depth structure of the scene. Due to the difference in the cameras' positions a feature from the photographed scene is present at different image coordinates in the two pictures. This can be easily observed from drawing 4.6. The distance between the two image positions at which the same feature is pictured is called *disparity*. In figure 4.6 this would be the distance between m_1 and m_2, with their coordinates given in their respective image coordinate system.

To keep the geometric model simple, some restrictions on the cameras' internal and external parameters, their positions and orientations will be introduced. Firstly, the internal parameters must be the same for both cameras. This can be (approximately) achieved by using two identical cameras. Secondly, both cameras must have the same orientation in respect to the scene coordinate system. And finally, the cameras must be aligned in a manner that the position of the second camera results from a translation of the first camera position along the x-axis of the camera coordinate system. Such a set-up is called *parallel axis stereo*.

From these restrictions follows that there are seven degrees of freedom for the camera setup - six for the position and orientation of one camera, and one for the distance of the two cameras. The distance between the cameras[2], is called *baseline*.

In practice it might be quite difficult to meet these restrictions. The cameras' internal parameters could differ in spite of both cameras being build equal. And without a rigid mount for the cameras, it might also be difficult to align them perfectly. Nevertheless the simple geometric model can be useful. As section 4.5 will show, there is a method for transforming two images

---

[2] to be more precise, the distance between the optical centres of the two cameras

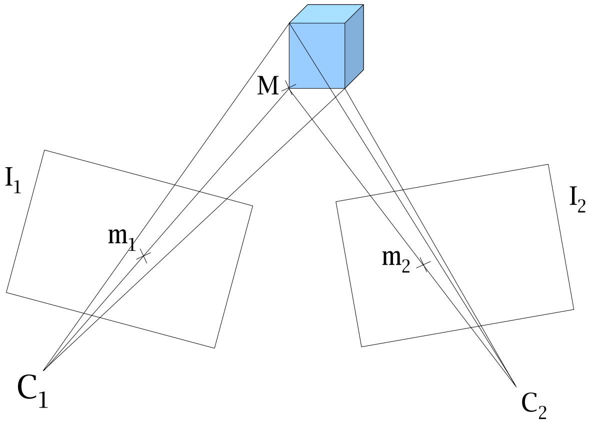

Figure 4.6.: Photographing a scene with two cameras

which were taken with arbitrary external and internal parameters, into images that were taken with a camera set-up which meets the mentioned requirements.

The advantage of axis parallel stereo is twofold. Firstly, its geometry is very simple, which is shown in figure 4.7. And secondly, it makes finding feature correspondences much easier because all features are shifted between both images only in horizontal direction. Two corresponding pixels always have the same y-coordinate. The problem of finding correspondences will be addressed in section 4.6.

It is obvious that the size of the disparity depends directly on the depth of the underlying three dimensional point. What is now needed is an equation that shows the relation between disparity and depth. With such an equation it is possible to calculate the distance of a 3D-point, for which a pair of corresponding image pixels is found, to the cameras. With this distance, and the known camera position in the scene, it is possible to determine the three coordinates of the point in the scene coordinate system.

According to [Sch05], the disparity δ is determined by equation 4.12.

$$\delta = u_1 - u_2 \tag{4.12}$$

From figure 4.7 can be inferred with similar triangles that the ratio between ρ and f is the same as the ratio between B and $\delta \cdot d_u$, with d_u being a scale factor that translates the disparity, which is measured in pixel, into the unit of the focal length (usually millimeters). This can be written as in equations 4.13 and 4.14 [Sch05].

$$\frac{\rho}{f} = \frac{B}{d_u \cdot \delta} \tag{4.13}$$

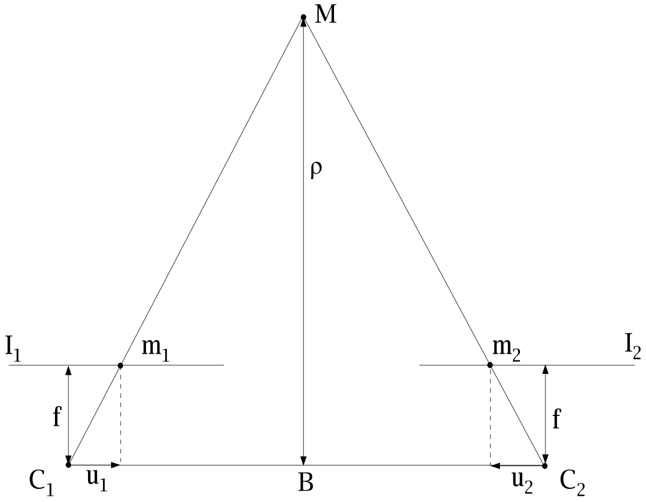

Figure 4.7.: Axis parallel stereo setup (top view)
Drawing taken from [Sch05].

$$\Leftrightarrow \quad \rho \quad = \quad \frac{B \cdot f}{d_u \cdot \delta} \tag{4.14}$$

With these equations it is easily possible to calculate the depth of a feature, for which two corresponding pixels in the left and the right image have been identified if a parallel stereo setup was used. Together with the external camera parameters, which describe the position and the orientation of the camera, it is possible to determine the 3D coordinates of that feature.

4.5. Rectification

As mentioned before, a parallel stereo setup is often desirable, but at the same time sometimes difficult to achieve. However, it is possible to transform the stereo pictures taken by an arbitrary stereo setup into two new pictures, which look like they were taken with a parallel stereo setup. This transformation is called *rectification*.

There are different approaches to rectification. [Har99] proposes an approach which does not require knowledge of the cameras' parameters (i.e. no calibration is needed). However, methods that use the cameras' parameters are much more simple. Since these parameters are needed anyway in the stereo vision approach presented in this thesis, there are no objections to use such a method. The rectification in this work has consequently followed the approach described in [FTV00].

When a pair of images is rectified, the images are projected virtually onto a new image plane. The new image planes are chosen as if they were the image planes of a parallel stereo setup.

The optical center of the old and the new image planes stays the same, so the transformation between the image planes is a rotation around the optical center.

According to the camera model discussed in section 4.2, the projection equation for the left[3] image is 4.15. The projection equation onto the new, virtual image plane is 4.16. The projection matrix $\mathbf{P}_o = [\mathbf{Q}_o|q_o]$ is obtained by calibration, and $\mathbf{P}_n = [\mathbf{Q}_n|q_n]$ can be chosen within the constrains stated above.

$$s_o\widetilde{m}_o = \mathbf{P}_o\widetilde{M}_w \tag{4.15}$$

$$s_n\widetilde{m}_n = \mathbf{P}_n\widetilde{M}_w \tag{4.16}$$

Because the optical centres are the same for both images, $\widetilde{m}_o$ and $\widetilde{m}_w$ are at the same optical ray, which connects $\widetilde{M}_w$ and the optical center C_1, thus equations 4.17 and then 4.18 are obtained. The scale factor s in equation 4.18 can be ignored, because $\widetilde{m}_n$ is a homogeneous coordinate representation. Thus all required parts of the rectification transformation $\mathbf{T} = \mathbf{Q}_n\mathbf{Q}_o^{-1}$, are given or can be computed easily.

$$C_1 + s_o\mathbf{Q}_o^{-1}\widetilde{m}_o = C_1 + s_n\mathbf{Q}_n^{-1}\widetilde{m}_n \tag{4.17}$$

$$\widetilde{m}_n = s\mathbf{Q}_n\mathbf{Q}_o^{-1}\widetilde{m}_o \tag{4.18}$$

4.6. Stereo correspondence analysis

Section 4.4 described how to determine the three dimensional position of a feature from its disparity, i.e. the distance between the positions of this feature in both pictures. However, the disparity cannot be determined before the same feature is identified in both images. The process of finding features in both images which show the same three dimensional structure is called *stereo correspondence analysis.*

It has already been noted in the previous section (4.5), that rectification makes stereo correspondence analysis much simpler, because corresponding features must have the same y-coordinate in both images. Therefore, the search space for the corresponding feature in the other image can be limited to one line. Still, stereo correspondence analysis remains a complex and error prone task.

In general there is no guarantee that a feature from the left image will have a corresponding feature in the right. The other feature might not be included in the right image, because it is out of scope of the right image (this often happens to features near the left border of the left image). Furthermore, the different camera positions make it possible that a feature visible in the left image is is concealed by some other object in the right image. Finally, a feature might look very different from the other perspective so that it is very difficult to identify it as the same structure.

---

[3] The steps needed for the rectification of the right image are the same, therefore it will be omitted

These problems tend to increase when the baseline of the camera setup is increased. But a large baseline is otherwise desirable because it makes the results of stereometry more exact. In this work, however, there was no possibility to test the results of different baselines because the camera mount had no provision to change the baseline. Thus, the baseline was fixed to approximately 12 cm, which seems to be a fairly reasonable choice for the assigned task. In [Joc06], different baselines in are tested with a different camera setup. These results support the claim, that 12 cm is a reasonable baseline for stereometry in the range from 1 to 10 meters.

On the other hand, it is possible that there is a feature in the left image, for which the correspondence analysis algorithm finds more than one corresponding feature in the right image, and cannot determine which one is the “real” correspondence because all seem equally good. The algorithm might also choose the wrong one because the correct correspondence could have been distorted by noise or it looks differently due to geometrical effects. Such ambiguous correspondences occur often when there are periodic structures in the image, areas without structure or when the images have much noise. They can lead to wrongly identified correspondences (miscorrespondences).

Stereo correspondence analysis can be performed at different semantic levels. At the lowest level corresponding pixels are considered, i.e. for each pixel in the left image a corresponding pixel in the right image is searched for and vice versa. At the higher levels more complex features are considered, starting from corners, over line segments, contours, regions, up to entire objects. Of course these features are exemplary, and other features may also be considered.

The choice of the level has great impact on the speed, error rate, information amount and correspondence ambiguity. At higher semantic levels, the correspondence analysis is performed faster and more reliably, because there a fewer features, thus fewer potential matches, and the features are more complex, so ambiguities are less likely to occur. However, these higher level features must first be extracted from the original image, which can be time consuming and error prone. Therefore, higher level feature correspondence analysis is not automatically faster and more reliable than correspondence analysis with lower level features.

Another downside of high level analysis is that because there are less features, there will be less depth information gained from the process, because depth information is only generated for corresponding features. Whereas in pixel based analysis depth information can be ideally gained for each pixel in the image, in object based analysis there will be depth information only for the few detected objects (or none if no object is detected at all).

Pixel based analysis seemed to produce too many miscorrespondences [Joc06] when used for this thesis. There is the danger that singular miscorrespondences and the resulting wrong depth information would clutter up the map with points (i.e. obstacles) at random positions. It could be difficult to filter them out, so this approach was discarded.

Instead, line based analysis seemed to be a much more workable approach. Lines segments are not too difficult to detect, as will be shown in 4.7, and with line segments miscorrespondences seem very unlikely. They should provide sufficient information, because in an office environment, obstacles nearly always feature at least some straight edges which can be detected. In other settings, however, straight edges are much less common. A robot navigating in a wilderness (i.e. forest, desert, meadows, etc.) would face difficulties in finding edges. For this reasons, it was decided to use line segment based analysis in this thesis.

4.7. Line segment extraction and processing

This section will describe how to detect line segments in images and how to match corresponding line segments from a pair of stereo images.

The detection of line segments is done in a three stage process. These stages will be described in detail in the following sub-sections. Firstly, an edge detection operator is applied to the image (4.7.1). Then, a Hough transform is performed on the resulting image (4.7.2), which can detect the main lines in an image. The last step is to divide the obtained lines into segments (4.7.3), by comparing the lines with the edge image.

The last subsection (4.7.4) will outline the methods that can be used to detect corresponding line segments from pairs of stereo images.

4.7.1. Edge detection

Within an image boundaries of objects often feature sharp changes in luminous intensity. This is because one object usually hides another region which differs in colour, texture, lighting, etc. As man made objects often feature linear boundaries, one can observe such luminous intensity changes along a line segment. The task of edge detection is to find those pixels in an image which have great changes in intensity compared to their neighbours.

Of course not every edge must necessarily be a boundary of an object as it can also be caused for many other reasons, e.g. a change of the object's surface structure or a shadow. However, this does not matter in this case as any detected edges should point to the presence of an object, and the aim is to find objects and measure their position.

Most methods for edge detection have one thing in common: They consider the first or second order derivate of the image to find regions with sharp intensity changes. Because an image is a two dimensional structure, the intensity change at any pixel depends on the direction of the change. The various methods for edge detection, like the Prewitt [GW03], Sobel [GW03], Laplace [MH80], Kirsch [Kir71], Roberts [Rob65] or Canny [Can86] operators, differ mostly in how they handle the direction of the intensity change (gradient). Whereas some operators, like Prewitt or Sobel are sensitive to a intensity change only in either vertical or horizontal direction, more sophisticated edge detection algorithms like the Canny Algorithm are independent of the gradient's direction.

For this thesis the Laplace-Operator was chosen because it provides independence of the gradient's direction and it is easy to implement. No systematic research was made as to which operator would provide the best results, because it seems not likely that other edge detection algorithms would improve the overall process significantly.

The Laplace-Operator is a convolution of the image matrix with a kernel, which represents a discrete approximation of a Laplacian of a Gaussian (LoG, see table 4.1). The LoG is shown in equation 4.19 with σ being the standard deviation, and figure 4.8 shows an exemplary graph. The kernel function is radially symmetric; therefore the direction of the image gradient does not influence the result.

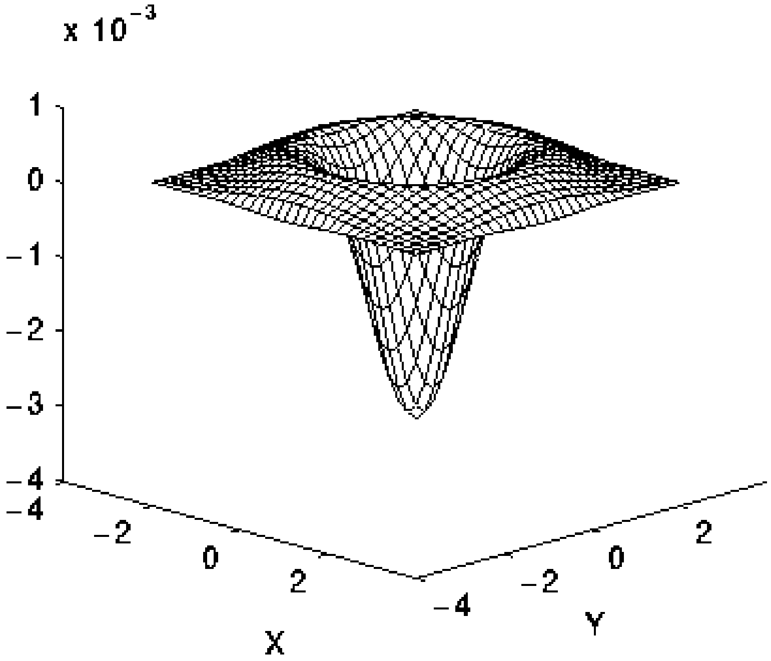

Figure 4.8.: The Laplacian of a Gaussian

-0.27	0.39	0.65	0.39	-0.27
0.39	0.52	-0.69	0.52	0.39
0.65	-0.69	-4	-0.69	0.65
0.39	0.52	-0.69	0.52	0.39
-0.27	0.39	0.65	0.39	-0.27

Table 4.1.: The convolution kernel, a discrete approximation of the Laplacian of a Gaussian

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.19}$$

The result of the application of the Laplace-Operator is a derivate of the original image. Regions of high intensity change (i.e. edges) are now easily identifiable. They have high pixel values, while all other areas have low values.

Figure 4.9 shows a typical scene in the robot laboratory. In figure 4.10, the result of the application of the Laplace-Operator on this image is shown[4].

---

[4] The image is actually inverted only to increase its visibility

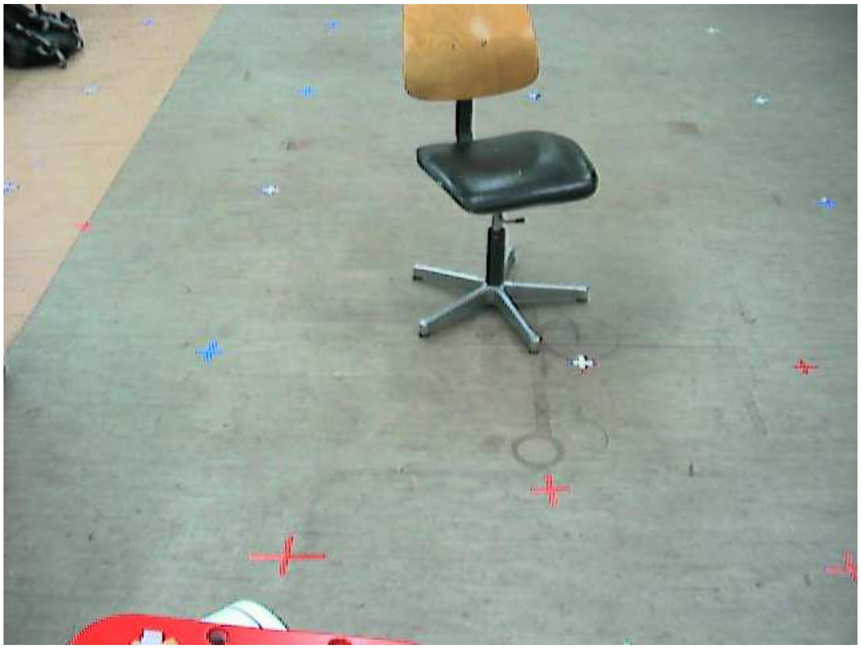

Figure 4.9.: The original chair scene

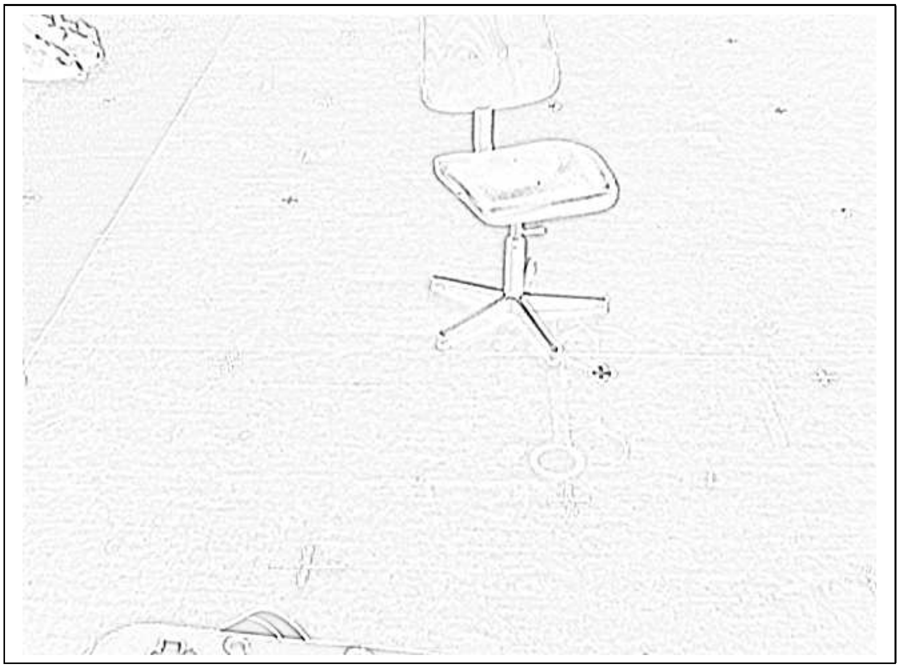

Figure 4.10.: The chair scene after edge detection

4.7.2. Hough transform

The edge detection only highlighted those pixels which are in a region of sharp intensity changes. What is needed to be done now is to group these pixels into line segments. This means that groups of edge pixels that can be approximated by a line segment have to be identified and the parameters of the line segment have to be determined. This can be accomplished with the help of the Hough transfrom [Hou62].

The Hough transfrom in general is able to detect features of a particular shape in images. This section will be limited to the ability of the Hough transform to detect straight lines.

For reasons which will become apparent later the Hough transform requires a parametric description of the lines. In the original presentation of the Hough transform [Hou62] the very common paramertic description of a line as in equation 4.20 is used. When values are assigned to the parameters a and b, the resulting line is the set of all points $P = (x, y)$, which fulfill equation 4.20. However, this representation is not a very convenient choice, because vertical lines cannot be represented with this description[5]. This in turn causes problems in the implementation of the Hough transform.

$$y = ax + b \tag{4.20}$$

Therefore, a different parametrical description will be used here, which is called *Hesse normal form*. It is shown in equation 4.21, with the parameters being r and θ.

$$x\cos\theta + y\sin\theta = r \tag{4.21}$$

With this parametrical description, r is the length of the normal from the origin of the coordinate system to the line, and θ is the angle between this normal and the x-Axis. This geometric interpretation is shown in figure 4.11. With the Hesse normal form θ is in $[0; 2\pi]$ and the lower bound of r is 0 while the upper bound is determined by the image size. The usage of the Hesse normal form with the Hough transform was introduced by [DH72].

For each edge point P its image coordinates are known, but what is unknown are the parameters of the line that passes through this point. A unlimited number of lines could pass through this point. The Hough transform now marks all possible combinations of (θ, r) (i.e. all possible lines through P) in the parameter space. They form a sinusoidal curve in this space.

This *point-to-curve* transform is now performed for all edge points, each one generating a different curve in the parameter space (Hough space). Points that are collinear in the image space generate curves in the parameter space which intersect in one common point. This follows directly from the presumption that they are collinear, therefore there must be one line common to all of these points. This line has a certain parametrical description (θ, r), and all the curves from the different points must contain this point.

The Hough transform can be implemented in a computer with an accumulator matrix. The parameter space is discretized into many buckets for each parameter. For each edge point, the

---

[5] As a line approaches vertical orientation, a approaches infinity

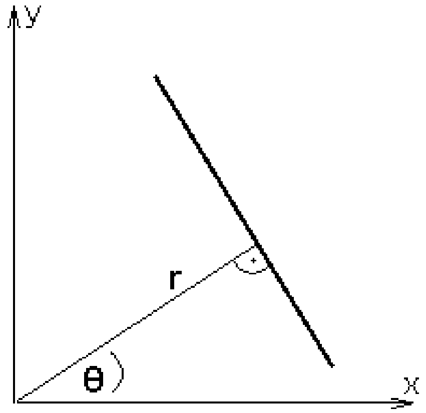

Figure 4.11.: The Hesse normal form

parameter space curve is calculated and the accumulator value of each bucket the curve passes through is increased. After this has been repeated for each significant edge point, the edge points of a common line have caused many accumulator increases in a common bucket. This is the bucket that represents the intersection of the parameter curves from the edge points. Now the buckets with the highest values in the matrix describe the most significant straight lines in the image.

4.7.3. Line segmentation

The Hough transform, as described in the previous section, provides information about the position and orientation of lines in the image. It does not provide the information where line segments begin and where they end. But together with the edge filtered image this is easy to determine.

Line segments are part of the lines obtained by the Hough transform. Therefore the image can be scanned along such a line while testing if a particular pixel is part of a line segment or not. A simple way to do this is to test the pixel's intensity. If the intensity value pixel exceeds a certain threshold, a start of a segment is assumed. If it falls below the threshold, the end of the segment is reached.

This simple algorithm can be modified to make it more robust against noise although the edge filtered image is already noise reduced. Modifications can include two different thresholds for the start and the end of a segment, making the choice dependent of the last few pixels, or varying the threshold depending on the last pixel values.

Figure 4.12 shows the chair scene after the hough transform has been applied and the lines have been divided into segments as described above.

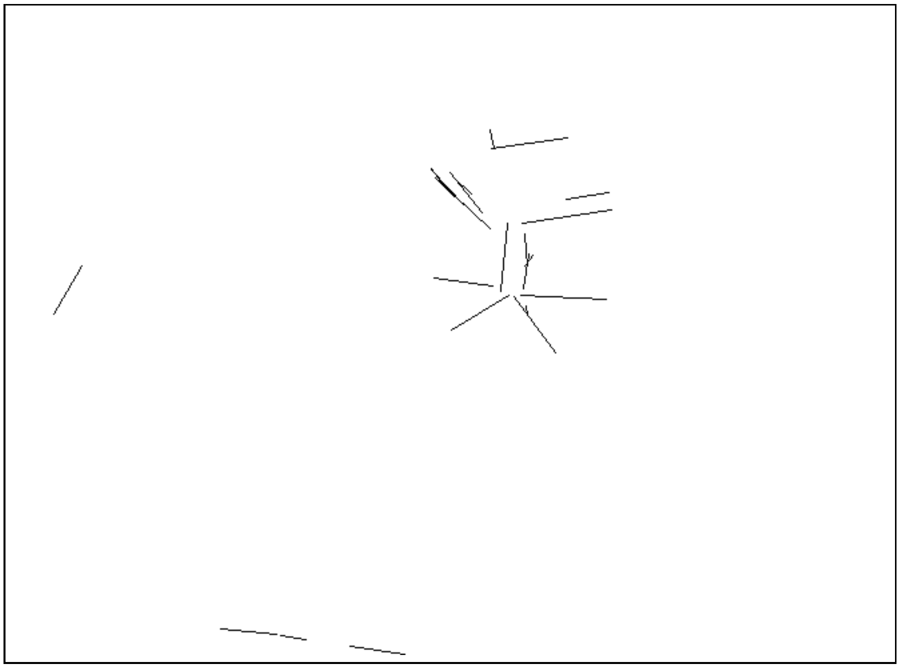

Figure 4.12.: The chair scene after Hough transform

4.7.4. Line segment matching

Section 4.6 stated the problems of stereo matching in general. This section will discuss how matching of line segments in particular can be performed.

Although rectification aligns pixels in both the left and the right image onto the same y-coordinate, corresponding line segments might have different y-coordinates at the start and the end anyway. The reason for this is, that the differences in the stereo image pair, which are caused by the different perspectives of the cameras, may have impact on the each processing stage (edge detection, hough transform, line segmentation).

For example, an object might occlude a pattered background. Then, the colour transition on the edge between the object and the background might be different on both images, due to the different perspectives. As a result, the edge is more intense in the left image than in the right, and therefore a longer line segment might be obtained from the left image than from the right.

This example shows that it cannot be taken for granted that corresponding line segments have the same y-coordinates. However, the difference in y-coordinates cannot become arbitrarily high, therefore a windowed search for corresponding segments is reasonable.

A common technique to decide whether two samples are similar is the use of the normalized feature vector distance. This means that several features of the samples are to be extracted and quantified. These feature values are then normalized and stored together in a feature vector. For each sample such a feature vector is calculated. The distance of the feature vectors of the various samples then is a measure of how similar the samples are.

If this technique is to be applied to compare line segments, following features can be used:

- The length of a segment
- The orientation of a segment
- The distance form the origin to the center of the segment
- The orientation of the vector from the origin to the center of the segment
- The (average) intensities left and right to the line segment in the image.

A problem arises with the use of orientations as a segment feature. Due to the periodic nature of angles, the quantification of the orientation should receive special attention. Otherwise a 1° orientation and a 359° orientation would cause a huge feature vector distance, where in fact the segments are very similar. Therefore a special provision must be made when calculating the feature vector distance, so that the difference of two angles cannot exceed 180°.

4.8. Summary

This chapter explained the many steps necessary to obtain depth information from stereo images. It began with a brief overview of the development and mode of operation of natural stereo vision. Then the camera model for artificial cameras was introduced and camera calibration was described, which estimates the parameters for the camera model. The geometry of the stereo setup was explained in the next section. With this geometrical model it is possible to establish the numerical relashionship between disparity and distance. Provided that there is exact calibration data, rectification allows to transform arbitrary images into images from a virtual parallel stereo setup, which simplifies the subsequent evaluation stages. To determine disparities, corresponding pairs of features have to be found in the stereo images. The last section explained how line segment features are extracted from images, and how the search for corresponding features can be achieved.

With this knowledge it is possible to implement a stereometry system which is able to detect obstacles and measure their distance to the cameras. The next chapter will elaborate on which problems arose during this implementation and what results have been achieved in conjunction with the navigational system.

Results and conclusion

5

5.1. Planning and movement

Many experimental tests of the planning algorithm with the robot have shown that the algorithm proves to be a robust and performant basis to the navigational system. Under many different conditions, the planing algorithm generated safe paths. Only in a few cases, in the presence of passages with only few centimeters of space to manoeuvre, the algorithm failed to produce a path. But to master these situations is only a question of parameters. With a higher resolution, which of course would increase planning time, a path would have been found.

The data from the laser range finders enables to robot to react to changes in the environment and to handle unforeseen situations (as a blocked doorway). Many test drives have been conducted with the robot, and the robot was able to navigate through various sets of obstacles, doorways and walking people without colliding. Initial problems with obstacles that get outside the visible area, as described in section 3.3 were alleviated by remembering obstacle positions over a period of time.

The planning algorithm has proved to be fast enough. Most of the time, a stop of motion to perform planning was hardly noticeable. Longer planning phases occurred rarely, and were in the order of a few seconds.

One downside of the algorithm is, that movements in narrow passages sometimes still are a bit erratic. This means, the robot moves a little step forward, then makes a turn, then the next step and so on. Although measures have been taken to avoid this, as stated in section 2.3.2 and 2.5, this behavior cannot eliminated completetely, because the software responsible of controlling the robot's drive limits movements to straight translations or rotations. This has no grave consequences, it only slows down the movement and looks a bit awkward. A control software that enables the robot to follow curves, e.g. B-splines, could be used to avoid such behaviour.

Another aspect that could be improved is the classification and memory of obstacles, as pointed out in section 3.3. With object recognition, the planner could make a better estimate whether an obstacle has rather a permanent or a temporary character. Planners that can resort to world knowledge, behaviors, and learning possibly could manage much more complex environments, than the office scenario. But these techniques are part of a broad field of research that still requires much investigation.

However, the limitations of this planning approach do not pose major problems in practice. Situations in which these limitations surface occur very rarely in the given scenario, thus the results achieved by these parts of the navigational system were entirely satisfactory.

5.2. Obstacle sensing and stereometry

The laser range finders provided data which resulted in an accurate representation of the environment. Therefore many obstacles can be detected by the navigational system and a new path around them can be planned. However, there remains the principal uncapability of the laser range finders to detect some obstacles reliably like tables or hanging objects. The stereo vision approach was assigned to resolve this problem.

Stereo vision is a much more complex task than reading and interpreting data from the laser range finders. Many stages of generating and processing data are involved, which have been explained in chapter 4. Unfortunately, in the initial step a problem has occurred, which led to a poor quality of the stereometry results.

Stereometry needs very exact calibration data, and especially the external camera parameters have a major impact on the results. From simple comparative analysis of the stereo images and the generated calibration data could be concluded that the camera model parameters, calculated by camera calibration (see 4.3), were not accurate enough.

The estimated camera orientation differed from the real orientation for only about one or two degrees. This in turn leads to wrong rectification and errors in the disparities of a few pixels. This may sound not much, but disparity values in this setup for objects that are a 2 meters away are in the order of 30 pixels[1], so a few pixels make a big difference.

An example for this inaccuracy is given in figure 5.1. It shows a rectified stereo image pair. The disparity of the white cross mark on the floor close to the bottom of the chair is 25 pixels, which translates to a depth distance of 3.83 meters, but the real distance is only 3 meters. Such levels of accuracy error are too large for useful obstacle distance measurement. Therefore the integration of depth information from stereo cameras had to be postponed until the reasons for the inaccurate calibration data has been identified and eliminated.

Possible factors that may contribute to the inaccuracy of the camera calibration include:

- Inexact placement or measurement of the calibration marks.
- Poor image quality (The right camera always produces slightly blurred images, as can be seen in figure 5.1).
- Poor working of the camera. If the placement of the CCD sensor is not exactly orthogonal to the optical axis, the camera model does not fit, and therefore many assumption made during image processing do not hold.

In a further work this problem could be investigated. It could try to reduce the inaccuracies by using another calibration setup, a different camera, a larger baseline or another calibration method. Once this problem is overcome, the results from the stereometry can be integrated into the navigation system. This would make it possible to verify whether this approach is able to detect obstacles reliably in an office environment. If so, this would make the navigation system more robust, as more obstacles could be detected reliably.

---

[1] The image size is 640*480 pixels

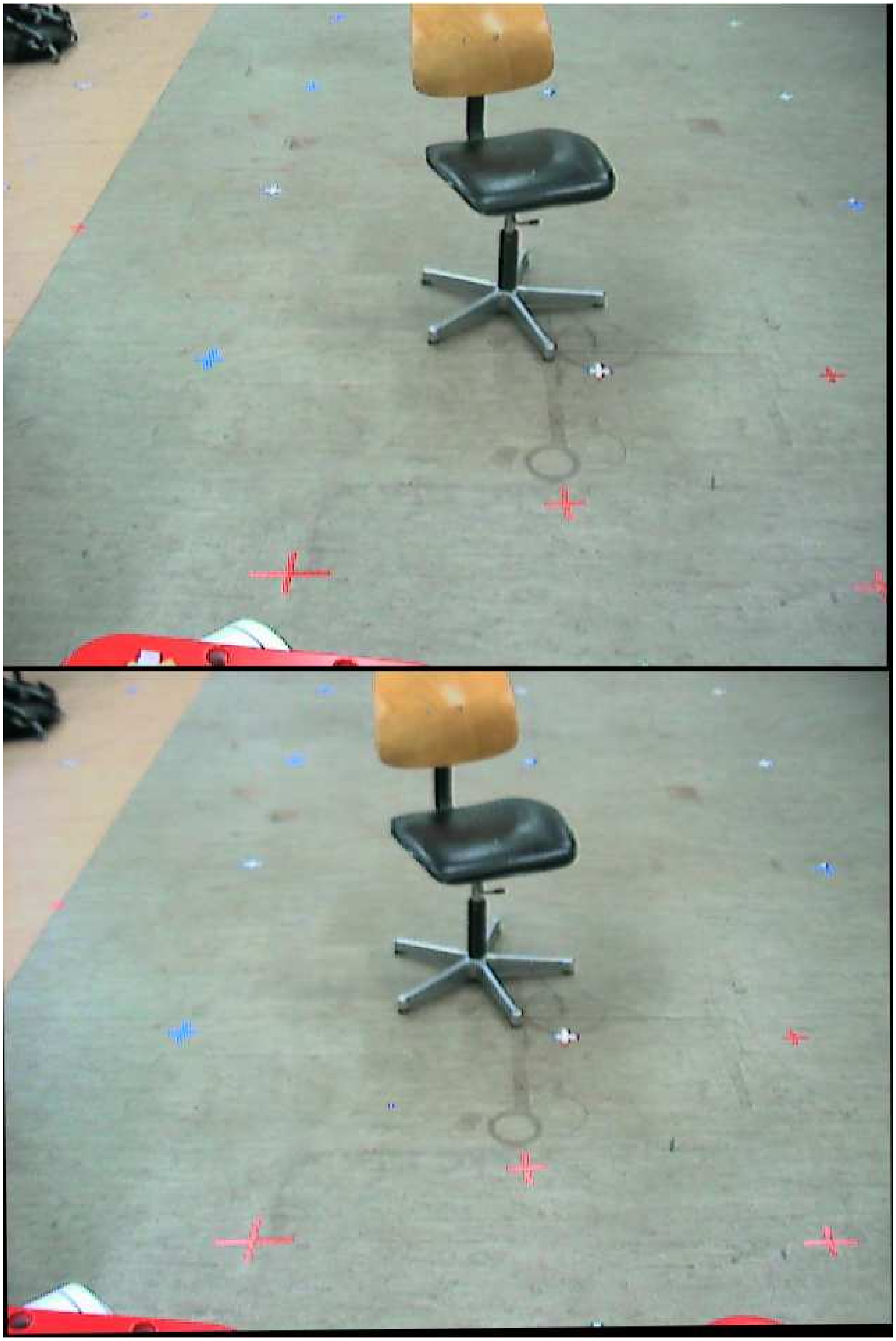

Figure 5.1.: A stereo image pair

Bibliography

[AH83] J. R. Andrews and Neville Hogan. Impedance control as a framework for implementing obstacle avoidance in a manipulator. *American society of mechanical engineers winter conference*, pages 243–251, November 1983. Reprinted in: Control of Manufacturing Processes and Robotic Systems, editors David E. Hardt & Wayne J. Book, American society of mechanical engineers.

[BA04] Geovany Araujo Borges and Marie-José Aldon. Line extraction in 2D range images for mobile robotics. *Journal of Intelligent Robotics Systems*, 40(3):267–297, 2004.

[Bar98] Robert A. Barton. Visual specialization and brain evolution in primates. *Proceedings of the Royal Society B: Biological Sciences*, 265(1409):1933–1937, October 1998. `http://www.journals.royalsoc.ac.uk/link.asp?id=mylye2x05ynrem5x`.

[Bar04] Robert A. Barton. Binocularity and brain evolution in primates. *Proceedings of the National Academy of Sciences of the United States of America*, 101(27):10113–10115, July 2004. `http://www.pnas.org/cgi/reprint/101/27/10113.pdf`.

[BWW03] Christian Brenneke, Oliver Wulf, and Bernardo Wagner. Using 3D laser range data for SLAM in outdoor environments. In *Proceedings of the 2003 IEEE International Conference on Intelligent Robots and Systems*, pages 188–194, 2003.

[Can86] John F. Canny. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[CLH$^+$05] Howie Choset, Kevin M. Lynch, Seth Hutchington, et al. *Principles of Robot Motion*. The MIT Press, 2005.

[CT90] B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4(2):127–140, 1990.

[DF95] Frédéric Devernay and Olivier Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. In *Proceedings of the SPIE Conference on Investigative and Trial Image Processing*, volume 2567, San Diego, California, USA, July 1995. `ftp://ftp-sop.inria.fr/robotvis/html/Papers/devernay-faugeras%3A95b.ps.gz`.

[DH72] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[Elf89] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. `http://eavr.u-strasbg.fr/~bernard/education/ensps_3a/tmp/elfes.pdf`.

[FTV00] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12:16–22, 2000.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 3rd edition, October 1996.

[GW03] Rafael C. Gonzales and Richard E. Woods. *Digital Image Processing.* Prentice Hall, 2003.

[Har99] Richard I. Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, 1999. `http://users.rsise.anu.edu.au/~hartley/Papers/joint-epipolar/journal/joint3.pdf`.

[HNR68] Peter E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths in graphs. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[Hou62] Paul Hough. *Methods and means for recognising complex patterns.* United States Patent 3,069,654, Dec 1962.

[HS01] Scott Husband and Toru Shimizu. Evolution of the avian visual system. In Robert G. Cook, editor, *Avian visual cognition*, Tufts University, 2001. Comparative Cognition Press. `http://www.pigeon.psy.tufts.edu/avc/husband/`.

[HW86] John E. Hopcroft and Gordon T. Wilfong. Reducing multiple object motion planning to graph searching. *Society for Industrial and Applied Mathematics Journal on Computing*, 15(3):768–785, 1986.

[HY04] Yanrong Hu and Simon X. Yang. A knowledge based genetic algorithm for path planning of a mobile robot. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 4350–4355, 2004.

[Jar73] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2:18–22, 1973.

[Joc06] Sascha Jockel. 3-dimensionale Rekonstruktion einer Tischszene aus monokularen Handkamera-Bildsequenzen im Kontext autonomer Serviceroboter. Master's thesis, Hamburg University, Faculty of Informatics, TAMS research group, Hamburg, Germany, 2006. `http://tams-www.informatik.uni-hamburg.de/personal/jockel/papers/Jockel06Diplomarbeit.pdf`.

[Kal60] Rudolph Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, 1960.

[Kha86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[Kir71] R. A. Kirsch. Computer determination of the constituent structure of biological images. *Computer Biomedical Research*, 4(3):315–328, June 1971.

[KM04] M. Kazemi and M. Mehrandezh. Robotic navigation using harmonic function-based probabilistic roadmaps. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 4765–4770, 2004.

[LA92] Y. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *International Journal of Robotics and Research*, 11(4):376–382, 1992.

[Lat98] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, USA, 5th printed edition, 1998.

[LGB02] Jean-Claude Latombe and Héctor H. González-Baños. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 21:829–848, October-November 2002. `http://ai.stanford.edu/~latombe/papers/ijrr-hhg/paper.pdf`.

[LH95] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems.* Society for Industrial & Applied Mathematics, September 1995.

[Lin04] Frank Lingelbach. Path planning using probabilistic cell decomposition. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 467–472, 2004.

[LPW79] T. Lozano-Perez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979. `http://www1.cs.columbia.edu/~allen/F05/vgraph.lozano.pdf`.

[ME85] Hans P. Moravec and Alberto E. Elves. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116–121, St. Louis, Missouri, USA, March 1985. `http://www.ri.cmu.edu/pub_files/pub4/moravec_hans_1985_1/moravec_hans_1985_1.pdf`.

[MH80] David Marr and E.C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B-207(1167):187–217, 1980.

[MJ97] Don Murray and Cullen Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 1694–1699, 1997.

[MSKS04] Yi Ma, Steffano Soatto, Jana Košecká, and S. Shankar Sastry. *An Invitation to 3-D Vision.* Springer, 2004.

[NMTS05] Viet Nguyen, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *Proceedings of the IEEE/RSJ Intenational Conference on Intelligent Robots and Systems, IROS.* IEEE, 2005. `http://asl.epfl.ch/aslInternalWeb/ASL/publications/uploadedFiles/nguyen_2005_a_comparison_of.pdf`.

[NN04] Hiroshi Noborio and Ryo Nogami. A new sensor-based path-planning algorith whose path length is shorter on the avarage. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 2832–2839, 2004.

[OF05] Guiseppe Oriolo and Luigi Freda. Frontier-based probablistic strategies for sensor-based exploration. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3892–3898, 2005.

[OVFT04] Guiseppe Oriolo, Marilena Vendittelli, Luigi Freda, and Giulio Troso. The srt method: Randomized strategies for exploration. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 4688–4694, 2004.

[PCGK95] Thomas V. Papathomas, Charles Chubb, Andrei Gorea, and Eileen Kowler. *Early Vision and Beyond*. Bradford Book, 1995.

[PG75] J. Packwood and B. Gordon. Stereopsis in normal domestic cat, siamese cat, and cat raised with alternating monocular occlusion. *Journal of Neurophysiolgy*, 38, 1975.

[PH74] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, c23(6):860–870, 1974.

[Rob65] Lawrence G. Roberts. Machine perception of three-dimensional solids. In James T. Tippett, editor, *Optical and Electro-optical Information Processing*, pages 159–197, Cambridge, MA, USA, 1965. MIT Press. `http://www.packet.cc/files/mach-per-3D-solids.html`.

[Rob96] Luc Robert. Camera calibration without feature extraction. *Computer Vision Image Understanding*, 63(2):314–325, 1996.

[Sch05] Oliver Schreer. *Stereoanalyse und Bildsynthese*. Springer, 2005.

[SE04] Juan Manuel Sáez and Francisco Escolano. A global 3D map-building approach using stereo vision. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 1197–1202, 2004.

[SFG$^+$04] Kohtaro Sabe, Masaki Fukuchi, Jens-Steffen Gutmann, et al. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 592–597, 2004.

[Ste94] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317, 1994. `http://www.frc.ri.cmu.edu/~axs/doc/icra94.pdf`.

[Ste95] Anthony Stentz. The focussed D^* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, August 1995. `http://www.frc.ri.cmu.edu/~axs/doc/ijcai95.pdf`.

[SW02] Axel Schneider and Daniel Westhoff. Autonomous navigation and control of a mobile robot in a cell culture laboratory. Master's thesis, University of Bielefeld, Faculty of Technology, Technical Computer Science, Bielefeld, Germany, 2002. `http://tams-www.informatik.uni-hamburg.de/personal/westhoff/publikationen/diploma_thesis.pdf`.

[Tsa87] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987. `http://www.vision.caltech.edu/bouguetj/calib_doc/papers/Tsai.pdf`.

[VVB05] Robert Vogl, Markus Vincze, and Georg Biegelbauer. Finding tables for home service tasks and safe mobile robot navigation. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3046–3051, 2005.

[WACW04] Oliver Wulf, Kai O. Arras, Henrik I. Christensen, and Bernardo Wagner. 2D mapping of cluttered indoor environments by means of 3D perception. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 4204–4209, 2004.

[WSS$^+$04] Daniel Westhoff, Hagen Stanek, T. Scherer, et al. A flexible framework for taks-oriented programming of service robots. In *Robotik 2004*, volume 1841, Munich, Germany, 2004. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, VDI Berichte. `http://tams-www.informatik.uni-hamburg.de/personal/westhoff/publikationen/robotik2004.pdf`.

[Zel92] Alexander Zelinsky. A mobile robot navigation exploration algorithm. *IEEE Transactions of Robotics and Automation*, 8(6):707–717, December 1992. `http://users.rsise.anu.edu.au/~rsl/rsl_papers/IEEE.ps`.

A

Experimental planning tasks

The experiments described in section 2.5 are based on seven different planning tasks. The following figures show the maps with obstacles (black) on which the planing tasks were performed as well as the specified starting and ending location. The path (blue) shown between these locations is only for illustration. In every experiment the generated path is a different one, because the parameters of the planning algorithm were changed.

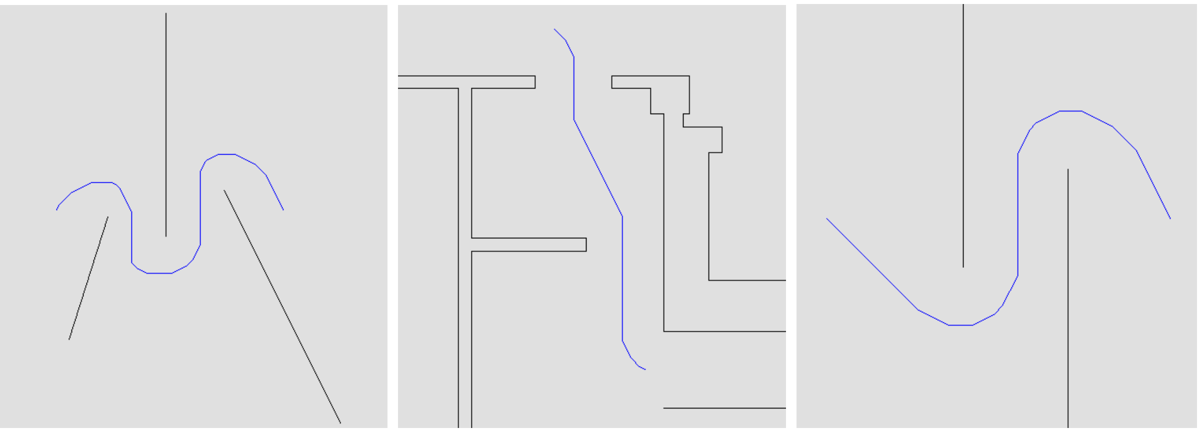

Figure A.1.: Planning tasks 1, 2 and 3

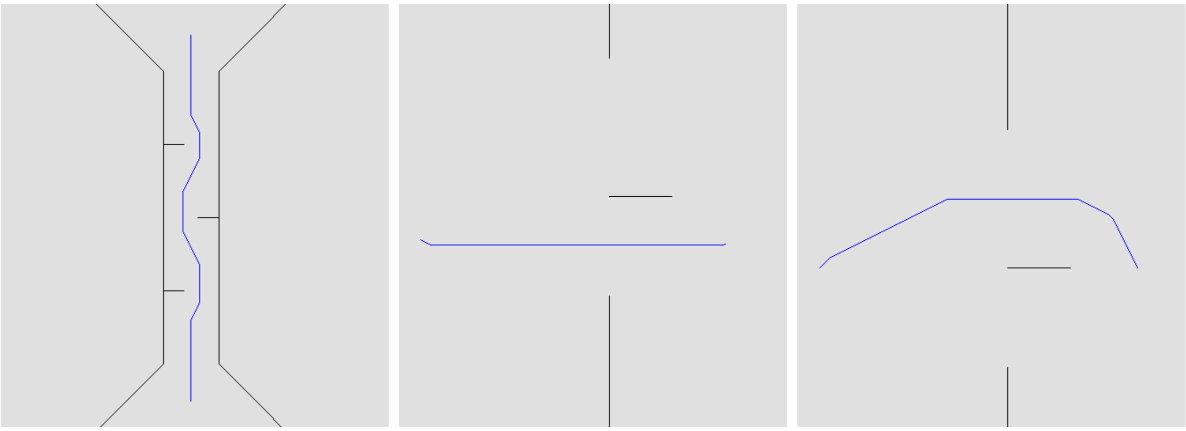

Figure A.2.: Planning tasks 4, 5 and 6

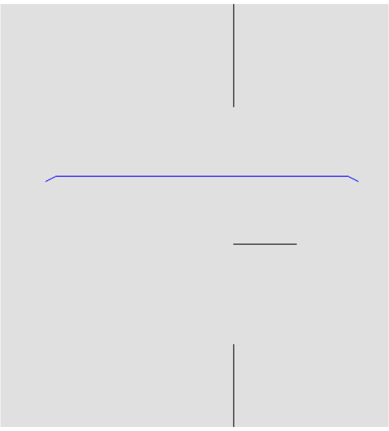

Figure A.3.: Planning task 7

B
Video recording

A video recording showing the navigation system in practice is available for download at the webserver of the TAMS research group (`http://tams-www.informatik.uni-hamburg.de/research/robotics/service_robot/videos/index.php`). It shows the robot navigating through the rooms of the TAMS laboratory. The system bases its planning on a static map of the rooms as well as on data from the laser range finders. The stereo cameras have not been utilized in this recording. Below a few comments to the scenes on the recording will be stated. They are tagged with the timestamps of the discussed scenes.

B.1. Scene 1 - The storage room

0:13 - 0:33 The graphical control interface of the navigation system is shown. It displays the current environment representation (shown as red lines), the position and the orientation of the robot (blue square), and the planned movement (purple line). Currently, the robot is ordered to move to the hallway. As the main doors of the room are blocked, the robot has to move through the cluttered storage room.

0:34 - 1:14 The robot is shown as it moves to the doorway leading to the storage room.

1:15 - 2:41 The robot moves through the doorway and crosses the storage room.

2:42 - 4:39 The robot exits the storage room and then moves down the hallway.

B.2. Scene 2 - Passing a small passage

4:51 - 5:26 The robot passes a narrow doorway with not much free space left to maneuver. The system is actually capable to move the robot through passages with even slightly less free space. However, the passage needs to be a few centimeters wider than the robot. Otherwise the planning software will try to find a way around the passage or will terming movement execution.

B.3. Scene 3 - Obstacle avoidance

5:36 - 6:24 This scene shows the robot moving towards its destination, when a person steps into its way. This forces the robot to terminate its movement and plan a different route.

B.4. Scene 4 - Dynamic pathplanning

6:36 - 6:45 The robot is instructed to move into the storage room, similarly like in scene 1. This time, however, the door to the storage room is closed. The robot cannot realize this from its initial position.

6:46 - 7:51 The robot approaches the closed door. When the data from the laser range finder (shown on the right hand screen) indicates that the door is closed, it turns around and tries to enter the storage room from the other doorway.

7:53 - 9:44 Here the robot is ordered to move back into the lab. Like before, the doorway will be blocked by a person, forcing the robot to take the other doorway.

B.5. Scene 4 - Dynamic obstacle avoidance in a cluttered environment

9:57 - 14:22 Many different obstacles are put into the robot's way. It navigates around them and adopts a new plan when the obstacles are moved. The swivel chair is of special interest: The laser range finders can sense only its central stand, thus its outline in the environment representation of planning software is much smaller than it should be. Therefore the robot comes very close to the chair and even touches it slightly (at 11:55 and 12:47). For this kind of situation the stereo camera approach was intended to detect the real size of the obstacle.

Erklärung / Statement C

C.1. Deutsch

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

David Melnychuk

C.2. English

I confirm that I made the preceding work independently and without foreign help, and that I did not use other resources than stated in the bibliography. All parts that have been taken from publications, literally or by meaning, have been marked as such.

I agree that this work will be added to the inventory of the department's library.

David Melnychuk