KUNGL
TEKNISKA
HÖGSKOLAN

Royal Institute of Technology
Dept. of Numerical Analysis and Computer Science

Digital elevation mapping using stereoscopic vision

by
Ivo Ihrke

TRITA-NA-E Nr. here

NADA
Nada (Numerisk analys och datalogi)
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, SWEDEN

Digital elevation mapping using stereoscopic vision

by
Ivo Ihrke

TRITA-NA-E Nr. here

Master's Thesis in Computer Science (20 credits)
at the School of KTH,
Royal Institute of Technology year 2001
Supervisor at Nada was Henrik I. Christensen
Examiner was Axel Ruhe

Abstract

The purpose of this thesis is, to create a digital elevation map of the local surrounding of a robot using stereo vision. The aim is to provide means for obstacle avoidance and local path planing.
For navigation in an outdoor setting it is necessary to have facilities for automatic mapping of the terrain and detection of obstacles. The outdoor domain is characterized by less structure than an in-door environment and the problem of mapping is thus more challenging. In this project the aim is to develop basic methods for terrain mapping. Vision is by far the most flexible sensory modality for sensing the environment. Through use of stereo mapping it is in principle possible to create a map of the environment, that is dense and contains most of the relevant features of a scene. The map building should be done in real-time to enable the robot to find its way in unknown environments. The mapping algorithm must be robust, because the robot is intended for outdoor use in rough environments. Thus few restrictions of the environment should be assumed, as opposed to indoor environments, that are quite regular. A sub-task will be position estimation of the robot in order to be able to fit a partial map obtained in every timestep into a bigger one that represents the nearer surrounding of the robot while the robot is moving. Sensory data of the robot can be used to verify position estimates and if necessary (due to mathematical ambiguities) guide the estimation. Because of memory restrictions and the SLAM-problem (Simultaneous Localization And Mapping), the map is intended to maintain the nearer environment only.
It is intended to do the mapping as far as possible from camera pictures alone, which leads to flexibility in the robot platform used. How far this is possible in real-time will be shown by experiments.

Sammanfattning

Syftet med det här examensarbetet är att utveckla en metod med hjälp av vilken en robot kan kartlägga sin närmaste omgivning med hjälp av datorseende. Syftet är mer specifikt att utveckla ett hjälpmedel för roboten när den ska planera sin färdväg samt undvika hinder. Synen är människans viktigaste sinne. Vi navigerar i vår omgivning huvudsakligen genom att se och undvika hinder. Det skulle därför vara mycket önskvärt för en robot att ha ett liknande sätt att uppfatta och föra sig i världen. Navigation i en utomhusmiljö ställer speciella krav på en algoritm avsedd för att kartlägga denna miljö. Ytor i en utomhusmiljö är inte lika strukturerade som i en inomhusmiljö, och situationer är generellt sett mindre förutsägbara i den förra miljön jämfört med den senare. På grund av detta måste robusta metoder användas och speciell vikt läggas vid att göra algoritmen okänslig för variationer i omgivningen. För att den utvecklade metoden skall vara användbar måste roboten kunna utföra kartläggningen av omgivningen i realtid. Kartläggningen är planerad att ske endast med hjälp av kamerabilder, det skall dock undersökas om detta är tillräckligt eller om andra sensorer måste integreras.

Contents

List of Figures

Notation

$\mathbf{x}$	euclidean vector
x, l	entity in 2D homogeneous coordinates (points, lines)
X, L, Π	entity in 3D homogeneous coordinates (points, lines, planes)
x_i, X_i	element of vector x or X
x^i, X^i	different vectors of the same kind
$X^{(i)}$	entity in or referring to a local coordinate system
$\mathbf{M}$	matrix
$\mathbf{M}^T$	transposed matrix
$\mathbf{M}^{-1}$	inverse matrix
$\mathbf{M}^{-T}$	inverse of the transposed matrix
$\mathbf{M}^+$	pseudo-inverse matrix
$[\mathbf{A} \mid \mathbf{B}]$	block matrix
$[\mathbf{t}]_\times$	skew symmetric matrix, representing the cross product with vector $\mathbf{t}$
$\overline{x}$	mean value
$\lvert\mathbf{x}\rvert$	absolute value $\sqrt{\mathbf{x}^T\mathbf{x}}$ of $\mathbf{x}$

Chapter 1

Introduction

1.1 Introduction

This report presents the results of a study into methods of stereoscopic vision. The availability of cheap CCD cameras makes video an increasingly available resource for robotic sensing.
Vision is a very flexible sense. Human beings rely mostly on that input when dealing with the world. We identify obstacles mainly by just "seeing" them. Such natural sensing would be very desirable for a robot to have. This project aims at creating a map of the nearer surrounding of a robot to equip it with a tool for obstacle avoidance and local path planing.
Video data as a source for map building is superior to other kinds of sensing devices like laser scanners or ultrasonic sensors because it delivers a very dense set of data relatively fast. Here lies also one of the challenging factors in the project. The data rate is very high and therefore effective, robust algorithms have to be used. The other one is the intended outdoor use of the robot. That means it must handle an unstructured environment, with unpredictable situations.
The research presented in this thesis is in finding ways to use the pictures from an stereo video stream to form a digital elevation map of the environment the robot is acting in. It is aimed at the reconstruction of a three-dimensional path of the robot, using this position estimate to transform local maps obtained for every image pair of the sequence into a common coordinate system. This allows for a combination of the local maps to form a global map that covers the surrounding environment that was previously seen by the robot.

1.2 Motivation

This research is motivated by the Center of Autonomous Systems at the Royal Institute of Technology. One of the ongoing projects is concerned with

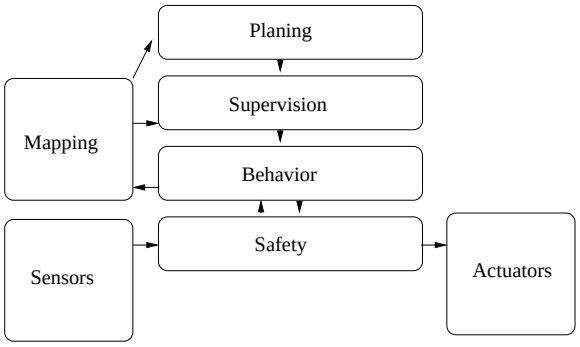

Figure 1.1. Components of a typical robot system

an autonomous robot for outdoor use. It deals with aspects of navigation in an unknown, rough outdoor environment and current research aims at the development of basic navigation and sensing methods that allow for higher level planing and navigation. The robot platform is thought to be used in hostile environments to provide information about the area without the necessity of human presence. Figure 1.1 shows the typical components of a robot system. The robot usually has a number of sensors and a number of actuators. These are connected by a low-level safety module. This module stops the robot if a dangerous situation occurs. On the behavioral level simple actions like explore, follow, go to etc. can be performed, but all sensor and actuator information passes through the safety module. The supervision part controls the actions of the behavioral part. This part of the system and the planning module need a higher level of world representation than raw sensor data. The mapping process provides this information by fusing the data of different sensors into a common map. This project concentrates on stereo vision as main sensor input to form a map of the local surrounding of the robot. It is intended as an aid for the supervision module to help in local navigation, like obstacle avoidance and accessibility of the surrounding area. The aim is to provide a local map for the robot in real-time. Of course the speed of movement of the robot can be adjusted to meet real-time requirements, but it is aimed at an acceptable speed of movement (about 1 m/s). Furthermore all environmental objects that could constitute a danger for the robot should be recognizable in the map, to provide a reliable basis for future path planing routines. The bumpers of the robot are mounted approximately 25 cm above the ground and sonar sensors are mounted right above it. That means that obstacles of a height bigger than 30-40 cm can be detected by the safety module. Other obstacles like holes and objects smaller than that height should instead be avoided by a local path planning procedure in the supervision module. This project tries to provide means for that task.

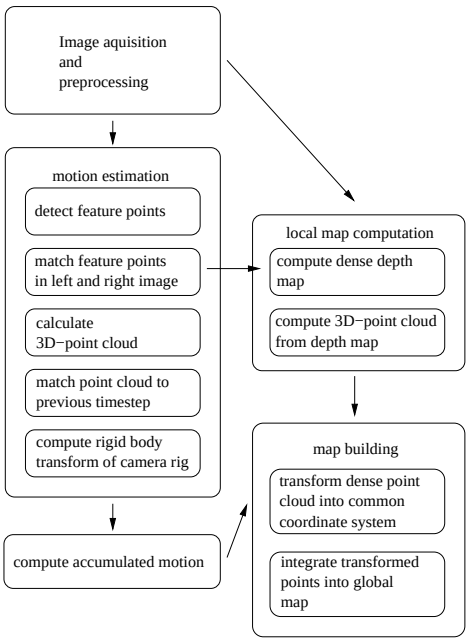

Figure 1.2. Outline of mapping algorithm

1.3 Approach

Figure 1.2 shows the outline of the algorithm developed in this thesis. The first sub-task is the acquisition and preprocessing of stereo images. These build the basis for all further computations. The preprocessed stereo images are passed to the motion estimation and the local map computation module. The motion estimation is based on 3D- to 3D-point correspondences that relate two successive frames of the stereo sequence. To find these correspondences first feature point detection has to be performed in both, the left and the right, images followed by a matching step to relate the computed feature points in space. This step yields a median displacement of corresponding feature points between the images. The median displacement is passed to the local map computation module, giving a hint for the adjustment of the search range in the computation of the depth map. The left-right correspondences between feature points in the two images are used to compute a sparse set of 3D-points, that in turn are related to the set of 3D-points obtained in the previous timestep. This step yields correspondences between 3D-points in different instances of time. These correspondences are used to compute the rigid body transformation that describes the movement of the camera rig during the time that passed between the recording of the two stereo images. The resulting transformation relates 3D-points in the current camera coordinate system to that of the previous timestep. Accumulating

these transformations for each new timestep makes it possible to transform 3D-points given in a local coordinate system in any of the camera positions visited so far, into a common coordinate system. The points that are to be transformed into the common coordinate system are computed in the local map computation module. A dense 3D-point cloud is computed there. The dense point cloud together with the transformation relating it to the common coordinate system are passed to the map building module, where the transformation is carried out and the transformed points are integrated into the final map.
The thesis is arranged in the following way. Chapter 2 gives an overview over methods for feature detection, matching and depth map computation as well as some background information for stereo geometry. Image acquisition, preprocessing and local map computation are covered in chapter 3. Some decisions concerning the hardware setup can be found there as well. Chapter 4 introduces the octree as a data structure for map representation. The octree is used to manage a three-dimensional occupancy grid of the environment. As mentioned before the motion of the camera rig has to be estimated in order to determine the transformations that relate point sets from different timesteps to a common coordinate system. A robust method for that task is described in chapter 5. The algorithm is then evaluated in chapter 6. Restrictions for an algorithm estimating motion from stereo information are discussed. Finally conclusions and suggestions for future work are presented in chapter 7.

Chapter 2

Stereo Vision Basics

2.1 Introduction

This chapter presents some background material that was necessary to develop the ideas in this thesis. It covers the main steps from 2D stereo pictures to 3D points representing physical features in the world, as well as the relationship between two different images of the same scene.
The basic principle of stereo vision is triangulation. Since vision is a passive sensing technique, the three points needed for the triangulation, on which the reconstruction of a 3D point is based, are a point in the physical world together with its pictures seen from two different positions in space. If the internal parameters[1] and the external parameters[2] of the cameras are known it is easier to reconstruct the three-dimensional structure. This is called a *calibrated* setup. The process of estimating the internal and external parameters of a camera/camera rig is termed *calibration*. It is also possible up to a certain level of ambiguity to reconstruct a scene without prior knowledge of the internal and external parameters of the cameras. This approach is therefore called *uncalibrated*. Both approaches have some steps in common [7]. These are:

1. feature extraction (Section 2.3)
2. feature matching (Section 2.4)
3. depth reconstruction (Section 2.2.1)

The calibrated and the uncalibrated approach differ mainly in how those steps are performed. In the calibrated case the so called *fundamental matrix*

[1] for example: focal length, center of projection, pixel-size etc. depending on the camera model

[2] translation of the center of projection and rotation of the optical axis between two images of the same scene

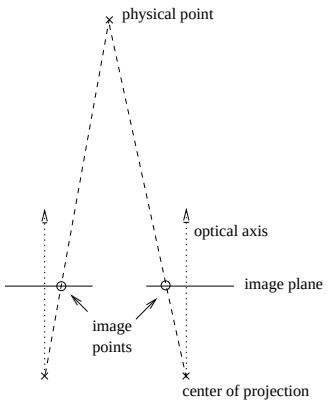

Figure 2.1. Parallel camera setup

(Section 2.2.3), that describes the relationship between two 2D pictures of one 3D scene, is known beforehand, whereas it has to be computed in the uncalibrated case.

A simple camera setup with two parallel cameras as shown in Figure 2.1 is used. For this setup it is especially simple and fast to perform feature matching between the left and the right stereo image as will be shown in section 2.2.4. The search performed in step 2 reduces to a search along corresponding scanlines of both pictures, thus providing a simple search space as well as a good access pattern in computer memory. This will be used to compute a dense *disparity map*. A disparity map is a buffer, containing the differences in position of corresponding pixels. This corresponds to the optical flow in video sequence analysis [35]. In our case it is sufficient to store disparities in the horizontal direction, because the vertical disparity is zero for the parallel camera setup. An overview of some techniques and problems in disparity computation is given in section 2.5.

2.2 Stereo geometry basics

The discussion of stereo geometry is restricted to two ideal pinhole cameras, which will be shown to be sufficient without loss of generality. The projected points on the respective camera planes can result from images taken by two cameras at the same time or from images taken by the same camera at different times. This section is based on the book “Multiple View Geometry” by Hartley and Zisserman [13]. The content is presented in reverse order to the implementation of a stereo reconstruction program.

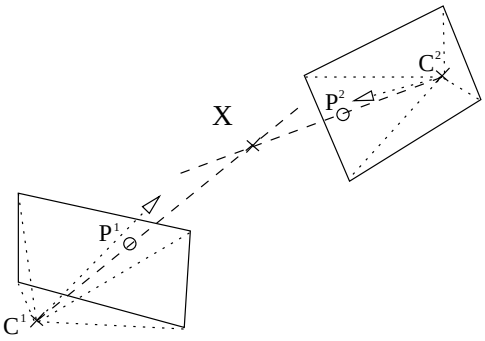

Figure 2.2. Depth reconstruction in general camera position

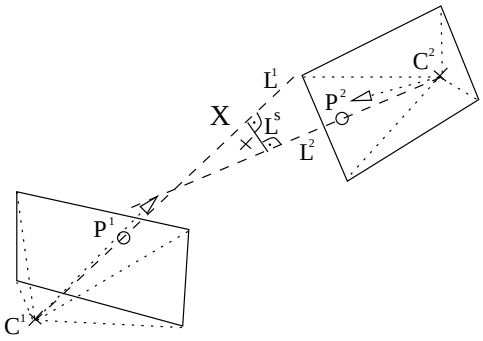

Figure 2.3. Depth reconstruction with errors present

2.2.1 Depth reconstruction

Assuming the positions of the image points ($\mathbf{P}^1$,$\mathbf{P}^2$) of a point ($\mathbf{X}$) in 3D-space and the positions of the projective centers of the cameras ($\mathbf{C}^1$,$\mathbf{C}^2$) that took the images are known with perfect accuracy, it is no problem to do a triangulation to actually compute the unknown coordinates of point $\mathbf{X}$: Solve $\mathbf{C}^1+s(\mathbf{P}^1-\mathbf{C}^1)=\mathbf{C}^2+t(\mathbf{P}^2-\mathbf{C}^2)$ for s and t and insert in one side of the equation to obtain $\mathbf{X}$. There exists a unique solution for s and t as long as the centers of projection are not the same point ($\mathbf{C}^1 \neq \mathbf{C}^2$). Otherwise the lines of projection coincide and it is impossible to reconstruct depth. This leads to problems with a purely rotational motion. The case of no solution can not occur because the point being viewed has finite coordinates. Only if the point lay at infinity, the rays of projection would be parallel leading to no solution.

In real applications the assumption of accuracy is of course incorrect. The first source of error is the resolution of the camera. A pixel covers an area and is not a perfect point. Even if sub-pixel accurate feature detection is used the result will never be the real projected point. The second one is the difference in position and orientation of the cameras. They are erroneous as well. Thus normally the reprojected rays will not intersect (Figure 2.3). The simplest way to deal with this inaccuracy is to find the shortest distance $\mathbf{L}^s$ between the reprojection lines $\mathbf{L}^1$ and $\mathbf{L}^2$ (the *geometric error*) and take the middle-point of $\mathbf{L}^s$ as the approximation of point $\mathbf{X}$. Nevertheless this is not the optimal way to go since the error was not introduced in three-dimensional space but in the image plane.
Another approach is therefore to find a transformation that makes it possible to transfer image points between the image plane of camera 2 and the image plane of camera 1. After that $\mathbf{P}^2$ is transformed into image plane 1, yielding $\tilde{\mathbf{P}}^2$and $\mathbf{P}^1$ into image plane 2 leading to $\tilde{\mathbf{P}}^1$. In the ideal case $\mathbf{P}^1$ should coincide with $\tilde{\mathbf{P}}^2$ and $\mathbf{P}^2$ with $\tilde{\mathbf{P}}^1$, but this is very unlikely to happen in

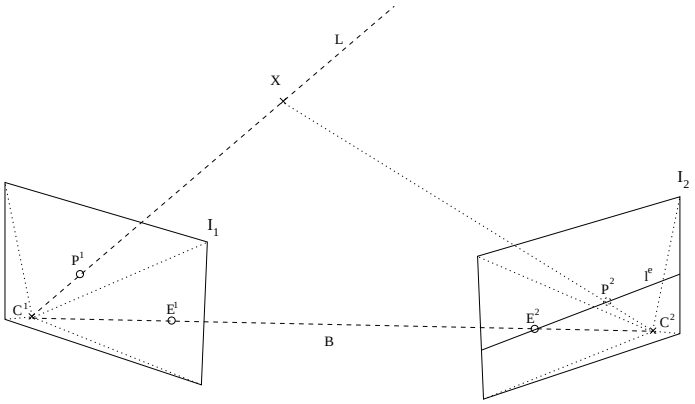

Figure 2.4. Point correspondence geometry

reality. The resulting error in the image planes is called the *reprojection error*. Now the middle-points of line segment ($\mathbf{P}^1$ - $\tilde{\mathbf{P}}^2$) and line segment ($\mathbf{P}^2$ - $\tilde{\mathbf{P}}^1$) can be back-projected and the middle-point of the shortest distance is taken as approximation of point $\mathbf{X}$ as before. The transformation that is needed to do that is the fundamental matrix that is described in section 2.2.3. It is a mathematical tool to describe the epipolar geometry which is covered in the next section.

2.2.2 Epipolar geometry

As was mentioned in the last section, the epipolar geometry is the geometric relation, that links two different views of one scene together. It is used to restrict the search space for corresponding points when matching feature points. The epipolar relationship is basically a geometric reasoning, that shows, that points in one image plane relate to lines in the other one and the other way around.

The basic layout of the problem is shown in Figure 2.4. From now on it is assumed, that point $\mathbf{P}^1$ in the image plane of camera $\mathbf{C}^1$ is given and the position of $\mathbf{X}$ and $\mathbf{P}^2$ are unknown. It is intended to find a point $\mathbf{P}^2$ in the image plane of camera $\mathbf{C}^2$ such that it is the projection of the same point $\mathbf{X}$ onto I_2 as is $\mathbf{P}^1$ on I_1. Since $\mathbf{X}$ could lie anywhere on line $\mathbf{L}$ the search space is restricted to the projection of $\mathbf{L}$ onto I_2. The projection of line $\mathbf{L}$ (l^e) is called the *epipolar line* for $\mathbf{P}^1$. Because all reprojected lines $\mathbf{L}$ pass through the center of projection $\mathbf{C}^1$, all epipolar lines intersect in one point ($\mathbf{E}^2$). This point is called the *epipole* and is the projection of camera center $\mathbf{C}^1$ onto image plane I_2. The line $\mathbf{B}$ joining the two camera centers is called the *baseline*. All points on the baseline project to the epipoles and their 3D position can not be reconstructed, since the triangulation degenerates.
Since the positions of $\mathbf{C}^1$ and $\mathbf{C}^2$ are interchangeable, line ($\mathbf{P}^1$-$\mathbf{E}^1$) is an

epipolar line for $\mathbf{P}^2$ as well. Moreover because $\mathbf{P}^2$ could lie anywhere on the epipolar line l^e, line ($\mathbf{P}^1$-$\mathbf{E}^1$) is an epipolar line for all image points on l^e. Line ($\mathbf{P}^1$-$\mathbf{E}^1$) and l^e are *corresponding epipolar lines*. They span the *epipolar plane*. All 3D points in this plane project to these two lines in both images respectively. All epipolar planes contain the baseline.

2.2.3 The fundamental matrix

It is convenient to have a mathematical description of this relation. This is done in terms of a rank deficient linear transform of the image points in homogeneous[3] 2D coordinates. When internal camera parameters are left aside (or normalized) it is described by the *essential matrix*. When normalization of the internal camera parameters is included in the matrix it is called the *fundamental matrix*.

Projective geometry

Projecting a point $\mathbf{X}$ in world coordinates onto an image plane of a camera with center of projection $\mathbf{C}$ is done by calculating the point of intersection of the line connecting $\mathbf{C}$ and $\mathbf{X}$ with the image plane. Assuming the camera being a pinhole camera (focal length 1), center $\mathbf{C}$ to be at $(0,0,0)^T$ and viewing direction along the z-axis, the projection is especially simple. As can be verified by simple triangulation, the coordinates of the projected point are $(\frac{\mathbf{X_1}}{\mathbf{X_3}}, \frac{\mathbf{X_2}}{\mathbf{X_3}})^T$ and the projection can be expressed in homogeneous coordinates by:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix}$$

The matrix in this equation is called *camera projection matrix* and will be referred to by $\mathbf{P}$. The camera projection matrix can have different entries, accounting for non-normalized camera geometries[4] denoted by $\mathbf{K}$ or a camera in a general position ($\mathbf{M} = [\mathbf{R} \mid T]$[5]). This will be written $\mathbf{P} = \mathbf{KM}$. A projection is then expressed as $x = \mathbf{P}X$.

Fundamental matrix

The following derivation of the fundamental matrix is taken from Xu and Zhang [38]. A point X is projected onto the image plane of a camera with

[3] See the appendix for a review of homogeneous coordinates

[4] See the appendix for more sophisticated camera models

[5] $\mathbf{R}$ is the rotation, T the translation of the camera from world origin, transformation $\mathbf{M}$ moves the camera to the world origin with viewing direction along the z-axis

$x = \mathbf{P}X$. If this equation is solved for X using the pseudoinverse of $\mathbf{P}$, the line back-projecting point x is given by $\mathbf{P}^+x$. This has a one parametric solution since $\mathbf{P}$ is a 3×4 matrix and is therefore a line. One point on this line which is different from the camera center is chosen and projected onto the image plane of the second camera with projection matrix $\hat{\mathbf{P}}$. The camera center of the first camera is projected to the second cameras image plane as well. The line joining the two points is the epipolar line. Algebraically:

$$\begin{aligned}\hat{x} &= \hat{\mathbf{P}}\mathbf{P}^+x \\ \hat{e} &= \hat{\mathbf{P}}C \\ \hat{l^e} &= \hat{e} \times \hat{x} = \hat{e} \times \hat{\mathbf{P}}\mathbf{P}^+x = [\hat{e}]_\times \hat{\mathbf{P}}\mathbf{P}^+x\end{aligned}$$

The fundamental matrix is given by $\mathbf{F} = [\hat{e}]_\times \hat{\mathbf{P}}\mathbf{P}^+$ and the epipolar relationship can be written $\hat{l^e} = \mathbf{F}x$. Given this expression the relation

$$\hat{x}^T\mathbf{F}x = 0$$

between two corresponding points in two pictures can be stated[6]. This condition which is true for any pair of corresponding points $x \leftrightarrow \hat{x}$ is used for the computation of matrix $\mathbf{F}$ from image correspondences alone. $\mathbf{F}$ contains all information about the two projection matrices and therefore all internal camera parameters of the two cameras as well as their relative position and viewing direction.

2.2.4 Special case - fixed parallel camera rig

The fixed parallel camera rig is a very special situation, that simplifies computations significantly. Consider a normalized camera model, the first camera at the world origin with $\mathbf{P} = [\mathbf{I}|0]$ and the second camera translated along the x-axis, same viewing direction. The camera projection matrix of the second camera is then given by $\hat{\mathbf{P}} = [\mathbf{I}|\mathbf{T}]$ where $\mathbf{T} = (\mathbf{T_1}, 0, 0)^T$. The epipole is in this case situated at infinity on the x-axis $\hat{e} = (1, 0, 0)^T$.

$$\mathbf{F} = [\hat{e}]_\times \mathbf{I} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The condition $\hat{x}^T\mathbf{F}x = 0$ reduces to $\hat{x}_2 = x_2$. That means corresponding points have to lie on corresponding scanlines. This fact will be exploited in the computation of disparities. The only problem is that the images must be in normalized camera coordinates. Therefore the cameras have to be calibrated and the calibration data has to be used to *rectify* the images. Rectification is the process of normalizing camera coordinates. The calibration of the camera rig used in this project is described in section 3.5.

[6] $\hat{x}$ lies on the epipolar line $\hat{l^e}$ and therefore $\hat{x}^T\hat{l^e} = \hat{x}^T\mathbf{F}x = 0$.

2.3 Feature extraction

Up to now it was assumed that matching image points are known. Since not all areas in the image are equally useful to obtain this information (i.e. areas of constant intensity are not suitable for establishing correspondences) *feature extraction* has to be performed in both images. The second step is then to establish correspondences between the extracted features. This is referred to as *feature matching* and is discussed in section 2.4.
This section gives some information on available feature extraction methods. It is concerned with *feature points* since points are projectively invariant. Lines such as edges or ridges or even surfaces behave differently under projection. They can not be reconstructed unambiguously from two views alone. Another advantage of points is the high localization of the feature.
The task of feature extraction has been studied thoroughly since it is one of the most basic, but nevertheless challenging tasks in computer vision. One of the biggest problems is the repeatability of point detection in different images of the same scene. This is a necessity for *feature tracking* and feature matching.
There are many approaches to feature extraction. The main classes are template based, contour based and intensity based methods [31]. In template matching an image is convoluted with a number of predefined template masks. The strongest responses are caused by features. An example is given in [40]. Template matching is based on biological vision that works in a similar way. Contour based methods seek to find contours in the image. They are mostly approximated by line-segments or splines, which are checked for intersections. Such intersections mark feature points. Intensity based approaches utilize the observation that the surrounding area of feature points (i.e. points with a high information content) imposes strong variations in image intensity. Most methods in this class use a combination of directional derivatives[7] or the autocorrelation function. Derivative based approaches are quite sensitive to noise. For that reason the filters used for computation of the derivatives are often convoluted with a Gaussian kernel to introduce some presmoothing to the image.
In this project an intensity based detector was used mainly because methods of this type are the most commonly used and their exist studies comparing the performance of different methods [31] [30] [15]. As a result of [31] it can be pointed out, that the widely used Harris detector is a good choice to detect features reliably and fast.
The Harris detector is based on the autocorrelation function:

$$f(x,y) = \sum_{(x_k,y_k)\in W} (I(x_k,y_k) - I(x_k+\Delta x, y_k+\Delta y))^2 \qquad (2.1)$$

[7] for example $K = \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_xI_y}{I_x^2+I_y^2}$ [22]

(W is the window centered at (x, y) that is used for computation, I the image intensity function). Equation 2.1 is a measure of dissimilarity, that means the higher the values the higher the change in direction $(\Delta x, \Delta y)$. That means if f gives high values in one direction and low ones in another there is an edge in the image, if it is high in all directions a corner is present. This motivates integration over all directions. The computational expense of integration can be avoided by using the auto-correlation matrix.
Taylor expansion of the image intensity function gives:

$$I(x+\Delta x, y+\Delta y) = I(x,y) + \begin{pmatrix} I_x(x,y) & I_y(x,y) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} + O(\Delta x^2, \Delta x^2) \tag{2.2}$$

Substituting 2.2 into 2.1 leads to:

$$\begin{aligned} f(x,y) &\approx \sum_{(x_k,y_k)\in W} \left(\begin{pmatrix} I_x(x,y) & I_y(x,y) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right)^2 \\ &= \left\{ \mathbf{d} = \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \right\} = \\ &\quad \mathbf{d} \begin{bmatrix} \sum_{(x_k,y_k)\in W} I_x(x_k,y_k)^2 & \sum_{(x_k,y_k)\in W} I_x(x_k,y_k) I_y(x_k,y_k) \\ \sum_{(x_k,y_k)\in W} I_y(x_k,y_k) I_x(x_k,y_k) & \sum_{(x_k,y_k)\in W} I_y(x_k,y_k)^2 \end{bmatrix} \mathbf{d}^T \\ &= \mathbf{d}\mathbf{A}(x,y)\mathbf{d}^T \end{aligned} \tag{2.3}$$

This shows that the structure of the local neighborhood of (x, y) is essentially contained in the 2×2 matrix $\mathbf{A}$. The eigenvalues of $\mathbf{A}$ reflect the change of the intensity function in the two main directions. If both are small, the change in intensity is small and therefore the structure in the image is low. If one eigenvalue is dominant there is an edge in the direction of the corresponding eigenvector in the image. If both eigenvalues are big a corner has been detected. Matrix $\mathbf{A}$ is rotational invariant. Harris uses $\det(\mathbf{A}) - \alpha \mathrm{trace}(\mathbf{A})^2$ as cornerness measure to avoid the calculation of eigenvalues and collect the results in one number.
Another detector that uses the same result is the Good Features to Track operator [33]. It is implemented in the *OpenCV* library [2] and was used throughout this thesis, because the results for the Harris detector should apply to it as well. To obtain sub-pixel accuracy for those detectors it is possible to fit a parametric surface (usually a polynomial) to the detected pixels neighborhood of autocorrelation values and compute the maximum.

2.4 Feature matching

Feature matching is the process of automatically assigning correct correspondences between feature points in two images. It is commonly referred to as

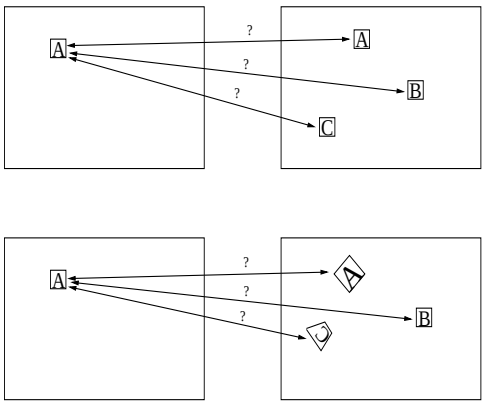

Figure 2.5. Stereo correspondence problem, in the case of a simple translation between images (upper) and with affine distortions present (lower)

the *correspondence problem*. Common practice is, to compare the intensity neighborhoods of candidate matches, assigning each pair a similarity score, based on some similarity measure[8]. The pair with the highest score is selected to be a match. Problems occur when a number of feature points have similar intensity neighborhoods. Often a pure translation of the neighborhood between the images is assumed. This is usually true for small camera motions in video sequences. There are critical motions such as rotation about the optical axis that violate this assumption. Also those assignments are not bijective. That means a feature point in one image can be assigned zero, one or more than one match, which does not reflect world properties, that are that a point has a unique position in world space. There are certain situations in which multiple images of one world point can occur, but this is due to reflections or refractions, which is of minor concern in most situations. Due to this fact and randomly similar neighborhoods, feature matching is an ambiguous process. Mismatches occur quite frequently and measures have to be taken to reduce the number of mismatched points i.e. *outliers*. Often the search area for corresponding feature points is restricted to a certain region around the position of the feature point in question. This reflects the assumption of restricted movement between frames and helps cutting down computational expenses.
To reduce the number of outliers a uniqueness constraint i.e. a matching phase from image 1 to image 2 followed by a matching of image 2 to image 1, keeping only the consistent matches is often enforced. This is called *left-right check*. Another constraint that can be applied is that the "median flow" in an area should be approximately the same [35]. That means movements of feature points in a region should have approximately the same direction, or

[8] for some popular similarity measures see the appendix

Figure 2.6. Stereo image

Figure 2.7. Corresponding disparity image

if the movement is small comparable lengths.
If the assumption of translational behavior of the intensity neighborhoods is wrong, more sophisticated models have to be found. The approach used here is to use affine transformations [33, 4, 29, 18, 26] i.e. linear warping and translation as a model for the changes (see figure 2.5). This accounts for scaling, rotation and translation. A patch around the feature point in one image is warped such that it fits the intensity neighborhood in the second image best. The best fit is again determined by a similarity score like cross-correlation. These techniques are known as *wide baseline stereo*, and needed because the geometric invariance of simple cross-correlation is not sufficient to cover perspective distortions occurring when using cameras with widely differing views.

2.5 Disparity computation

The aim of computing a disparity map is to obtain a dense depth map as in figure 2.7 from stereo images like in figure 2.6. When building a map it is obviously advantageous to have dense depth information. With the methods described up to now only sparse depth structure can be computed. There exists a lot of literature on the computation of disparity maps and many different methods have been developed. The following discussion assumes parallel horizontal epipolar lines for simplicity.

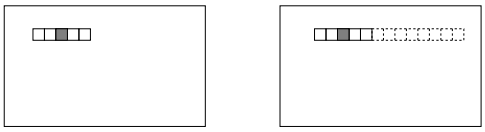

Figure 2.8. Pixel-wise disparity computation in a parallel camera setup

The approaches can be divided in two groups i.e. global approaches that compute the whole disparity map or at least whole scanlines at once and local ones, that compute disparities pixel-wise. The global methods are often based on energy minimization [36]. A function like $\sum_p \min_d(\text{cost}(p,d))$ is minimized and gives a discrete function $D(p)$ representing the sought disparities. The cost function is high for unlikely matches and vice versa. It represents constraints like surface smoothness and the *ordering constraint*[9] A simple cost function is for example $\text{cost}(p,d) = [I(p) - I'(p+d)]^2$ the squared difference of intensities. I and I' are the intensity functions of the two pictures respectively, p is the pixel to consider and d the disparity value that is checked. The values of $D(p)$ are often determined by dynamic programming techniques. An example for this approach can be found in [6].Another formulation of this principle is to state it as a maximum flow problem [27]. Here the cost for matching a specific pair of pixels defines a capacity for a flow from one image to the other. One image is connected to a sink and the other one to a source. Between the images a 3D system of pipes is created, linking neighboring pixels as well as neighboring disparities to each other (6-connected). The maximum flow from the source image to the destination image saturates now the pipes with the lowest capacities i.e. the lowest matching costs. The resulting minimal cut surface (the surface containing all saturated edges) is the sought disparity image. It is smooth since the nodes in the graph are 6-connected and satisfies a global surface smoothness constraint and gives also values for occluded areas. Unfortunately though giving better results than local methods the global methods suffer from computational expense and are thus not suitable for real-time computation.

Pixel-wise correlation based stereo is the simplest and therefore the fastest approach to disparity computation. In this case a parallel camera setup is usually used because corresponding epipolar lines are corresponding scanlines, making an efficient computation possible. The disadvantage of pixel-wise disparity computation is the missing connection to the neighboring results. Pixels are assigned totally independent disparities, thus neglecting additional information that could be used using assumptions like the local ordering constraint or Marr and Poggio's basic rules for match-

[9]The ordering constraint states that pixels appear in the same order in both images (e.g. left to right)

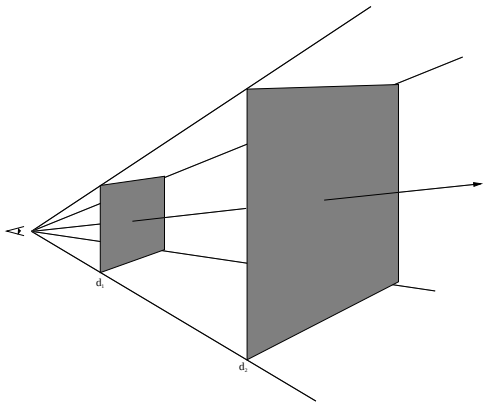

Figure 2.9. Planes of equal disparity

ing left - right images, namely uniqueness (every point can be assigned to one and only one point in the other image) and continuity (disparities vary smoothly almost everywhere, except at depth discontinuities).
For the actual computation a small window around the pixel for which a disparity is sought is taken and compared to a range of pixels in the second image as shown in figure 2.8. The pair that is most similar is kept. The comparison is carried out using one of the measures of similarity listed in the appendix. It is believed that the zero mean cross correlation performs best. This should be true under largely varying illumination conditions, under practical conditions however the sum of absolute differences performs quite well [36] [21] and can be computed much faster. For example on Intel Pentium III systems there exists a machine instruction for computing the sum of absolute differences for 8 or 16 bytes simultaneously. The Small Vision System [21] was used throughout this thesis to compute disparity maps. It uses sum of absolute differences as well. The range of pixels that the original pixel is compared with defines the interval of representable depth values.

Figure 2.9 shows planes of equal disparity. Let d_1 represents the maximum and d_2 the minimum disparity. Then the cut pyramid in figure 2.9 visualizes the representable depth values in the disparity range between d_2 and d_1. However since the comparison is done pixel-wise the disparities are discrete and therefore the reconstructed 3D-points all lie on planes between the two extremal planes [10]. Points situated outside the cut pyramid can not be reconstructed correctly. They give rise to random disparity values, that cause outliers in 3D-point reconstruction. The position of the cut pyramid can be adjusted using an offset, that is the comparison does not start at the corresponding point in the second image, but an offset is added. Thus an

[10]The fact that surfaces of equal disparities are planes is caused by the parallel camera setup and by the planar projection surfaces found in normal cameras (counter example - human vision)

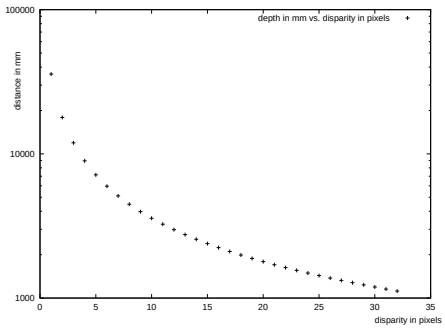

Figure 2.10. Relation depth - disparity for our camera system (213 mm baseline, 6.3 mm focal length)

adaptation can be performed to have as much of the world points as possible in the cut pyramid, reducing the number of outliers in the 3D-data.
Disparities and depths are related inversely. Using a parallel camera setup and centering the global coordinate system at the left camera, the equations for depth reconstruction simplify to:

$$\begin{aligned} \mathbf{X_1} &= \frac{b\mathbf{x_1}}{d} \\ \mathbf{X_2} &= \frac{b\mathbf{x_2}}{d} \\ \mathbf{X_3} &= \frac{bf}{d} \end{aligned} \tag{2.4}$$

where b is the distance between the two camera centers i.e. the length of the baseline, d the disparity (since the $\mathbf{x_2}$ - disparity is zero as was shown in 2.2.4, $d = \mathbf{x_1}^l - \mathbf{x_1}^r$) and f the focal length of the camera. The relationship between depth $\mathbf{X_3}$ and disparity is shown in figure 2.10. This plot shows the characteristics of the system used in this thesis. The baseline is 213 mm and the focal length of both cameras is 6.3 mm. The CCD chip is 6 mm x 4.2 mm and the used resolution is 160 x 120 pixels. Assume 16 disparity levels i.e. the pixel in the left picture is compared to 16 neighboring pixels in the right picture, then it can be seen, that depending on the offset different depth ranges can be covered with different detail. At offset zero the disparities cover a depth range of $\mathbf{X_3} \approx 2400$ mm $-$ inf in a very coarse resolution. With offset 16 the range is much smaller ($\mathbf{X_3} \approx 1100$ mm $- \approx 2250$ mm) but the resolution is now approximately 10 cm instead of meters. This underlines the importance of variable offsets or variable number of disparity levels. Since the Small Vision System [21] supports fixed ranges of disparity only, it was chosen to use variable offsets for the disparity computations.

Chapter 3

Image acquisition and local map computation

3.1 Introduction

Image acquisition is the first task in a computer vision project. It has to be decided, what hardware to use and how to set it up in a way that it fits the needs of the project. The main concern in this project was performance, since the map building has to be done in real-time, if it is supposed to be a navigational aid for a robot that moves quickly in an unknown environment. Therefore everything has to be done to make computations as simple and data transfer as fast as possible. The first point demands a parallel camera setup (see section 2.2.4) and the latter the use of two framegrabbers on the computer.
Gray images were used because color does not give substantial information gain (low textured areas stay low textured) and the data rate is smaller. Another factor that influences both requirements is the resolution of the pictures and the number of disparities that are used. Furthermore it has to be decided, how to mount the stereo camera onto the robot to have an advantageous point of view.
The second aspect covered in this chapter is the computation of a local map from a stereo image, since this is a precondition for global map building. The influence of some parameters like the baseline and the resolution of the images and image preprocessing are discussed as well.

3.2 Camera setup

The aim of the project is to provide a map of the local environment of the robot and through this a help for local obstacle avoidance and path planing. This brings up the question of how to choose and mount the stereo camera rig in order to have a wide area of view with only a small influence of the

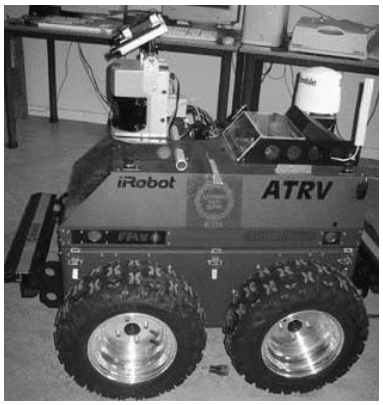

Figure 3.1. Robot system

body motion of the robot. Those are requirements that oppose each other. Wanting a camera that is as stable as possible suggests mounting it very low. On the other hand it should be possible to see the ground directly in front of the robot, allowing the recognition of the most dangerous obstacles, like holes and stones. Furthermore the accuracy of depth computation is better for short distances as shown in section 2.5.
The viewing direction is also of importance. In general it is favorable to have a lot of reoccurring features in two successive images of a video sequence. It is also preferable if those feature points do not move too far, since this introduces additional complexity for stereo matching and new problems like newly visible, formerly occluded areas in the surrounding of feature points, which makes the matching of those more complicated. This is called *Wide Baseline Stereo*. Unfortunately this is what happens when looking at the ground. The objects are quite near to the camera, causing big disparities and large movements in the picture. The alternative of looking straight forward is not promising either. It simplifies feature tracking, but the three-dimensional position is less accurate and potential obstacles have to be tracked from a distance so that they can be avoided safely. For coverage of the area in front of the robot, this is a suitable approach, but when turning in place new objects may escape the view-cone and result in failure to detect small objects close to the robot.

The final decision was to mount the cameras on top of the robot, looking down with an approximate 30° angle. This causes the most vibrations but gives the best view of the ground. The vibration of the camera influences the camera path only while the robot is moving. Since a method for the reconstruction of the camera path was developed (Chapter 5) that uses camera pictures as the only information the computed path will include those vibrations, leading to a correct stitching of the final disparity map. This approach led to problems as shown in chapter 6.

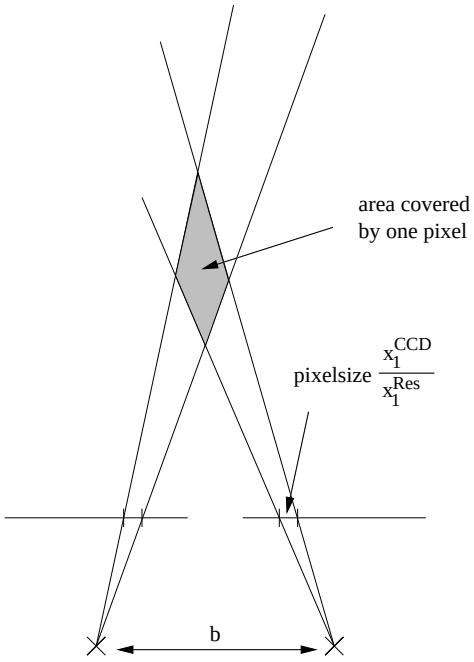

Figure 3.2. Space region covered by one pixel pair

The camera centers heights above the ground are $\approx$ 900 mm. The focal length is 6.3 mm and the CCD chip has a size of 6.0×4.2 mm. Intersecting the view cone with the ground plane gives an observable range between approximately 700 mm and 5000 mm.

3.3 Baseline and resolution considerations

The baseline between the two cameras and the used resolution influence the accuracy of the depth reconstruction in several ways. Figure 3.2 shows a typical setup. The gray parallelogram visualizes a region that gets the same disparity value. Consider the baseline being shortened then the skew of the parallelogram increases and the accuracy of depth reconstruction is reduced. The resolution affects the result in a similar manner. The horizontal length of the CCD chip stays the same, but the number of pixels covering this area is changed. Therefore one pixel covers different areas at different resolutions. Increasing the resolution decreases the area covered by one pixel, also decreasing the horizontal and vertical extension of the parallelogram.

From equation 2.4 it follows that the baseline has a linear influence on the depth values. The same is true for the resolution: $\mathbf{X_1} = \frac{\mathbf{x_1}^p \mathbf{x_1}^{CCD}}{\mathbf{x_1}^{Res}}$[1], therefore

[1] $\mathbf{x_1}^{CCD}$ denotes the horizontal length of the cameras CCD chip, $\mathbf{x_1}^{Res}$ the horizontal resolution of the frame grabber and vector elements accompanied by a p refer to pixel

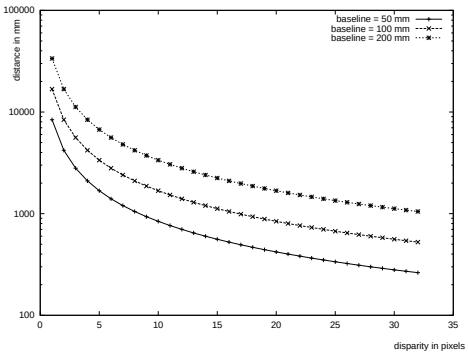

Figure 3.3. Depth vs. disparity for different baselines

$d = (\mathbf{x_1}^{lp} - \mathbf{x_1}^{rp})\frac{\mathbf{x_1}^{CCD}}{\mathbf{x_1}^{Res}}$ and equation 2.4 becomes $\mathbf{X_3} = \frac{bf\mathbf{x_1}^{Res}}{(\mathbf{x_1}^{lp}-\mathbf{x_1}^{rp})\mathbf{x_1}^{CCD}}$. This is depicted in figure 3.3. The baseline was doubled in each step and the depth values follow this behavior. It is intended to use as few disparity levels as possible to keep the computations fast, but enough to ensure a reasonable coverage of depths. The Small Vision System [21] allows for 16, 24 and 32 disparity levels. The disparities are interpolated to give 1/4 sub-pixel accuracy. In general it seems to be better to use a wide baseline, because the accuracy at small disparities (large depths) gets better. The disadvantages are large disparities for nearby objects and widely differing viewpoints, giving rise to difficulties in the matching process. The main problem are occluded areas that become visible in the second view. It was chosen to use 16 disparity levels (interpolated to 64) and a baseline of 213 mm, where the odd number was chosen for practical reasons. As can be seen in figure 3.3 this choice covers the 1-5 m range quite well and the resolution at low disparity levels is acceptable as well for the depth range of interest. The resolution of the acquired images has two effects on the performance of the system. First it defines directly the data rate that has to be processed. Second reconstruction accuracy is influenced in all three space directions. Also the number of disparity levels needed to cover a certain depth range is influenced directly by the resolution. If the resolution is doubled, the number of disparity levels needed to cover a fixed depth range doubles as well.

3.4 Image normalization

For the feature matching step (see section 2.4) it is very important that the two images that are used for intensity comparison have the same contrast and the same intensity range. This is even more important when the sum of absolute differences is used as a measure of similarity. Consider a region

coordinates

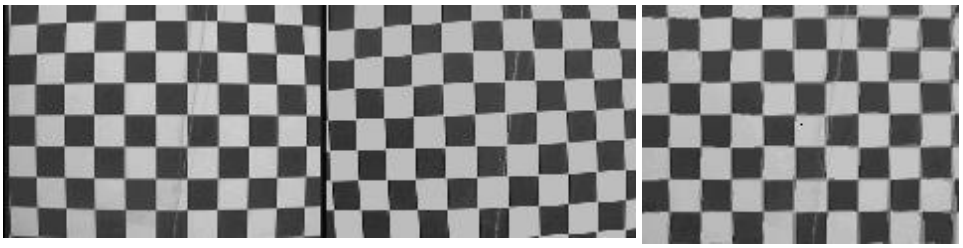

Figure 3.4. Calibration of images: The left part of the image shows an uncalibrated image pair. The images are recorded from the same position for both cameras, i.e. it is not an image of the stereo rig from one position, but the stereo rig was moved to obtain the images. The right hand side shows the rectified images superimposed on each other. The gray pixels correspond to imperfect calibration. Ideally all pixels should be either black or white. Note the small radial distortion in the original images (the left and middle pictures for images taken by the left and right camera respectively)

of uniform intensity that is, due to differences in aperture imaged at different intensity levels. The sum of absolute differences yields a big score and therefore the match is considered bad. The effects of different apertures can not only be introduced by different camera settings, but also by different lighting conditions (for example one camera in the shadow and the other one not). Its effects on the matching quality can be reduced by normalizing the image. The simplest way of doing this is to stretch the current range of gray values to fit into the representable range of values, i.e. a byte in our case. This is not a very good way of normalizing the images because highly reflecting surfaces will always have intensities near to the maximum value, distorting the real range of intensities in the image. A better technique is therefore to calculate the mean and the standard deviation of the intensities in each image. It is assumed that the intensities follow a Gaussian distribution. If this is true the most intensities ($\approx 95\%$) will lie in the interval $\overline{x} - 2\sigma \leq \overline{x} \leq \overline{x} + 2\sigma$. Therefore a normalization, that stretches this interval to the available range of intensities was used. Values falling outside this interval are clamped to black or white respectively. Using an efficient normalization on the images made it possible to use the fast sum of absolute differences as measure of similarity in a way that it outperforms cross-correlation (see section 5.2).

3.5 Camera calibration/Image rectification

Camera calibration is the task of determining the internal and external parameters of a camera system. Internal parameters are for example the *principal point* i.e. the orthogonal projection of the center of projection onto the image plane, aspect ratio, focal length, pixel size and radial distortion coef-

ficients. External parameters describe the physical transformation between two camera centers. Most internal parameters are denoted on the camera or can be measured directly. For the rest of the parameters, i.e. radial distortion coefficients and rotation around the viewing axis an adhoc approach was used. Pictures of a checkerboard seen from the same position were recorded with both cameras. The result can be seen in the left part of figure 3.4. Both, the left and the right picture have a slight radial distortion, the right camera has an additional rotational disturbance. The radial distortion was considered insignificant at first, but experiments proofed the importance of correcting it (see section 6.4). The center of the radial distortion, i.e. the principal point was assumed to coincide with the center of the image. As a model for the radial distortion a 3rd-order polynomial was used.

$$r_c = c_1 r + c_2 r^2 + c_3 r^3$$

where c_i are the coefficients, r is the current and r_c the corrected radius of a pixel. The radius is the pixels distance from the center of radial distortion. A small tool was implemented, that allows to change the parameters by hand. To check the results, the two corrected images where superimposed (figure 3.4 right side). The visible area on the CCD chip and with that the pixel size was determined by triangulation using a known distance to the checkerboard object and the known focal length of the camera. Because of the strong influence of the calibration on the performance of the motion estimation better methods for calibration should be investigated.

3.6 Local map computation

For the rest of this report the following notation will be adopted. A “local” map is a map constructed from one view only. A map referred to as being “global” is constructed from several views i.e. it is a combination of several “local” maps. The quotation marks will be dropped. The word map is used to describe some representation of the 3D world. A local map is usually a 3D-point cloud whereas a global map refers to a more sophisticated representation (chapter 4 contains a deeper discussion).

Figure 3.5 shows an example of a typical 3D-point cloud. The cloud is in camera coordinates and represents the ground plane with the stone from figure 2.6. The stone is situated on the right side of the image. The viewing direction of the camera in figure 3.5 is from right to left. The grid represents the x-z plane in camera coordinates. The point cloud is obtained from a disparity map like in figure 2.7 using equation 2.4. The only thing to take care of is the shift in disparities introduced by the offset mentioned in section 2.5 and that the disparity map is usually not defined everywhere in the picture (this might occur due to a too bad matching score or because the search window does not fit completely into the second image).

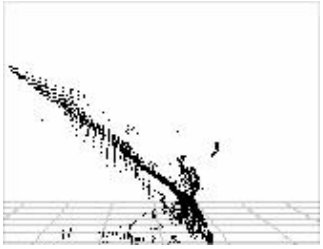

Figure 3.5. Example of a local map

Since a variable offset is important when using a fixed number of disparities it was included into the implementation of the map building algorithm. The importance of this variability is the ability of selecting different depth ranges that can be computed correctly. To select an appropriate offset feature points are matched between left and right stereo images[2] (see section 5.2). The mean disparity of those matches is then used to compute the offset for the disparity computation. Since the mean disparity value should be in the center of the interval of pixels, that are checked for correspondence, the final offset is calculated as mean-(number of disparity levels / 2). It is important to vary the offset because pixel correspondences outside the search range cannot be found, yielding random disparity values, that are similar to white noise.

3.7 Summary

The cameras are mounted high on the robot looking downwards in an approximate 30° angle. This ensures a good view on obstacles in front of the robot. The baseline was chosen such as to give a compromise between accuracy in depth computation and a reasonable search range for the disparity calculation. The resolution was set to 160x120 pixels mainly because higher resolutions demand too much computational power. It was found that image normalization is vital to the matching process when using the sum of absolute differences as measure of similarity. The same attention should be paid to camera calibration as it influences the accuracy of the motion estimation of chapter 5 considerably.

[2]It could be argued that left-right matching is unnecessary since the disparities could be taken directly from the disparity map. This argument assumes, that all disparities can be reconstructed correctly, which is not the case when using a fixed number of disparity levels.

Chapter 4

Map representation

4.1 Introduction

The aim of this chapter is to describe an efficient data structure for the representation of a 3D world for navigation purposes. The data structure is intended to integrate several local maps of the environment observed at different positions into a global map that represents the environment visited so far. Because of memory restrictions and the difficulty of the simultaneous localization and mapping problem parts of the map are discarded after some time.
Requirements for such a data structure are

1. **efficiency** in computation, analysis and storage space
2. **accuracy** the structure should be able to represent all relevant[1] objects
3. **robustness** against outliers in the local maps
4. **flexibility** inexpensive extension and deletion of parts of the map

It is assumed that the robot moves in a static world i.e. moving or deforming objects are not intended to be represented and robustness against those is not aimed at. Nevertheless it turns out that the chosen structure is - with slight extensions - able to handle moving objects. But no experiments have been conducted to prove this claim.
Often a $2\frac{1}{2}$D representation of the global map is chosen i.e. it is assumed that the environment can be represented in the form $f(x, y) = z$. It is argued that for navigational purposes it is not necessary to model overhanging objects correctly. They can be projected down onto a common plane to mark this regions maximum height. A problem for an algorithm using dead reckoning is the choice of the common plane. Because of accumulated error in the

[1] relevant are all objects that are higher than the radius of the wheels ($\approx 200mm$)

camera-position the plane does not stay in the same global position, causing maps merged into that representation to be misaligned. Singh and Digney [34] use a quadtree data structure [28] to represent the heights above the common plane.
For indoor applications often an *occupancy grid* [23] is chosen to represent a global map. An occupancy grid represents the world as an array of cells, whose members are being marked as impassable or empty cells. To account for sensor noise a cell is not treated as a boolean value but contains the probability of the represented area being occupied by some object.
The main idea of this chapter is to create a three-dimensional (boolean) occupancy grid using an octree data structure, with nodes that keep track of neighborhood relationships. The space partitioning is stopped after some levels (depending on the accuracy needed and the depth resolution of the disparity computation) and the cubes at the highest subdivision level that contain at least a certain number of points are interpreted as belonging to a rigid object present in the world. Using neighborhood relationships a flood-fill algorithm common in computer graphics is applied to the tree nodes, making it possible to mark occupied regions in space. This in turn allows for the extraction of the boundaries of those and the creation of a polygonal model. The occupancy grid representation is useful for navigation purposes whereas the boundary representation is better suited for visualization of the computed map.

4.2 The octree

The chosen data structure was the bucket PR-tree described in [28] section 2.8. It is a space partitioning tree based on regions. PR stands for point region and describes the fact that points are grouped in regions in the tree. The term *bucket tree* refers to a tree with more than one point present in the leafs. When the number of points in such a bucket reaches a certain adjustable threshold the leaf is split into equal sized child nodes. In the three-dimensional octree case a node is split into 8 sub-regions. The bucket size determines the space requirement of the tree [2]. A PR-tree was chosen because relatively few assumptions can be made about the distribution of the incoming 3D-points since an outdoor environment is much less structured than an indoor environment for example. A region-based tree is suited best to represent unknown data distributions because it subdivides regions equally, making it possible to represent every region in the space with the same accuracy. The price is a big cost in storage capacity for a fully occu-

[2] the assumption is made that equal point sets are used. The distribution of the data points that are to be inserted into the tree has the biggest influence on the storage space requirements. It is assumed that the data points represent a two-dimensional surface in a three-dimensional world, thus covering only a sparse region of the whole 3D-space. The space requirement should therefore be comparable with that of an fully occupied quadtree

pied tree as it would occur for totally random data. But since our range images are supposed to represent *surfaces* in the 3D-world the tree will be sparsely populated, keeping the storage costs at a reasonable level. Making it a bucket-tree has the additional advantage of introducing a parameter (the bucket size) for outlier robustness. Outliers are by definition randomly distributed. Therefore outliers will seldom occur at the same position in contrast to the real data, leading to splitting in places of real surfaces and discarding outliers. The sensitivity of the tree depends on the bucket size. It can be effectively used to discard outliers, that are not systematic.
The current implementation works with a fixed maximum tree depth to avoid an explosion in space requirements. At the finest level of subdivision individual point coordinates are being dropped and only the mean value of them is stored in the corresponding region. This measure helps reducing the amount of stored data considerably. The region-tree representation is advantageous because it replaces the point-wise representation of the local maps by a regional representation. It comes at the cost of loss of accuracy, but can be adjusted using the maximum split level and the size of the region that is covered by the global map. Therefore it can be kept at the same level as introduced by the discrete disparity values. A further advantage is the introduction of neighborhood relationships that are not present in the point data representation of the local maps. Furthermore sparsely populated regions (i.e. regions without objects in the world) are easily identified, yielding a suitable tool for path planning.

4.2.1 The neighbor concept

It is important to keep track of the neighbor relationships in the tree to support region marking and boundary extraction in a later stage (see section 4.3 and 4.4). Because the measured point data are noisy, a surface in the real world will not actually correspond to a distinct surface in the integrated map. Instead a band of points describes a certain surface. These stay, depending on the resolution of the tree and the variance of the noise, in one region of the tree or spread out over more than one. Then it is necessary to mark those regions, that most likely contain a surface. These have the form of more or less narrow bands in 2D or blobs in the 3D-case. The whole situation is visualized in figure 4.1. For simplicity a 2D example was chosen. The region to be represented is now a part of a plane and the "surface" to be extracted is a curve. The upper picture shows the tree for perfect point data. The tree is split in the vicinity of the curve. The rectangles show the splitting, the filled rectangles correspond to regions marked as belonging to the curve, the circles represent the mean value of all points that are present in the corresponding region (for regions belonging to the curve the circles where dropped to clarify the extracted boundary) and the lines connecting the regions show up the neighbor relationships. The fat line shows the

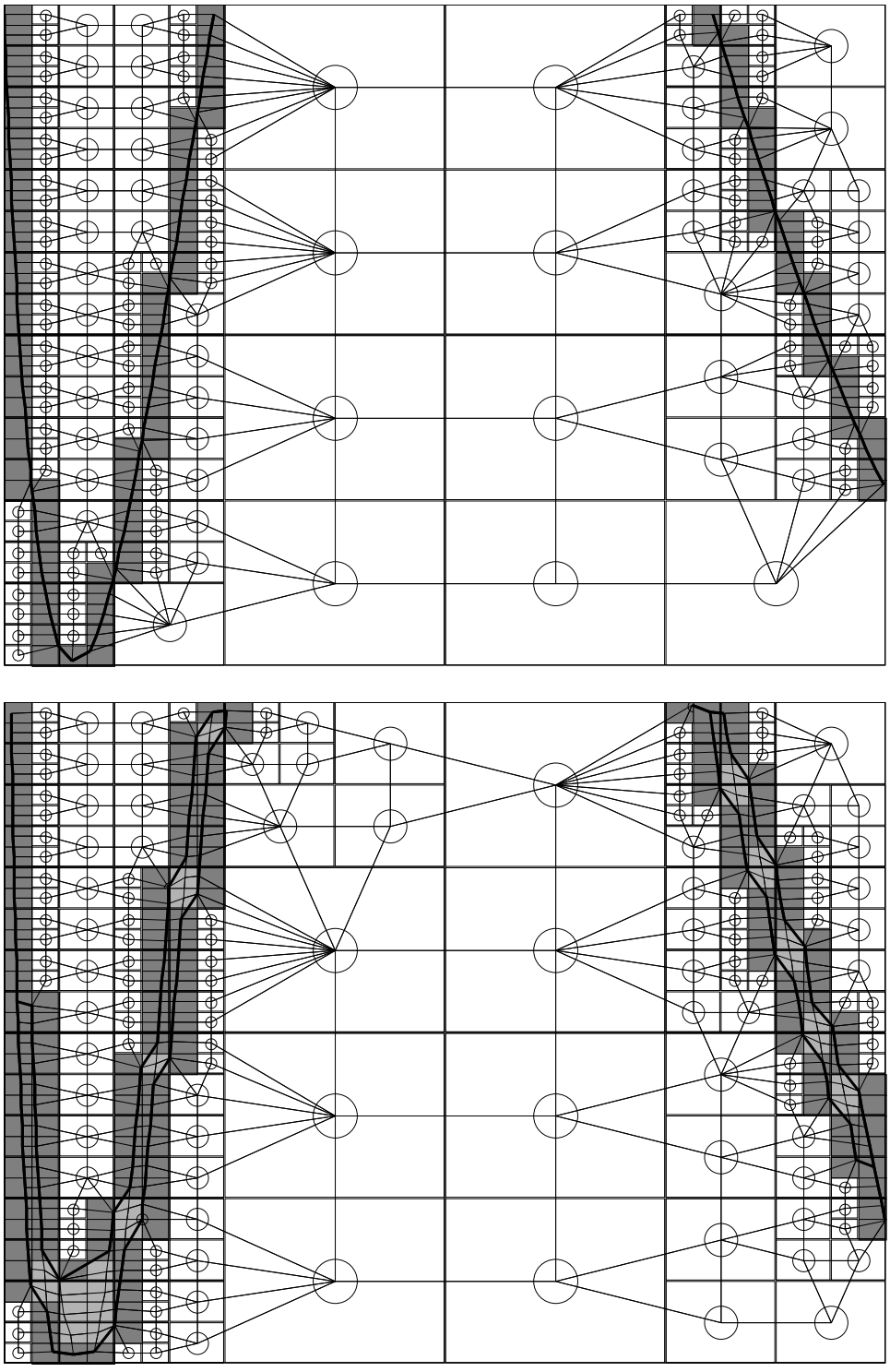

Figure 4.1. A quadtree example - perfect data (upper) and noisy data (lower) was inserted

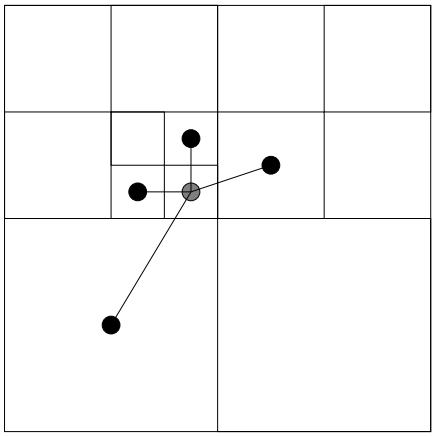

Figure 4.2. A split region with marked neighbors

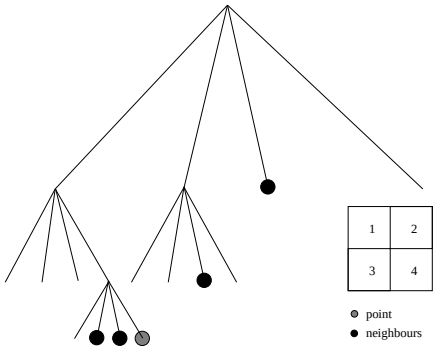

Figure 4.3. Corresponding tree

boundary of the "surface". The lower picture shows the same situation, but with noisy data points. The split region in the tree is more spread out and the "surface" forms a band. Boundary extraction leads to a closed curve.

The neighbor information is explicitly kept track of because the extraction of this information from the tree itself is nontrivial and can cause expensive search operations in the tree. A simple example is depicted in figures 4.2 and 4.3. The neighbor information is needed frequently enough in the filling phase of the algorithm to justify the additional expense at insertion time. There are also different types of connectivities to choose between. In 2D they are called *4-connectivity* and *8-connectivity*. 4-connected neighbors share a common side, whereas 8-connected neighbors share common sides or common corners. This concept can be extended to 3D as well, leading to three cases: shared faces, shared faces or edges and shared faces, edges or corners. The first case will be the generalization of 4-connectivity (*2n - connectivity*), whereas the third case is the corresponding 8-connectivity generalization ($(3^n - 1)$-*connectivity*)[3]. When the terms 4- and 8-connected are used in the three-dimensional context it refers always to those two cases. Using 8-connectivity does not yield a significant advantage but increases the expenses considerably. In three dimensions there are already 27 neighbors (instead of 6 for 4-connectivity) to process. Therefore 4-connected leafs where chosen.

4.2.2 Insertion

Building the tree is a simple procedure. The whole starts with a single box spanning the whole region. This region spans the area around the robot that

[3] see the appendix for a fast way of determining the connectivity

is going to be available for the robots navigational decisions. It is moved when the robot is moving to keep track of the surrounding environment. The procedure of moving the represented region is described in section 4.5. A node describing a region contains its border coordinates as well as a list of children and a list of neighbors. The single region in the beginning has of course no neighbors. Now points are inserted as long as the bucket size is not reached. The region is halved in every dimension when the bucket size is reached. The points contained in the parent region are distributed over the children, depending on their position in space and the neighbor relationships are updated. The siblings of each of the new regions together with the neighbors of the parent region are tested for neighbors with the new regions and depending on the result inserted into the corresponding neighbor list. Finally the neighbors of the parent region are removed from the neighbor list of the parent region and the parent region is removed from the neighboring regions neighbor lists. This simple procedure keeps the neighbor information in the tree consistent for all insertion operations and is much simpler to perform than a search operation, because it involves only local information [4].

The pseudo code for the insertion function is shown in figure 4.4. In the implementation the useless recursive calls to insert with children that do not contain the Point are avoided since insertion and check for neighborship are the determining factors for the speed of execution. Every local map contains at most $160*120 = 19200$ points to be inserted into the tree. At 5 frames per second this is already near 100000 points. Therefore the insertion routine is one of the bottlenecks of the program developed in this thesis.

4.2.3 Deletion

Deletion of a sub-tree is necessary to follow the movement of the robot with the tree. Once the robot leaves a certain inner region of the tree a part of the tree that lies in the opposite direction of the movement has to be deleted and one in the direction of the movement has to be newly created. Details are given in section 4.5. The deletion of a node is quite simple - all sub-regions of the region in question have to be deleted and the neighbor list of regions having one of the to be deleted sub-regions as neighbors have to be updated. Then the node is removed if it is a direct descendant of the root otherwise it is turned into an empty sub-region. If all siblings of the sub-region are empty as well, the parent node can be marked empty or taken away as before.

[4] neighbor relationships spanning large portions of the tree (for example the pairs 1-4-4 and 2-3 or 1-4-4 and 3 in figure 4.2) are built incremental instead of searching for them in the whole tree in every step

```
function insert(Point,TreeNode)

    if (Point not in TreeNode.Region) exit

    if (TreeNode.BucketSize >= TreeNode.NumberOfPoints + 1)
        insert Point in TreeNode.PointList
    else
        if (TreeNode.Depth < MAX_TREE_DEPTH)
            split(TreeNode)

            insert all points in TreeNode.PointList
            in corresponding child

            for all children do
                update Child.Neighbors with siblings
                update Child.Neighbors with TreeNode.Neighbors
                insert(Point,Child)
            end
            remove TreeNode.Neighbors
        else
            update TreeNode.MeanValue
        end
    end
end
```

Figure 4.4. Pseudo-code insertion

4.3 Region filling

After the insertion of data points, a tree similar to that in figure 4.1 with exception of the filled rectangles and the boundary is in existence. The nodes most likely belonging to a rigid object remain to be marked. This is achieved through the application of an flood-fill algorithm, usually used in computer graphics to fill (pixel-wise) oddly shaped, closed regions. In this algorithm a so called *seed position* is selected. If the position is valid, that is it does not have the boundary color, it is colored with the selected fill-color. Afterwards the function is called recursively with all the neighbors of the seed pixel.
Since the algorithm is solely based on neighbors and an abortion criterion it can be used to mark connected regions in a tree as well. The only things to be defined are the validity of a node and the neighboring nodes (already

```
function delete(TreeNode)

    if ( not leaf-node(TreeNode) )
         for all children do
             for all neighbors of TreeNode.Child do
                 delete TreeNode.Child from neighbor list
             end
             delete(TreeNode.Child)
         end
    end

    if ( all siblings are empty )
        delete TreeNode.Parent
    end
end
```

Figure 4.5. Pseudo-code deletion

done in section 4.2.1). As validity criterion the split level of the node was adopted. Nodes at the highest level of subdivision that fulfill an additional occupancy criterion are considered to be fill-able. To fulfill the occupancy criterion a node has to contain at least a certain number of data points. This is needed to exclude nodes at the highest subdivision level that were introduced by splitting a node with an unevenly distributed data set. These regions are usually nearly empty and thus do not belong to a surface.
Now the only missing part for the algorithm to work is a seed position. Actually there can be more than one connected region contained in the tree, which demands a seed point for every one of those connected regions. In figure 4.1 there are for example 2 independent connected regions (one on the left and one on the right side).
Since membership in a connected region is unambiguous, that is a node belongs to one and only one connected region, it is sufficient to find any point of the region and use it as a seed position in order to fill the whole region. That implies that traversing the tree and applying the flood-fill algorithm to every fill-able node is an appropriate method to find and mark every connected region in the tree. Of course it is unnecessary to apply the algorithm to already marked nodes.
After this step the map is already usable for navigation, given a method for fitting the single local maps together in a consistent way and assuming that the robots field of view stays in the region covered by the tree. Chapter 5 describes a method to meet the first requirement and section 4.5 treats the

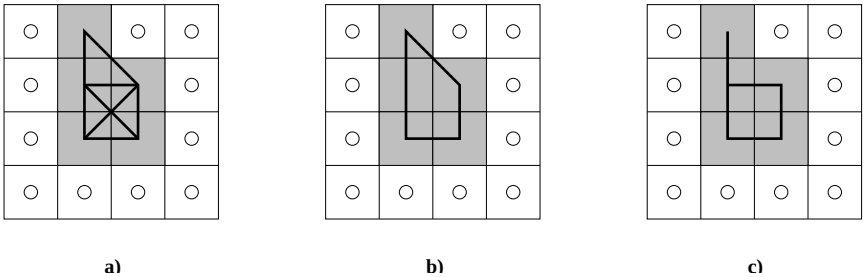

Figure 4.6. Boundary extraction: a) all neighboring (8-connected) boundary nodes are connected, b) neighboring (8-connected) boundary nodes that share a common empty neighbor are connected, c) neighboring (4-connected) boundary nodes that share a common empty neighbor are connected

movement of the region covered by the tree in a way consistent with the movement of the robot.

4.4 Boundary extraction

This section deals mainly with visualization issues. It is necessary, because an octree structure is hard to check for correctness, making it difficult to analyze the behavior of the developed algorithms. It is therefore important to have an efficient visualization of the computed map. The boundaries of the connected regions computed in the last section were found to be a good choice for this purpose. Therefore this section deals with boundary extraction and mesh generation. These two steps can be performed separately. The first - marking of boundary nodes - can be done during region-filling, avoiding a second traversal of the tree. Nodes that have an empty node as neighbor or share a border with the outer limits of the region represented by the tree are considered to be boundary nodes.
The second step - mesh generation - requires the connection of "appropriate" boundary nodes. The term "appropriate" needs to be defined in this context. Figure 4.6 shows a situation that can also be found in the lower picture of figure 4.1 (for example in the upper left corner). As shown in figure 4.6 a) neighboring boundary nodes cannot simply be connected, because cross connections would occur. This suggests to introduce the further requirement that *connectable* nodes need to share a common empty neighbor as well. If the nodes are situated at the outer boundary of the octree they are required to share the same outer boundary. Therefore two boundary nodes being connectable means they are neighbors and they share a common empty neighbor. The last decision in this context is the connectivity under which the two boundary nodes are neighbors. Figure 4.6 b) and c) show the possibilities in 2D. Here 8-connectivity was chosen, because the

visual appearance of the resulting boundary is more pleasant and the estimation of the boundary is more conservative[5] than in the 4-connected case c).
The simplest method to generate a 3D-model is now to check all empty, or alternatively all filled, leafs for neighbors marked as border nodes. If two border nodes are connectable they are connected by a line. The result is a wire frame model. It has the disadvantage, that front- and back-side of the extracted blobs are visible at the same time, making it difficult to recognize the structure of the surfaces. Because of that a triangular mesh was constructed from the tree. The algorithm uses the same principle of connectable border nodes as neighbors of empty nodes, but now three border nodes have to be mutually connectable to produce a valid triangle of the mesh. The algorithm produces too many triangles [6], but since the polygonal model is, at the moment, intended for visualization only, overlapping triangles are a minor problem.

4.5 Following the movement

Up to now it is possible to represent the surrounding environment of the robot in a fixed area. Because the robot is moving, it is necessary to follow the movement of the robot with the tree. To keep the stored data in a reasonable size some of the things seen have to be forgotten. To move the area represented by the tree two things are necessary. The first is to estimate the direction the robot is moving in and the second the reorganization of the tree. The direction of movement can be estimated by the number of three-dimensional points, calculated from the disparity map and projected from the current camera position, that fall outside the tree compared to the number of points being integrated into the tree. If the quotient is high, it is not possible to represent a large number of points in the current extensions of the tree. The tree has to be moved or enlarged. Enlargement is not a good choice because of increased computation time and storage space. The worst case storage requirement is 2^{d^n} where d is the dimension of the tree (in this case 3) and n the number of splits. As mentioned before it is expected that the storage space requirement behaves like a worst case scenario in two dimensions i.e. 2^{2^n}. This leaves us with a quadruplication of storage space for every split level added. Most of the computations are not affected seriously, but since there is a complete traversal involved in the region marking process it was decided to move the region represented by the tree instead of extending the tree in such a way that it spans the uncovered region as well. To get the direction of the movement in a certain time-step a histogram of

[5] the mean values of the nodes are used to construct the boundary, diagonal connections in the 8-connected case put the boundary farther out than in the 4-connected case

[6] for example consider four connectable nodes; those can be connected by four triangles

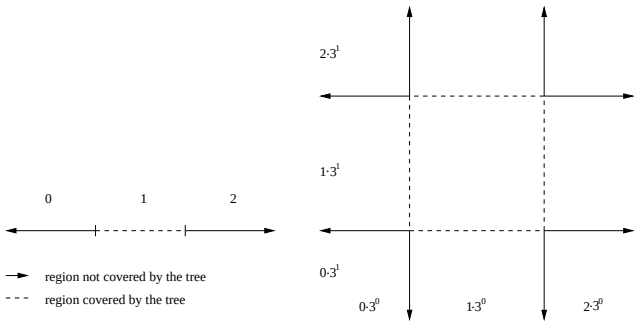

Figure 4.7. The directions of movement can be encoded in a system with basis 3

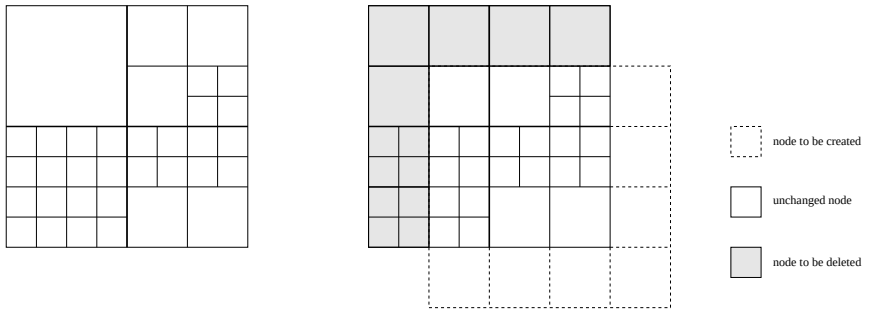

Figure 4.8. Movement at split level 2, example showing the affected nodes and the old and new region covered by the tree. The upper left node has to be split to allow movement movement at that level

the points falling out of the tree (during insertion) is used. The directions are encoded as shown in figure 4.7. A system with base 3 is used and it is checked in every dimension in which bin the point falls[7]. Each dimension corresponds to a digit position in base 3. If a threshold of points falling out of the tree ,i.e. $\frac{\sum_{i=0..26, i\neq 13} b_i}{b_{13}} > t$, is reached the tree should be moved into the direction with the most support (except for the inner region of course) indicated by the histogram. To move the tree some considerations have to be made. The reorganization of the tree should be as cheap as possible. On the other hand only a reasonable amount of information should be lost. If reorganization of the whole tree is to be avoided, the tree movement has to be done in step-sizes that are powers of $\frac{1}{2}$ the side-lengths of the cube that is spanned by the tree, because most of the nodes below this level can be left unchanged. The simplest way is to use the first split level, designated by the thick lines in figure 4.8, but in the worst case, which is movement in all

[7] $b_1 3^0 + b_2 3^1 + b_3 3^2$ gives one of the 27 directions in 3D. $1 \times 3^0 + 1 \times 3^1 + 1 \times 3^2 = 13$ is inside the region represented by the tree

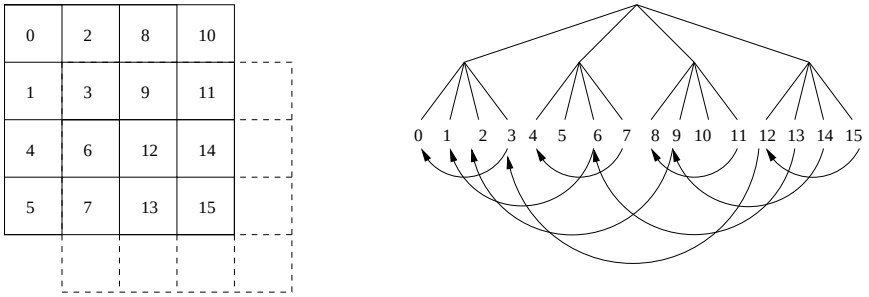

Figure 4.9. Reorganization of the tree: arrows mark mappings, nodes with no emanating arrows are to be deleted in the old tree, arrows without incoming arrows are to be created in the new moved tree

dimensions, only $\frac{1}{8}$ of the information is kept. The figure shows movement at level 2 in two dimensions. In general the worst case information loss is given by $1 - \frac{(2^{level}-1)^n}{2^{level^n}}$ where n is the dimension of the tree. This suggests a movement at a high split-level. The disadvantage is that the tree has to be split completely up to that level to make the reorganization caused by that movement possible, giving a fully occupied tree up to that depth. That is of course highly undesirable. Therefore it is preferable to do the movement at level 2 or 3.

Because the mapping of nodes for a specific movement is always the same and the number of possible movements is restricted, it is possible to pre-calculate the mappings and store them in a look-up table. The nodes on a specific level have to be numbered uniquely in order to do that. Nodes that have to be deleted or newly created can be marked specially. An example is given in figure 4.8 and 4.9. Figure 4.9 shows the reorganization that is necessary. The majority of the nodes, together with their corresponding sub-trees, can simply be reattached to another node in the new tree. The arrows in the figure mark those mappings. The neighborhood relationships between those nodes stay the same. Nodes without outgoing arrows are deleted. Their information content is lost and they have to be removed from the neighbor-list of the remaining nodes as explained in section 4.2.3. Finally new nodes have to be created in the newly covered space regions. The neighbor-lists of the original nodes have to be updated in a special way, not covered by the insertion algorithm of section 4.2.2. This is the case because neighbor information is not available for the newly created nodes, making it impossible to build it up incrementally.

As was mentioned before the mapping between the preserved nodes as well as the nodes that have to be created or deleted can be determined beforehand. Unfortunately that does not apply to the neighbor relationships. They change in an unpredictable way and have to be computed every time the tree is moved.

4.6 Extensions

The octree as presented in this chapter is a very flexible data structure to represent unordered point sets and order them in a way to enable identification of occupied space regions and extraction of their boundaries. Here are some ideas how to further enhance this flexibility.

1. The marking of occupied regions can be based on the number of points inside a node in relation to the total number of points present in the tree instead of having a fixed threshold that considers the number of points in a node only. This facilitates robustness against moving objects, because evidence for static objects is higher (they appear in several frames) while moving objects are "forgotten" after some time. Additionally merging of regions that become uninteresting can be incorporated.

2. The data points that are to be inserted into the tree could be calculated with an estimate of the covariance of their error. The tree could then be converted into an occupancy grid that accounts explicitly for sensor noise.

3. The fill-state of leaf nodes can be passed upwards in the tree to simplify the detection of occupied regions. This fastens up the marking and allows the extraction of a polygonal outline at different levels of detail. The polygonal model can be used for collision detection as well, starting at coarse levels and concentrating on regions of possible collision, taking full advantage of the hierarchical structure of the tree.

4. Regions that where used during the last point insertions can be marked "dirty". This measure speeds up the process of region filling as well.

5. The step-size of the movement can be based on mean and variance of the distance of the data points falling out of the tree.

6. Deleted nodes do not have to be deleted but can be collected in a pool of available node-structures, decreasing the number of memory allocations and the fragmentation of the main memory that occurs due to deletion and creation of nodes while moving the tree.

Those measures could help to speed up the implementation and enhance the usability of the data structure. They were left out for time reasons, but even without them the octree is a feasible and extendible data structure suitable for the representation of a global map of restricted size.

Chapter 5

Map registration

5.1 Introduction

Map registration is the process of aligning several local maps in such a way, that they fit together and make up a bigger global map. In order to do that it is necessary to find the overlapping area of two or more 3D free-form surfaces. This is a difficult problem and several approaches have been tried. The approaches can be divided into two main groups. The first of those are algorithms that rely on the knowledge of the camera position to project the local map from the current position into space. Afterwards the projected points are being integrated into the global map. The camera position is usually computed from a number of sensors like odometry, compass etc., thus introducing measurement errors. This position estimation known as *dead reckoning*[1] yields an estimate relative to the last position. The algorithms using that technique are fast but suffer from an accumulation of errors in the path calculation[2]. Due to their speed this is the group being used for real-time navigation purposes, for example in [9, 34]. An algorithm that takes the problems of dead reckoning into account is said to address the *Simultaneous Localization And Mapping problem*[3] (SLAM) [10].
The second group of algorithms on the other hand computes the best fit of two complete 3D-models - the already known global map and the current local map. This approach is computational expensive, but gives better results than the first one. This is because the whole information collected up to the current local map is used to fit it into the global map obtained

[1] Dead reckoning navigation estimates the current position from the last known position using measurements of direction, velocity etc. Columbus used dead reckoning on his journeys to America.[25]

[2] As can be seen in Columbus' route. He intended to sail directly westwards, but ended up with a route that headed west-south-west (although he was using the compass).

[3] Localization is the process of finding the own position in an a priori known map, while mapping constructs a map, given the position of the observations. SLAM is more difficult since both tasks have to be performed.without a priori information.

so far. It implies big amounts of data and therefore long processing time. Also the local map has to be preprocessed to introduce topological information to the data obtained from the range sensors, before using it for the alignment computations. In [17] for example complete polygonal meshes are fitted and the registration together with the necessary preprocessing takes about 7 minutes for each local map. The number of data points seems to be much higher than in this application though. The interest in 3D-model matching is big, because it is needed in map-building as well as in object recognition and localization. In object recognition a partial model (usually obtained from a single view) is compared to a number of complete models in a database. The one with the highest matching score is the recognized object. In map-building applications the best place in the global map is sought and for localization purposes the best fit of the observation to the known map is searched for. Methods for 3D-model matching are for example given in [32, 19, 17, 20].

The approach taken in this thesis uses dead reckoning and images alone to calculate the motion of the camera. The purpose with that is to have a flexible vision system, that is not dependent on a special robot platform. The task of motion computation is essentially the estimation of the fundamental matrix (see 2.2.3) for two successive images in the video sequence. Because the same cameras are used for all images in the sequence it is the essential matrix that is sought [4]. The computation of the fundamental matrix was tried, but rejected due to sensitivity to noise in the data set even when robust methods like RANSAC [13] where used. For this computation it is necessary to have, depending on the method, 7 or 8 corresponding pixels in the two images, that are to be related by the fundamental matrix. The problems in the computation could not be located exactly, but arose probably from the small data set (≈ 30 matches) and missing sub-pixel accuracy in the feature point detection.

However since a stereo-camera is available all information of it should be used. An approach similar to photo stitching [40] is used to address the problem of surface matching. Two successive stereo images (depicted in figure 5.1) are used to find consistently visible points in all 4 views. From those, two sets of 3D-points are computed, that are finally used to calculate a best rigid motion between them. The rest of the chapter covers the individual steps more thoroughly.

[4] $\mathbf{F} = \mathbf{C}^{-T}\mathbf{E}\mathbf{C}^{-1}$ [13] where $\mathbf{C}$ is the 3×3 part of $\mathbf{K}$ the camera calibration matrix; $\mathbf{E}$ contains only rotation and translation between the two camera coordinate systems: $\mathbf{E} = [t]_{\times}\mathbf{R}$

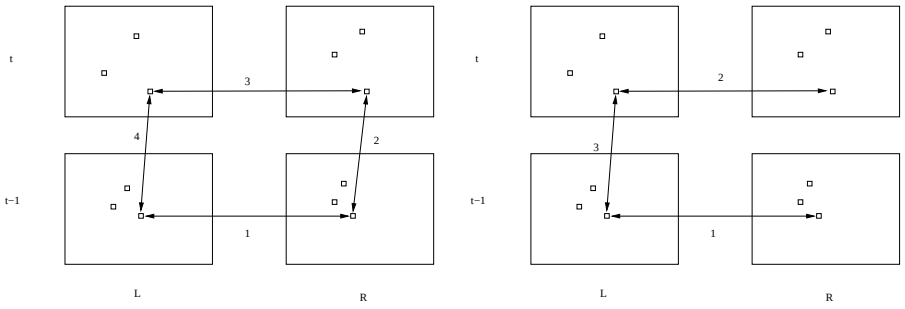

Figure 5.1. In the circular matching (left) each matching step (1-4) is done with an L-R check, eliminating outliers

5.2 Finding consistent matches

A prerequisite for an algorithm relating two images/maps is to find related points in them. The difficulty of doing that for two local maps, i.e. two unordered[5] 3D-point clouds, led to the approach of finding the correspondences in the image intensity information. Matches between the left and the right image are used to compute sparse 3D-point clouds for each time-step and matching in time (figure 5.1) to establish relationships between 3D-points in the two clouds. Feature point detection is performed using the Good Features To Track operator [33] (see also 2.3). This detector computes feature points distributed evenly over the image and the output number of points can be adjusted. Around 250-350 feature points in every image were used. Relationships between those have to be computed. Correlation based matching was used to do this. Figure 5.1 shows two ways of obtaining consistent matches over time and space. The left picture shows a circular check. The matching steps (see 2.4) indicated by the arrows are performed separately. A back-check is done in every step to increase the probability of a good match, also a restriction of matching feature points being not farther away then $\frac{2}{3}$ of the resolution, mirroring the assumption of a restricted movement between frames was applied for efficiency reasons. Afterwards the matches are checked for consistency. If they make up a closed path they are considered to represent the same point in space. This circular check proved to be too restrictive. Therefore a second method, depicted in the right picture of figure 5.1, was examined. Here the left-right matches are performed first. The feature points that pass those checks are used for matching in time. The resulting matches are considered consistent. For both methods a post processing was implemented, that refuses matches that would exchange the positions of two points (i.e. $\approx 180°$ turn of the camera around the optical axis). Matches being consistent does not mean, that they are outlier-free,

[5] unordered in this sense means that connectivity information between the points is unknown

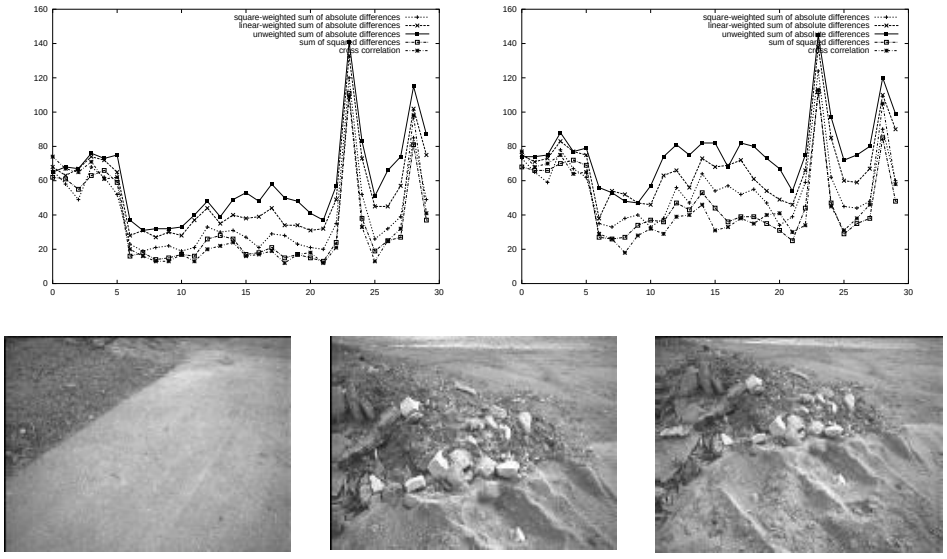

Figure 5.2. Upper: Comparison of similarity measures for feature point matching, shown is the number of consistent matches after matching 4 views using the circular (left) and the c-form check (right) of 5.1 for a video-sequence of 30 images versus the image index. 350 feature points were used and the sequence is a straight movement in a environment with high texture. Lower: picture 1, 23 and 30 of the video sequence

but with a large reduction of them. By visual inspection the outlier ratio seems to lie between 10% and 20% for a reasonably textured environment. Figure 5.2 shows the performance of some measures of similarity for both consistency checks. The number of points passing the test is plotted versus the image index of a 30 image video-sequence. The sequence starts with the robot standing for 5 frames. The start positions field of view consists approximately to two thirds of asphalt and one third of textured gras ground. The fact that no movement takes place during this period results in a relatively high number of consistent matches, although this performance is degraded by the low textured regions (asphalt) in the image. Nearly all consistent matches are found in the high texture area. From frame 5 to frame 20 a straight forward movement takes place. The number of consistently matched features decreases. Differences in performance of the measures of similarity become visible. Since the number of outliers is approximately the same after the consistency check for all similarity measures presented here, it was found, in conformance with [36, 21], that cross-correlation is not always the best choice for matching stability. Instead the sum of absolute differences is fast to compute and performs best in the experiment. When using the sum of absolute differences as measure of similarity careful normalization of the input images must be performed (see section 3.4). The peak around frame 23 arises from a change of direction. The robot slows down, stops and drives

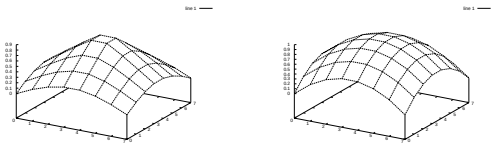

Figure 5.3. Weights for weighted sum of absolute differences - left linear, right squared

backwards afterwards. The peak is the position where the robot stops. The difference to the first situation is, that the robot views a highly textured area now. The low after the this peak is introduced by the backward movement, followed by another stop in the end of the sequence.

The two weighted sum of absolute differences measures where included because it was tried to achieve some rotational invariance for the matching step. The weights, shown in figure 5.3, were distributed circular around the center of the 8×8 box that was used for correlation purposes, once linearly decreasing and once with squared decrease for increasing radius. The 8×8 correlation window size was chosen to be able to efficiently use the Intel MMX instruction set [1], that features an instruction to compute the sum of absolute differences for 8 bytes simultaneously. The weighted sum of absolute differences is less efficient to compute and performs worse for translational movements, which are the most likely ones to occur. Even when the robot is turning the camera movement has a big translational component, because it is mounted in the front of the robot, whereas the center of rotation is situated at the center of the robot body. The presented matching methods perform similar in other video sequences not included here. As result the sum of absolute differences together with the c-form consistency check were chosen for further computations.
Now two 3D-point clouds with known correspondences between the points of the two sets are available for motion computation.

5.3 Calculation of motion

For the calculation of motion from the two sparse 3D-point clouds computed in the last section, the assumption is made that the robot is moving in an otherwise static world. This allows to calculate the rigid body motion that the cameras undergo from all the points in the two point sets. If this assumption is incorrect the points in the two point sets belonging to moving objects have to be identified. Since this is left out, the task consists of estimating the rotation and translation, that fits two noisy 3D-point sets together. The point sets contain outliers, but the majority of points are related by such a transformation because they represent points in a static

world. First a least squares fit of two noisy sets, containing no outliers is discussed and second two robust methods are compared and evaluated for their ability to remove outliers.

5.3.1 Least-Squares fitting of two 3D point sets

Apart from iterative methods, which were not investigated because they are usually slower than closed-form methods, there exist mainly two methods for this estimation problem. The first is based on the singular value decomposition [3] and the second on unit quaternions [16]. Another formulation of the SVD method, using geometric algebra, can be found in [8]. For a introduction to geometric algebra see [12, 14]. A summary of [3] is given here. The estimation problem can be formulated as

$$\hat{\mathbf{x}}^i = \mathbf{R}\mathbf{x}^i + \mathbf{t} + \mathbf{n}^i$$

with $\mathbf{R}$ and $\mathbf{t}$ being the sought rotation matrix and translation vector respectively. The noise component by which point $\mathbf{x}^i$ is degraded is $\mathbf{n}^i$. $\mathbf{R}$ and $\mathbf{t}$ have to minimize the squared error term

$$e^2 = \sum_{i=1}^{N} |\hat{\mathbf{x}}^i - (\mathbf{R}\mathbf{x}^i + \mathbf{t})|^2$$

to find the best least-square solution. The computed rotation takes place around an axis passing through the origin. This assumption is valid for the motion computation, because both point clouds are given in their respective local camera coordinate systems, with the centers of projection as origin. The centroids of the least squares solution and that of the aim point set are the same [3], making it possible to decouple the computation of $\mathbf{R}$ and $\mathbf{t}$. The computation is done as follows.

1. Move the two point-clouds so that they are centered at the origin:
   $\mathbf{q}^j = \mathbf{x}^j - \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i \qquad \forall j = 1..N$
   $\hat{\mathbf{q}}^j = \hat{\mathbf{x}}^j - \frac{1}{N}\sum_{i=1}^{N} \hat{\mathbf{x}}^i \qquad \forall j = 1..N$

2. Calculate the 3×3 matrix
   $\mathbf{H} = \sum_{i=1}^{N} {\mathbf{q}^i}^T \hat{\mathbf{q}}^i$

3. Find the SVD of $\mathbf{H}$
   $\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^T$

4. Calculate the possible solution
   $\mathbf{R} = \mathbf{V}\mathbf{U}^T$

5. Check whether the solution is valid
   If $\det \mathbf{R} = +1$ then the solution is valid
   If $\det \mathbf{R} = -1$ then the algorithm fails

6. Compute the translation
   $\mathbf{t} = \frac{1}{N} \sum_{i=1}^{N} \hat{\mathbf{x}}^i - \mathbf{R}\left(\frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^i \right)$

With that result in hands it is now possible to compute the least-squares best motion between frames. This is a good tool to handle noise, but outliers seriously affect the correctness of the computation. Therefore robust methods have to be applied to discard the outliers.

5.3.2 Achieving robustness against outliers

Robust estimation is a common problem in computer vision. A wide variety of literature exists on this subject. An introduction to estimation techniques is given in [39]. More detailed descriptions are for example available in [24, 37]. In the beginning the RANdom SAmpling Consensus (RANSAC) method was used. Later it was discarded because it tended to produce bent maps and had problems with estimating rotation correctly. Therefore another method was used, that uses the singular value decomposition iteratively, leaving out the worst points after each application. This method works well for rotations, but has problems estimating the translation correctly if rotation is present. Therefore the two methods were combined, resulting in a better algorithm.
The RANSAC-algorithm was developed to deal with a large fraction of outliers in the data set. In contrast to a least-squares fit that uses as much data as possible, it is based on a estimation of the parameters from a minimal data set. Because a rotation and a translation in 3D are to be estimated this minimal data set contains 3 points. This is sufficient to specify the motion uniquely. The idea of RANSAC is to test a large number[6] of minimal sets and to check the resulting estimate for support among the other data points. The estimate with the most support is kept and a re-estimation takes place with a least-squares technique including all data points that support the best estimate. A simple situation is shown in figure 5.4. The regression line for the data set is sought. A least squares estimate would be severely affected by the two outliers in the set. For a line estimate the minimum data set consists of 2 data points. One of those and the resulting estimate for the line is shown in the figure. All data points that fall in the region of support, support this estimate. The two outliers and one correct data point are discarded. If this was the best estimate, a least-squares re-estimation

[6] How many depends on the probability of an outlier to occur in the data set. The number of tests is usually specified so as to ensure a 95% probability of a set with no outliers occuring during random sampling.

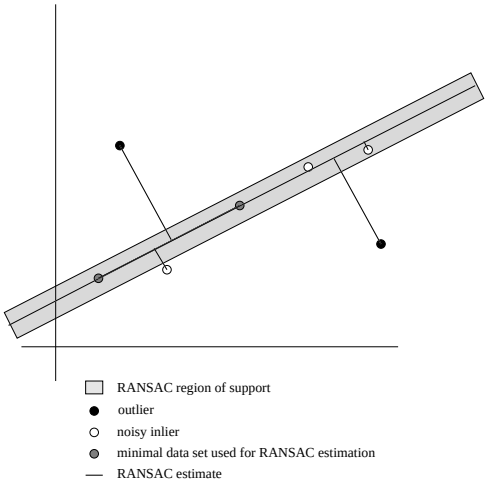

Figure 5.4. Robust line estimation with RANSAC

using the 4 supporting data points, would be carried out after all testing has been finished to obtain the robust estimate.
For the application of this algorithm it is necessary to specify the region of support and the minimal point set. The region of support was defined as spherical regions around each of the points in the aim data set. To check an estimate for support the estimated motion is applied to all points in the source data set[7]. The resulting points are checked against their corresponding regions of support. The choice of spheres as region of support is not the best choice, but simple to compute. The regions should ideally be formed like ellipsoids, that describe the uncertainty in the data point (see figure 3.2), taking into account the error covariances of the data points.
The RANSAC method was found to estimate the translation between to successive camera positions quite well, but showed deficiencies in the estimation of the rotational part of the movement. Therefore another method, based on the M-estimator was considered. The method consists of a repeated weighted least-squares estimate. After each iteration the data points are re-weighted dependent on the error they produce. For simplicity a step function in the range of 0 and 1 was used. Therefore it was possible to use the least-squares estimate of section 5.3.1, successively leaving out the data points that have the biggest distance from their counterpart after applying the estimated motion. As stopping criterion for the iteration a measure based on the standard deviation and the maximum-error after the first estimate was used. If the mean of the current estimates error falls below a threshold times the original mean the iteration is aborted. This was mo-

[7]The source data set is the data set that is to be translated and rotated to fit the second (aim) data set

tivated by the observation that gross outliers influence the mean stronger than the standard deviation. If the maximum error is large but the standard deviation low, it can be expected that a narrow band of "good" matches is disturbed by a number of gross outliers and therefore the original mean error should be reduced considerably, whereas a standard deviation and a maximum of comparable size suggest the outliers to lie embedded into the band of "good" matches. Those cannot be removed without discarding good correspondences. The threshold for stopping the iteration was computed as the introductory standard deviation[8] of the error divided by the introductory maximum of the error. This was found to be too lax. Therefore the square of the maximum error was tried, which in turn was too restrictive, discarding in some cases too many good matches as well. That lead to the use of a power function in between.

$$\text{thresh} = \frac{\sigma}{(\max|\text{error}|)^n}$$

n is the only threshold that has to be set now. A value of 1.7 was found to work satisfactory. This method performed good for rotations, but the translations in case of a combined translation/rotation motion were estimated badly. Because of that the two methods were combined in the final version of the stereo motion estimation algorithm, first performing a rotation estimate with the M-estimator, followed by a RANSAC translation estimation using the rotated data points. The performance of this combined algorithm is evaluated in chapter 6.
This section covered the steps necessary to find the transformation that relates a point cloud in the current camera coordinate system to that of the previous frame. This is needed because the point clouds are represented in their respective camera coordinates. The motion computed in this section relates these two camera coordinate frames, making it possible to register points from both coordinate systems in a common one, that is used to accumulate the final map.

5.4 Motion accumulation

The next step is to accumulate the single motions between frames and compute a full path. The motion of the previous section can be written in homogeneous coordinates:

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

Transformation of 3D-points from one frame to the previous one reads now

$$X^{(0)} = \mathbf{M}^{(0)} X^{(1)}$$

[8] the one of the first estimate

and further to transform the second frame to the zeroth

$$\begin{aligned} X^{(1)} &= \mathbf{M}^{(1)} X^{(2)} \\ X^{(0)} &= \mathbf{M}^{(0)} \mathbf{M}^{(1)} X^{(2)} \end{aligned}$$

In general the transformation from the Nth coordinate frame to the zeroth (with the product carried out with right multiplication) is:

$$X^{(0)} = \left(\prod_{i=0}^{N} \mathbf{M}^{(i)} \right) X^{(N)}$$

This representation is widely used because it allows to express rotation and translation in one matrix expression [11]. Nevertheless there are some problems. When accumulating a large number of motions, the accumulated motion matrix looses orthonormality in the $\mathbf{R}$ part due to roundoff errors. This introduces shears into the transformation. Pure rotations have to be represented by an orthonormal matrix, therefore re-normalization (for example using the Gram-Schmidt orthogonalization) is necessary from time to time. This problem occurs frequently in computer graphics and if re-normalization is to be avoided a quaternion representation should be used. With the accumulated motion it is now possible to relate every timestep's local camera coordinate system to a global (the first camera's) coordinate system. This allows for the transformation of the 3D point clouds, that were computed from the disparity map into one common coordinate system. Once transformed into the global coordinate system, the point clouds are inserted into the octree data structure of chapter 4 where they build clusters of occupied regions. With that step the map building algorithm is complete and can be used for iterative map generation. Although the first camera position's coordinate system is used as global frame, the accumulated error present in the global map accumulates not infinitely[9]. This is because the region covered by the octree is moved with the robot and previous data is forgotten. Thus the error present in the actual map is restricted to the error that accumulated since the insertion of the "oldest" element that is still present in the tree.

[9] At least if the robot moves in a way that forces the represented region to be adapted

Chapter 6

Experiments and results

This chapter presents some experiments that were conducted to determine the utility of the suggested methods. Unfortunately major deficiencies were discovered for the (rotational) motion estimation part. Since motion estimation is a crucial step for obtaining a good fit of the single map pieces when using dead reckoning, the reasons for the failing of the suggested algorithms were explored and a different approach based on the sensors of the robot was used to determine the motion, therewith demonstrating the usefulness of the rest of the system. The experiments include a forward-backwards test to determine the accuracy of reconstruction for straight movements and a 90 degree turning test with different radii to find out about the performance of reconstructing combined translational and rotational movements of the camera rig. Those are the two movements that can occur on the robot platform (see figure 6.1). A rotation will always be combined with a lateral translation since the camera centres are displaced from the center of rotation of the robot. The difficulty of separating translational and rotational components of a combined motion, and therefore the difficulty of estimating a combined

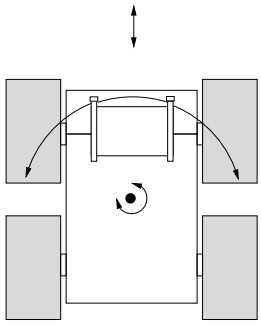

Figure 6.1. The figure shows the possible robot motions and the resulting motion of the camera rig

motion correctly, will be shown. Pitch and roll[1] were not considered, mainly because of the necessity to fuse sensory data from a number of sensors other than odometry to get an estimate of the real movement. This research is ongoing at the institute at the moment. Furthermore the results obtained for yaw should apply to pitch as well because the estimation problem has a similar structure in that case (see section 6.2.2 for the problems and 6.3 for the reasons). Roll movements on the other hand should not pose a serious problem. This will be explained in section 6.3 as well. A test that demonstrates the importance of camera calibration is included as well. All tests are based on real outdoor image data, but restricted to a movement in a plane (a parking lot) to allow the use of odometry data as "ground truth" data. This is reasonable because the odometry data is much more accurate than the stereo-reconstruction of motion.
The odometry was biased when performing the tests, therefore the angles for the turning experiment where scaled to fit the expected values. This might introduce some error into the error estimation of the viewing direction because the robot was controlled by hand when recording the data.

6.1 Straight forward and back

This experiment consisted in driving the robot distances of 2,3,4 and 5 meters forward and back to the starting position. The aim was to find out about the influence of the accumulated error and the accuracy of estimating a forward translation (which is a forward-upwards translation in camera coordinates, since the camera is looking downwards to the ground). The test area was a parking lot with a fence. The fence was used to keep the right direction and to add some structure to the scene. Furthermore some material (leafs and earth) was distributed over the test area to add some texture to the otherwise nearly featureless asphalt. The plots in figure 6.2 show the results using the iterative least squares estimation technique for the 2m,3m and 4m run in the upper, middle and lower section respectively. In each section a number of tests for different displacements of the camera between the images of the video sequence where performed to demonstrate the effects of different speeds of movement. This was achieved by simply skipping some of the pictures. The displacements correspond to use every second, forth and sixth frame in the video sequence. The sequence was recorded with 6 frames per second with a driving speed of 0.1 meters per second. The upper row of plots shows the reconstructed path projected onto a plane[2], while

[1] Roll, pitch and yaw are the three Euler angles. When putting an airplane in the origin of a coordinate system in direction of the positive z-axis then roll corresponds to a rotation around the z-axis, pitch around the x-axis and yaw around the y-axis

[2] To make the three-dimensional reconstructed motion and the two-dimensional odometry information comparable the reconstructed path was projected onto the least squares fit plane through the reconstructed camera positions. Because the movement should yield

the lower row contains corresponding plots for the absolute distance of the reconstructed camera positions from that plane. The two plots on the right side show the path reconstructed from odometry information and the absolute positional error of the 3 cm displacement experiment to the odometry reconstructed path in each time step. This is the overall accumulated error.

The analysis of those plots yields two insights. The smaller the displacement between two successive images, the noisier the reconstructed individual motion becomes. This is due to the influence of the resolution of the images and will be explained in section 6.3. If the displacement in image coordinates is only a few pixels the influence of the region of uncertainty (see figure 3.2) becomes the dominating factor for the reconstruction error. If the displacement on the other hand becomes too large, the feature matching step becomes more unreliable. The assumption that the intensity neighborhood for a feature point stays the same in two different images of that point is violated and an affine transformation model as mentioned in section 2.4 is a better choice.

Figure 6.3 shows two reconstructions of the test area. The upper map is reconstructed using motion from odometry information, while the lower map uses stereo information alone. The map derived from stereo information is noisier and even has some artifacts caused by errors in the disparity map that could not be removed by filtering, but the main structure of the scene is reconstructed correctly. The size of a region at the lowest level in the octree-representation was about 8 cm^3. As a result of this test it can be stated that the reconstruction accuracy concerning straight movements is reasonable for the intended task of maintaining a map of about 5m surrounding area, if the robot is not staying too long in one area (for example moving back and forth all the time) and is not moving too fast. The border of feasibility lies between 7 and 10 cm of displacement between frames. It proofed, that reasonable performance for the map building task in the case of translational movement can be reached using stereo vision alone. While not perfect the reconstruction of the path is accurate enough to keep a map in a neighborhood of about 5m correct for translational forth and back movements when staying in a certain corridor for the displacement between successive frames.

---

a line, the x-axis was chosen to be the other basis vector of the plane. This corresponds to the fact that the baseline between the cameras is parallel to the ground.

[3] 5m × 5m × 5m covered by the complete tree,maximum of 6 subdivisions

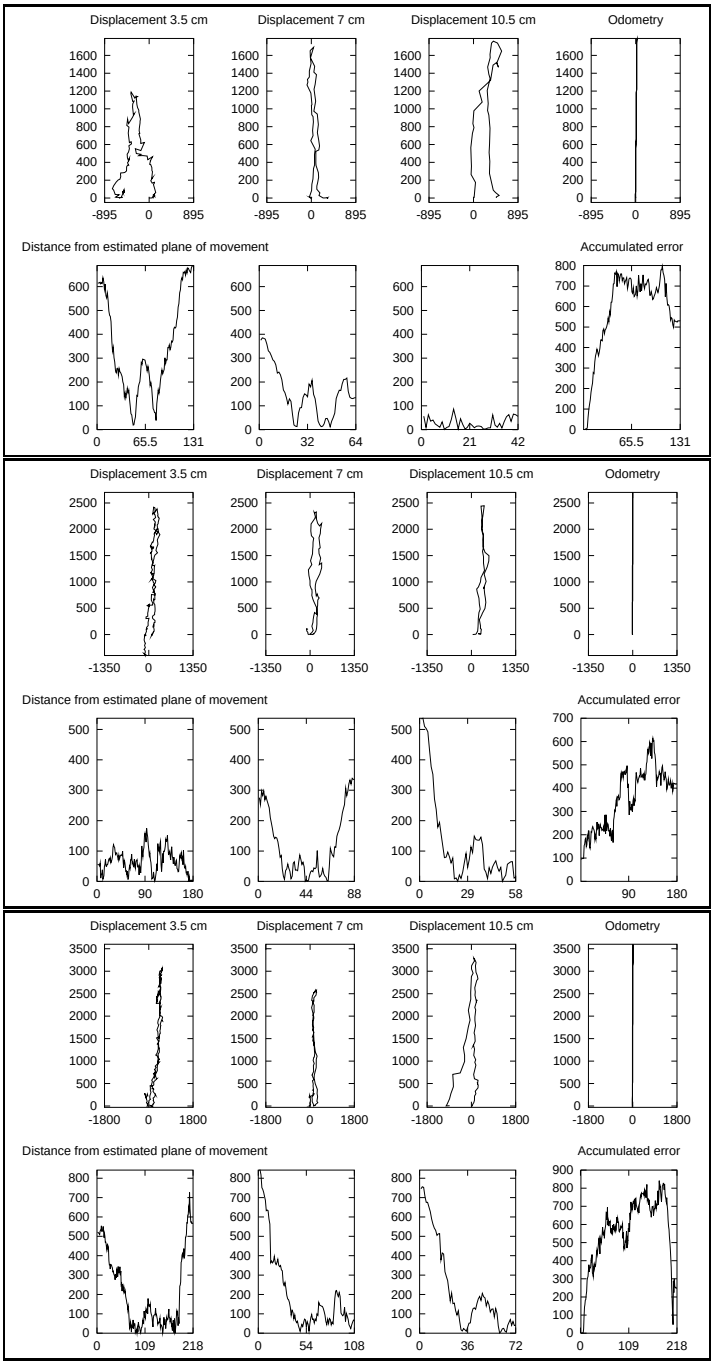

Figure 6.2. Path reconstruction for the straight forward and backward test. The plots show the results for driving 2m (upper) 3m (middle) and 4m (lower). The upper row depicts the reconstructed path in the plane of movement and the lower row the distance from that plane in every timestep. The accumulated error plot shows the absolute difference in position between the 3.5 cm displacement per timestep and the odometry information. All distances are in mm.

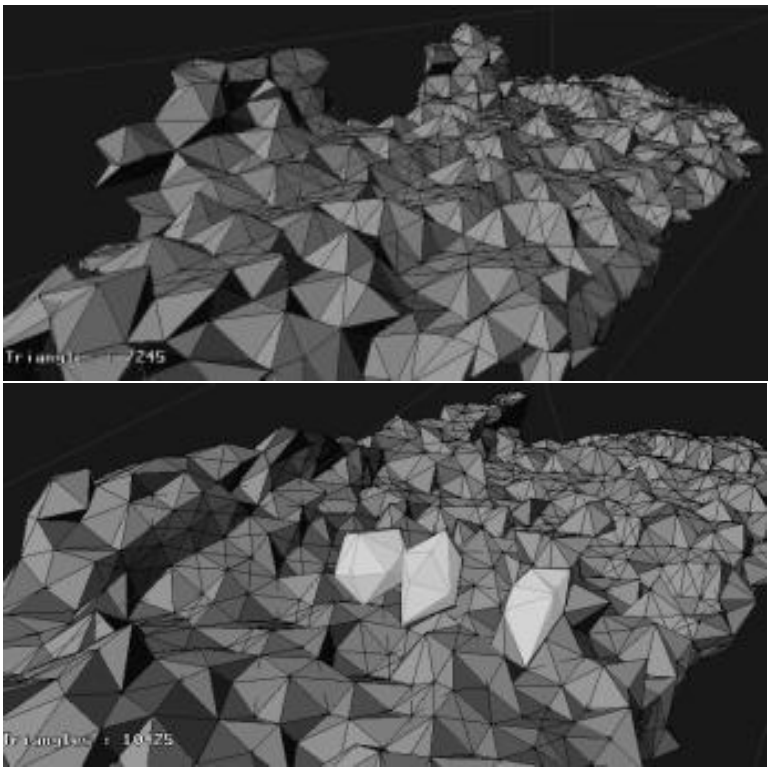

Figure 6.3. Reconstructed map of the area that was used for the straight forward test. It shows a parking lot with a fence (≈ 50 cm high) on the left side. The two bumps are poles of the fence surrounded by some plants. The rest of the fence could not be extracted by the boundary extraction algorithm, because there are no inner nodes, but boundary nodes present in the area that is connecting the two poles. The upper picture was done using odometry information from the robots wheels, whereas the lower picture used a motion computation from stereo pictures alone. The lower picture has some artifacts (lightened up) caused by noise in the disparity map that do not correspond to physical features. They are not connected to the main structure.

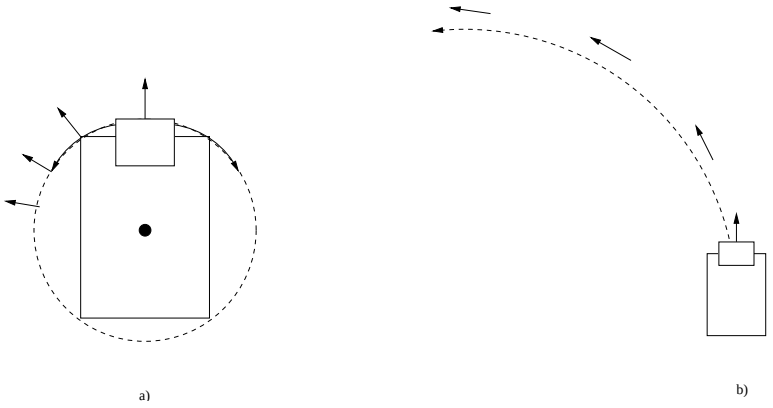

Figure 6.4. The different turning cases for small a) and big b) radii respectively

6.2 90 degrees turn - different radii

The motion occurring in the turning test is much more difficult to estimate. The reasons for that will be explained later in section 6.3. As mentioned before, the turning experiment deals with combined translations and rotations. Two fundamentally different cases occur for different turning radii as demonstrated in figure 6.4. For small radii, the extreme case being turning on the spot, the translation is nearly orthogonal to the viewing direction. The feature points have approximately the same distance to the observer after the rotation. This is the most difficult situation , because the rotation gets confused with a lateral translation. The other case is a long stretched arc. Here the direction of movement is nearly parallel to the viewing direction[4]. This situation is much nearer to the straight forward movement and easier to handle. The test area was the same as for the straight movement experiment.

6.2.1 Large turning radius

Figure 6.5 shows the results of the turning test for a large radius (3m). The speed of movement was 0.2 meters per second. Recorded was again with 6 fps i.e. the displacements in the four cases are 3.5, 7, 10 and 20 cm. The first row shows the reconstructed path with viewing direction in every reconstructed camera position. As a comparison the odometry information is plotted in the same graph. The middle row of plots shows the accumulated error in the 3 space directions and the last row the accumulated angular error of the viewing direction. The jump in angular error for the 3.5 cm displacement case is due to outlier-influence.

[4]projected into two dimensions

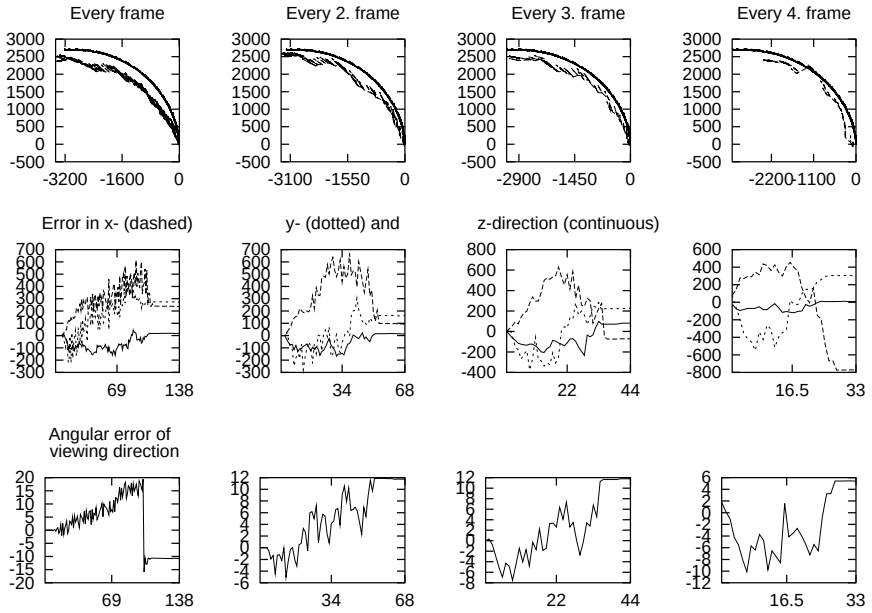

Figure 6.5. Experimental results - large turning radius (3m), displacements and errors are in mm, angles in degree

The results show, that for a large turning radius (case b) of figure 6.4) the performance of the algorithm is comparable to the case of straight movement. The estimates are reasonable for displacements smaller than 10 cm between successive frames. Therefore the same conclusions as in section 6.1 apply. Figure 6.6 shows the border case, where the type of combined movement changes to situation a) of figure 6.4. The speed of movement was $0.12\frac{m}{s}$ here and the corresponding displacements 2,4,6 and 8 cm. It can clearly be seen, that the estimations reliability decreases and the sensitivity to large displacements increases. The viewing direction is as expected not tangential to the curve anymore. The estimation of the viewing direction stays at the same error level as for the large radius case.

6.2.2 Small turning radius

For turning on the spot the stereo-algorithm failed to reconstruct the camera path correctly as can be seen in figure 6.7. The reason for that will be shown in the next section. It is nevertheless noteworthy, that the rotational component of the motion is estimated with approximately the same accuracy as in the other two experiments. Furthermore the direction of the translation behaves as expected for case a) of figure 6.4. The translations occur perpendicular to the viewing direction nearly everywhere. That means the qualitative behavior of the motion estimate is locally correct. It should be noted as well, that the error is much stronger in the y-direction of the

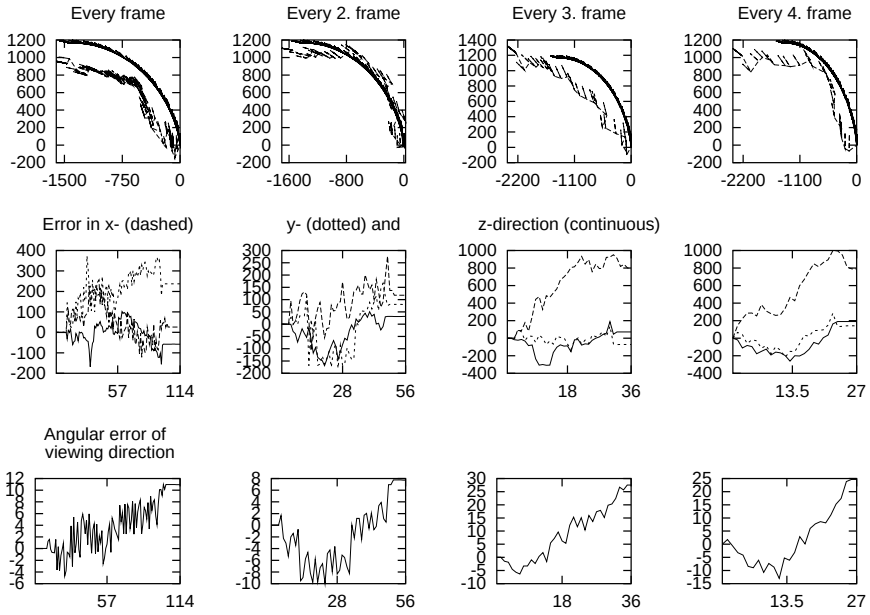

Figure 6.6. Experimental results - medium radius (border case - radius 1.5m)

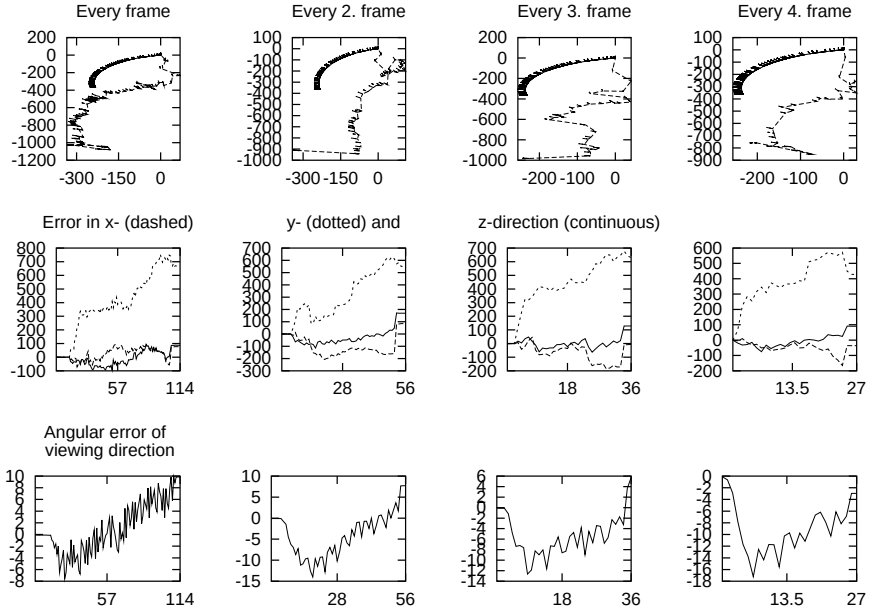

Figure 6.7. Experimental results - small radius ($\approx 0.3m$) for turning on a spot

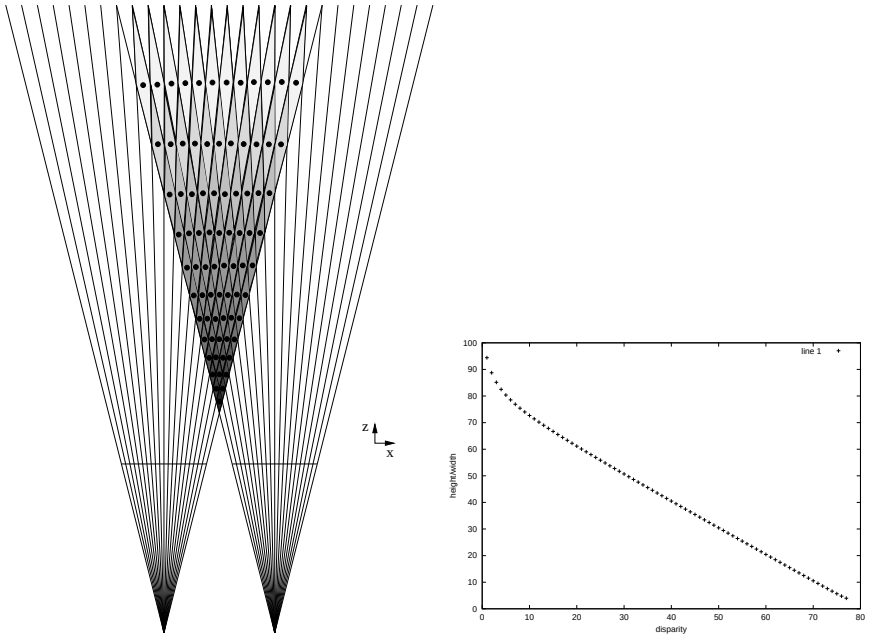

Figure 6.8. On the left side a parallel camera setup with the levels of disparity visualized in 2D. On the right side the quotient between the height and the width of the region of uncertainty versus disparity level.

diagram than in x-direction.
The turning test revealed that a general problem in the motion estimation using 3D- to 3D-point correspondences[5] exists. It is less obvious in the case of large translations along the viewing axis, but introduces large errors when approaching the case of lateral translation combined with rotation.

6.3 The resolution problem

The main problem for the estimation of motion from 3D- to 3D-point correspondences, that were acquired using stereo vision, was found to be the shape of the region of uncertainty (see figure 3.2 for individual reconstructed points. The situation for the whole field of view is depicted on the left side of figure 6.8. There are two problems that can be pointed out. First all points in a region of uncertainty map to the center of that region. Therefore only a fixed number of discrete z-values exists in the final set of reconstructed 3D-points. Second the regions of uncertainty are very elongated in z-direction. A plot for the stereo rig used in this thesis is shown on the right side of figure 6.8. The curve shows the quotient between height and width of the region of uncertainty for increasing disparity. Usually the disparity values

[5] with range data obtained from stereo reconstruction

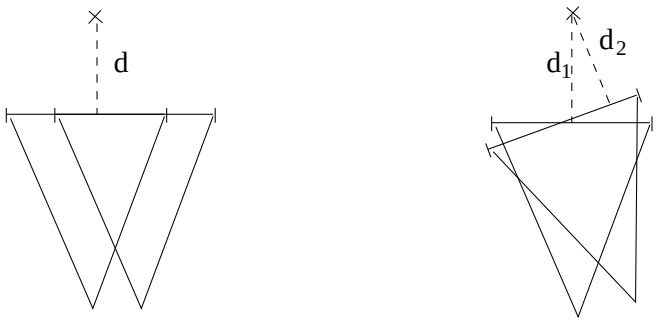

Figure 6.9. Rotation - translation ambiguity

computed from the stereo pictures are below $\frac{1}{4}$ the image size, i.e. ≈ 30 pixels for the setup used in this thesis. This yields a factor around 50-100. The consequences are, that pixel noise has much more effect in z-direction than in x and y.
There is a quadratic dependency between depths and errors of the reconstruction [5].

$$\begin{aligned} \mathbf{X_3} &= \frac{bf}{d} \\ \frac{\partial \mathbf{X_3}}{\partial d} &= -\frac{bf}{d^2} \\ &= -\frac{\mathbf{X_3}^2}{bf} \end{aligned}$$

Consider a plane perpendicular to the viewing direction. The movement towards that plane has to be at least so large, that one disparity level is crossed, otherwise the movement cannot be detected by the stereo-algorithm. Fortunately the reconstructed points are usually spread over the different disparity levels, making it possible to use their mean in order to determine translations that lie between those jumps. Unfortunately the fraction of points crossing a disparity level is not related to the movement that is to be interpolated. Because the original points can be anywhere in the region of uncertainty it is random if a point crosses a disparity level. Because of that the reconstructed paths for small displacements between successive frames of the sequence are noisier than the ones with a bigger displacement. The error due to reconstruction uncertainty of the points is added once for every motion computation step. Therefore crossing several levels of disparity reduces the influence of this error considerably. For rotational movements the reconstruction gets more prone to error because the main difference between a lateral translation and a rotation is the change of a points distance to the image plane (see figure 6.9). The observable movement in the image plane denoted by i is nearly identical. The problem is now, that the difference be-

tween d_1 and d_2 is low for small rotations. Therefore it should be expected, that points do not cross disparity levels and the rotation gets mistaken for a lateral translation. Tendencies for that can be seen in the small radius rotation experiment (section 6.2.2). Surprisingly the rotation is estimated correctly, but the magnitude of the translation is overestimated, destroying the correct estimation of the path.
After this analysis it is possible to guess the outcomes of a pitch and roll experiment. Pitch will suffer from the same problem as the yaw rotation - the movement will be confused with a vertical translation. Roll on the other hand should be determinable with much better accuracy, because the movement of the point takes place in planes of equal disparity and the x- and y-extensions of the region of uncertainty are much smaller than the z-extension. Of course this argument should be supported by further experiments.
The problem described here is caused by the poor discrete disparity resolution of the system. To remedy this situation the use of sub-pixel accurate feature detection using the OpenCV [2] routines was tried, but no significant change in behavior was achieved. Instead sometimes nearly half of the feature points disappeared, disturbing the following estimation considerably. Therefore sub-pixel accurate feature detection was left out of the final algorithm. Another option is to increase the resolution of the images. While not enhancing the quotient between height and width of the region of uncertainty, at least the absolute number of disparity levels is increased, probably resulting in a better estimate. Unfortunately doubling the resolution quadruples the computational effort. If only yaw - rotations are of concern it could be tried to increase the resolution in x-direction only, but as soon as pitch is to be estimated as well the full resolution has to be increased.

6.4 Influence of radial distortion

This experiment uses the same test data as the large radius turning experiment. The only difference between the two is the missing correction of radial distortion of the cameras. The results are included because it was expected that the radial distortion for a camera with 6 mm focal length would be negligible, but turned out to have a significant influence on the estimation of rotation.

The plots show a serious under-estimation of the rotational component of the motion. This experiment shows the significance of correct camera calibration. it suggests that the adhoc approach (see section 3.5) used to calibrate the cameras for this thesis project might not be sufficient and that better results can be obtained when using carefully calibrated cameras.

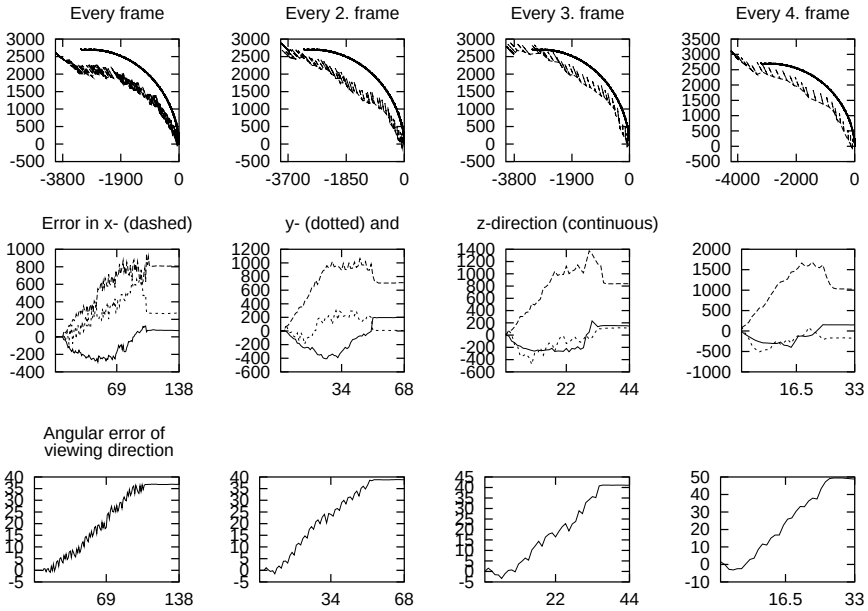

Figure 6.10. Experimental results for the large radius turning experiment with uncorrected radial distortion

6.5 Some final experiments

In this section some maps acquired in an unaltered environment are presented. The three situations of figure 6.11 cannot be detected by other sensors currently installed on the robot platform. Certain areas in the visualization of the maps where lightened up or darkened artificially to emphasize interesting parts. In the first situation the robot approaches a ramp. The ramp rises to 50 cm elevation in 1.5 m, the right side of the ramp stays at the same elevation level. The darkened area in the first map corresponds to the discontinuity in height. The discontinuity is softened out by different sampling positions of the local maps integrated into the tree, but still recognizable. A path planning algorithm would probably use an elevation gradient measure to decide if an area is passable or not. To avoid areas like the one presented here, such an algorithm would have to use a conservative threshold for the gradients in elevation, as they tend to be smeared out due to sampling reasons in sampling positions of the local maps as well as in the octree itself. The second example is a steep inclination, that poses a danger for the robot of falling down. The reconstructed map is hard to visualize because a lot of empty spaces exist. The left picture of the middle situation of figure 6.11 shows the reconstructed map from a similar perspective as the photograph. The plane the robot is moving in was lightened up so it can be differentiated from the wall, that is behind it. The lower large picture shows the same map seen from a position looking frontal at the wall. The two

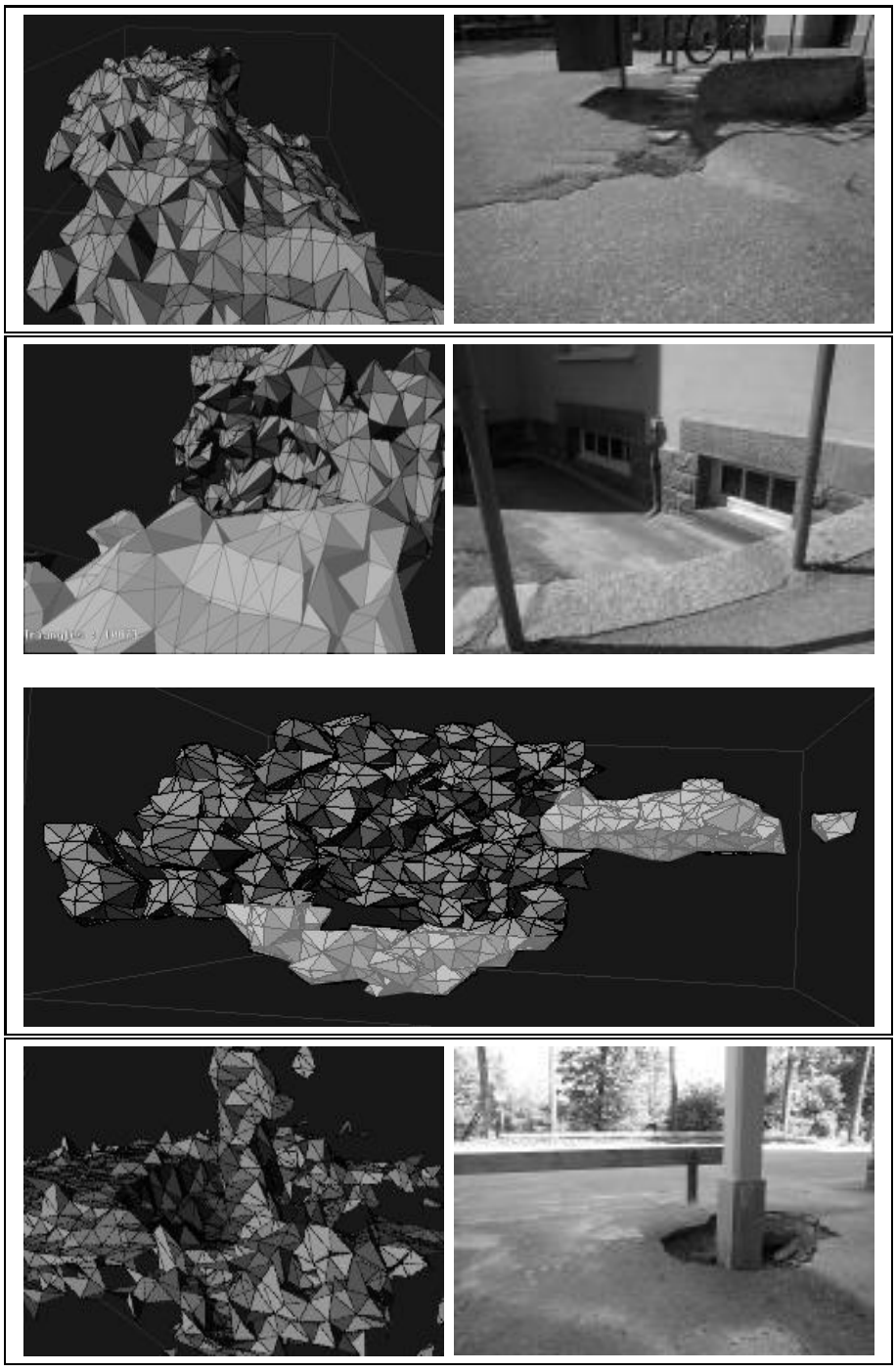

Figure 6.11. Some scenes and their corresponding maps

light areas are two planes, the upper one is the one, the robot is moving in whereas the lower plane is the ground below the inclination. The rest of the picture is covered by the wall. In 3 dimensions a large free space can be seen between the upper and the lower plane, “representing” the discontinuity in height. A path planning algorithm should not let the robot move into areas marked as unknown in the map. This leads to difficulties as well, because sometimes insufficient texturing causes holes in the map.
The last situation is a pole with a surrounding hole. While the pole should be recognized safely, trying to avoid it the robot could fall with one or more wheels into the hole. The scaling of the pictures of the maps is not the same as in the photographs. For example in the pole sequence the plane is extended to the left side outside the picture, because the robot was approaching from the left. The visible part was chosen such as to present the interesting parts of the reconstructed maps.
A result of those experiments is that the reconstruction can be made as good or bad as wanted by choosing the environment in a certain way. Low textured areas are missed in the reconstruction because the disparity maps yield ambiguous areas, whereas carefully chosen areas like in the accuracy experiments yield good reconstructions, at least if the motion features reasonable displacements in depth. The critical point of all computations is the ability to establish correct matches between frames (between the left and right image, if motion is computed from sensor information and additionally in time if motion is to be computed from stereo information alone). The matching between the left and the right picture is necessary to determine the placement of the (fixed) range of disparity search. This placement is needed because disparities outside this range result in random disparity estimates, causing the map fusion algorithm to integrate random points. Therefore even when using sensors to determine the motion between frames it is needed to have a textured environment.

Chapter 7

Conclusions and future work

7.1 Conclusions

As a result of this thesis it can be stated, that stereo vision is a good modality for three-dimensional map building, but suffers from some significant shortcomings. That suggests that this method should not be used alone, at least not if not sufficient time/computing power is available and some special cases cannot be excluded.
The cases posing difficulties to stereo vision are insufficient amount of texture in the images and pitch and yaw rotation with small radii. In the first case the feature matching will fail and random movements will be estimated, resulting in useless maps. The second case leads to a systematic error of overestimated lateral translations with the same result of a useless map. Also the motion is restricted to lie in corridor of "allowed" ranges. If the motion is too small, pixel noise dominates the estimation, while the motion being too large causes random matches which degrades the performance of the motion estimation as well. While being quite successful in providing the structure of a scene, the estimation of motion from stereo sequences is unreliable[1] and computationally expensive. Another insight is that much attention has to be put into preprocessing of images and calibration of the camera rig.
The aim of the project was to provide a map of the nearer environment of the robot for local navigation purposes in real-time. The goal of real-time performance was not reached, but the system performs in near real-time (between 1s and 2s per frame). This includes the motion from stereo algorithm. When using additional sensory data, this time falls to somewhere between 0.5s and 1s. While not optimizing the implementation to the limit, some care was taken in the implementation of the algorithm to assure efficiency of execution. The main time of the algorithm is used in the motion estimation.

[1] In a sense of being much noisier than other sensor information and prone to special movements

Next expensive is the integration of the local maps into the octree. The rest is disparity computation and frame grabbing. On the other hand the goal of map building was achieved. A framework for local mapping is in place, a good and extendible data structure - the three-dimensional occupancy octree was developed. The region covered by the tree can be moved, making it possible to continuously follow the robots movements and update the map of the local environment, "forgetting" information after some time to restrict memory consumption. The motion from stereo problem was analyzed and the cause of the problems identified. To demonstrate the utility of the rest of the algorithm tests using odometry information were performed.

7.2 Future work

At this point the work can be developed into two directions. One is to create a general robot-independent stereo vision system. This could be done through improvements of all parts of this work. Such a system should preferably work offline, extending the possibilities to use more advanced techniques. The current work was, due to performance reasons, restricted to use the simplest methods available. Especially investigation into sub-pixel feature detection and advanced camera calibration techniques is promising. For map registration the dead reckoning approach should be replaced by a method that fits the local maps in a best way to all previous map pieces, not only the last one. Another difficulty encountered during the work was lack of features in the images, resulting in bad, unreliable matching of feature points, leading to bad estimates of the 3D-motion. In those cases the disparity map was still of acceptable quality. It could be tried to match the disparity maps directly. Difficulties are to be expected, because the point clouds obtained from the disparity maps are unstructured. Neighboring pixels do not have to lie on the same physical object, making it difficult to triangulate the point clouds. Another difficulty is the establishment of correspondences between the two maps. The advantage of using direct disparity map matching would be the extended range of textures that could be treated. Even asphalt areas should be possible to map. The disadvantages are high computational costs and ambiguities introduced by special scene structure, e.g. the movement on a plane cannot be determined unambiguously.
The other direction is a system, fit to a special robot platform. Here all available sensors should be used to obtain an accurate (at least much more accurate than from stereo vision alone) estimation of the motion. This motion estimate replaces the one used in this thesis and the developed framework can be applied. Since the integration of the local map into the octree is the most time-consuming part of the resulting system, either the resolution of the octree should be restricted or the use of a quadtree, using a

$2\frac{1}{2}$D representation should be considered. With those measures taken it should be possible to do a real-time mapping for planar movement of the robot. If mapping is desired to identify dangerous obstacles alone, another high-speed possibility would be to consider histograms of the disparity maps only. If a certain number of points fell below a minimum threshold or above a maximum one, the situation could be considered dangerous and the robot could be stopped or slowed down, starting a more thorough analysis of the environment, using techniques developed in this thesis. This seems to be a promising approach for an applicable system for robot navigation and obstacle avoidance.

Appendix

Mathematical annotations

Homogeneous coordinates

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

is an 2D homogeneous vector with $x = \lambda x \quad \forall \lambda$ except $\lambda = 0$.
A point x with $x_3 = 0$ represents a direction vector, whereas vectors with $x_3 \neq 0$ refer to positions in 2D space, namely:

$$\mathbf{x} = \frac{1}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

A line l is described by a homogeneous 2D vector as well. $a\mathbf{x_1} + b\mathbf{x_2} + c = 0$ defines a line in normal 2D space. Equally holds $l_1 x_1 + l_2 x_2 + l_3 x_3 = 0$ in 2D homogeneous space. This can easily be verified by dividing by x_3.
This can conveniently written $l^T x = 0$ using the scalar product.

A line l connecting two points x and $\tilde{x}$ is written:

$$l = x \times \tilde{x}$$

The point x at the intersection of two lines l and $\tilde{l}$ is given by:

$$x = l \times \tilde{l}$$

Matrix representation of cross product

The cross product of $\mathbf{a} = (\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3})^T$ and $\mathbf{b} = (\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3})^T$ is defined as $\mathbf{a} \times \mathbf{b} = (\mathbf{a_2 b_3} - \mathbf{a_3 b_2}, \mathbf{a_3 b_1} - \mathbf{a_1 b_3}, \mathbf{a_1 b_2} - \mathbf{a_2 b_1})^T$.
This can be written in matrix form as:

$$[\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -\mathbf{a_3} & \mathbf{a_2} \\ \mathbf{a_3} & 0 & -\mathbf{a_1} \\ -\mathbf{a_2} & \mathbf{a_1} & 0 \end{bmatrix} \mathbf{b}$$

Camera models

for the projection $x = \mathbf{P}X = \mathbf{K}\mathbf{M}X$ ($\mathbf{K}$ is the camera calibration matrix, $\mathbf{M} = [\mathbf{R}|T]$ is the rigid body transformation of the camera). It is assumed, that the homogeneous vectors X are normalized to a homogeneous component X_4 of 1.

1. **ideal pinhole camera**

$$\mathbf{K} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2. **pinhole camera with focal length**

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

3. **pinhole camera with focal length and shifted optical axis**

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

4. **pinhole camera with non-square pixels**

$$\mathbf{K} = \begin{bmatrix} a_x f & 0 & a_x p_x & 0 \\ 0 & a_y f & a_y p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

   a_x:a_y is the aspect ratio

Similarity measures

1. **standard cross-correlation** measures how much of one vector projects onto the other one, returns values $0 \leq C_1 \leq 1$, has to be maximized:

$$C_1(x, y) = \frac{\sum_i^n x_i y_i}{\sqrt{\sum_i^n {x_i}^2 \sum_i^n {y_i}^2}}$$

2. **zero mean cross-correlation** same as 1 but with normalized vector values, returns values $-1 \leq C_2 \leq 1$, has to be maximized:

$$C_2(x, y) = \frac{\sum_i^n (x_i - \overline{x}_i)(y_i - \overline{y}_i)}{\sqrt{\sum_i^n (x_i - \overline{x}_i)^2 \sum_i^n (y_i - \overline{y}_i)^2}}$$

3. **sum of squared differences** measures the euclidian distance between two vectors, has to be minimized:

$$C_3(x, y) = \sum_i^n (x_i - y_i)^2$$

4. **sum of absolute differences** measures the L_1 distance between two vectors, has to be minimized:

$$C_4(x,y) = \sum_i^n |x_i - y_i|$$

5. χ^2 **test** is a statistical measure of the similarity of two distributions, has to be minimized:

$$C_5(x,y) = \sum_i^n \frac{(x_i - y_i)^2}{\frac{(x_i+y_i)}{2}}$$

Determining connectivity of two boxes

To determine if two rectangles overlap or not the sum of the edge lengths of the original rectangles is compared to the edge lengths of the big (dotted) rectangle (figure 1).

$$\begin{aligned} s_1 &= x_s + \hat{x}_s \\ s_2 &= y_s + \hat{y}_s \\ t_1 &= \max(|\hat{x}_e - x_a|, |x_e - \hat{x}_a|) \\ t_2 &= \max(|\hat{y}_e - y_a|, |y_e - \hat{y}_a|) \end{aligned}$$

The comparisons give rise to the following results

$(s_1 > t_1) \wedge (s_2 > t_2)$	if rectangles overlap with area
$(s_1 = t_1) \wedge (s_2 = t_2)$	if the corners are connected
$(s_1 = t_1) \vee (s_2 = t_2)$	if at least one edge is shared
$(s_1 < t_1) \vee (s_2 < t_2)$	if rectangles are separated

and therefore

$(s_1 \geq t_1) \wedge (s_2 \geq t_2)$	with equality for *at least* one pair	for 8-connectivity
$(s_1 \geq t_1) \wedge (s_2 \geq t_2)$	with equality for *exactly* one pair	for 4-connectivity

This can easily be extended to higher dimensions. Now X is a vector and the equations read:

$$\begin{aligned} s_i &= X_s^i + \hat{X}_s^i \\ t_i &= \max(|\hat{X}_e^i - X_a^i|, |X_e^i - \hat{X}_a^i|) \end{aligned}$$

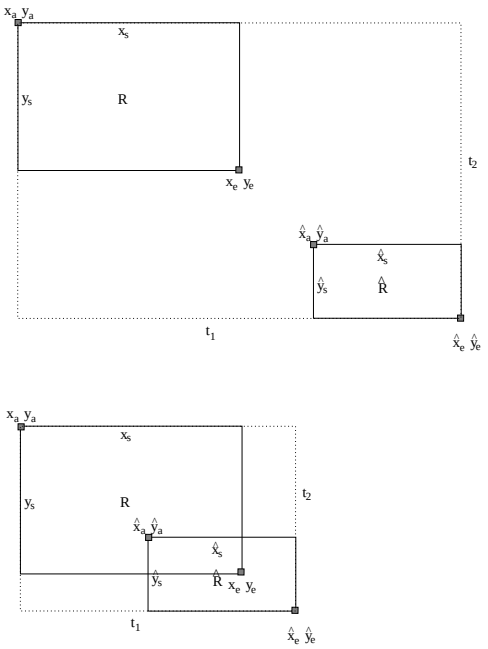

Figure 1. Not overlapping (upper) and overlapping rectangles (lower)

The conditions are now formulated as:

$$\bigwedge_i (s_i > t_i) \qquad \text{if boxes overlap with area}$$

$$\bigwedge_i (s_i = t_i) \qquad \text{if the corners are connected}$$

$$\bigvee_i (s_i = t_i) \qquad \text{if the boxes have at least one common edge}$$

$$\bigvee_i (s_i < t_i) \qquad \text{if the boxes are separated}$$

and equivalently

$$\bigwedge_i (s_i \geq t_i) \quad \text{with equality for } \textit{at least} \text{ one pair} \qquad \text{for } (3^n\text{-}1)\text{-connectivity}$$

$$\bigwedge_i (s_i \geq t_i) \quad \text{with equality for } \textit{exactly} \text{ one pair} \qquad \text{for } 2n\text{-connectivity}$$

Bibliography

1. IA-32 Intel Architecture Software Developer's Manual, volume 2: Instruction Set Reference. developer.intel.com, Order number 245471, 2001.
2. Open Source Computer Vision Library. www.intel.com/research/mrl/research/opencv/, 2001.
3. K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE transactions on pattern analysis and machine intelligence*, PAMI-9(5), September 1987.
4. A. Baumberg. Reliable feature matching across widely separated views. Canon Research Centre Europe Ltd., 2000.
5. M. Björkman. *Real Time Motion and Stereo Cues for Active Visual Observers*. PhD thesis, Royal Institute of Technology, 2002.
6. I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, May 1996.
7. U. R. Dhond and J. K. Aggarwal. Structure from stereo - a review. *IEEE transactions on systems, man, and cybernetics*, 19(6), November/December 1989.
8. C. Doran. Geometric algebra and computer vision , lecture notes. Conference on Clifford Algebras and their Applications in Mathematical Physics, Mexico; www.mrao.cam.ac.uk/~clifford, June/July 1999.
9. O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation-based stereo: algorithm, implementations and applications. Rapport de recherche, INRIA, 1993.
10. T. Fitzgibbons and E. Nebot. Application of vision in simultaneous localization & mapping. Australian Centre for Field Robotics University of Sydney, 2002.
11. Foley, van Damm, Feiner, and Hughes. *Computer Graphics: Principles and Praxis, second edition in C*. Addison-Wesley, 1997.
12. S. Gull, A. Lasenby, and C. Doran. Imaginary numbers are not real. - the geometric algebra of spacetime. University of Cambridge, February 1993.
13. R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

14. D. Hestenes. Vectors spinors and complex numbers in classical and quantum physics. *American Journal of Physics*, 39(9), September 1971.

15. A. Heyden and K. Rohr. Evaluation of corner extraction schemes using invariance methods. Dept. of Mathematics Lund University FB Informatik University of Hamburg, 1996.

16. B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4), April 1987.

17. D. F. Huber and M. Hebert. A new approach to 3-d terrain mapping. The Robotics Institute, Carnegie Mellon University, 1999.

18. IEEE International Conference on Robotics and Automation. *Matching of Affinely Invariant Regions for Visual Servoing*, May 1999.

19. A. E. Johnson. *Spin-Images: A Representation for 3-D Surface Matching.* CMU-RI-TR-97-47, Carnegie Mellon University, 1997.

20. A. E. Johnson, O. Carmichael, D. Huber, and M. Hebert. Toward a general 3-d matching engine: Multiple models, complex scenes, and efficient data filtering. Jet Propulsion Laboratory California Institute of Technology and The Robotics Institute Carnegie Mellon University, November 1998.

21. K. Konolige. Small vision systems: Hardware and implementation. Artificial Intelligence Center, SRI International, 1997.

22. T. Lindeberg. Lecture notes datorseende. KTH, 2001.

23. L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. Carnegie-Mellon University, 1988.

24. P. Meer, D. Mintz, and A. Rosenfeld. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6(1), 1991.

25. M. Obregon. Columbus' first landfall. Phileas Conference November 1989 Ft. Lauderdale, December 1989.

26. P. Pritchett and A. Zisserman. Wide baseline stereo matching. University of Oxford, 1998.

27. S. Roy and I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. *Sixth International Conference on Computer Vision (ICCV '98)*, pages 492–499, January 1998.

28. H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison - Wesley Publishing Company, Inc., 1990.

29. F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. University of Oxford, 2001.

30. C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. INRIA Rhône-Alpes, 1997.

31. C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. INRIA Rhône-Alpes, 1997.

32. W. B. Seales and O. D. Faugeras. Building three-dimensional cad/cam models from image sequences. University of Kentucky and INRIA Sophia-Antipolis, 1994.

33. J. Shi and C. Tomasi. Good features to track. 1994.

34. S. Singh and B. Digney. Autonomous cross-country navigation using stereo vision. CMU-RI-TR-99-03, Carnegie Mellon University, January 1999.

35. P. Smith. Motion analysis of video sequences. First year report, University of Cambridge Department of Engineering, May 1998.

36. R. Szeliski and R. Zabih. An experimental comparison of stereo algorithms. Microsoft Research, Redmond, WA and Cornell University, Ithaca, NY, 1999.

37. P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 1996.

38. G. Xu and Z. Zhang. Epipolar geometry in stereo, motion and object recognition. Kluwer Academic Publishers, 1996.

39. Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Rapport de recherche no.2676, INRIA Sophia-Antipolis, October 1995.

40. I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. 1997.