Unterschrift (Betreuer)

DIPLOMARBEIT

Real-time 3D Reconstruction for Autonomous Football Playing Robots Using a Feature Based Stereo Approach

ausgeführt am
Institut für Rechnergestützte Automation,
Arbeitsbereich für Mustererkennung und Bildverarbeitung
der Technischen Universität Wien

unter der Anleitung von
a.o. Univ. Prof. Dipl.-Ing. Dr. Robert Sablatnig

durch

Harald Entner
Margarethen Straße 88-90 / 44
A-1050 Wien

09. Januar 2005

Unterschrift (Student)

Abstract

This diploma thesis addresses the topics of stereo 3D reconstruction and shows the results of a feature based real-time stereo algorithm. For understanding issues, it uses the pinhole camera to describe perspective projection. In reality, a lens is used to project the light rays onto the sensor chip, thus an extended model is presented in order to describe the projection using a lens and to introduce camera calibration. In a dynamic environment stereo from motion is not possible, as a result a multi-camera system is needed. Epipolar geometry describes this special geometry and can be useful to minimize the computational cost of searching corresponding points in two images. The feature extraction uses the Canny algorithm followed by a new iterative line detection. Straight lines are considered to be an appropriate representation of the football field and the robots. After features have been found the correspondences are searched and used to calculate the 3D position of the features. Chapter 2 covers the theoretical background, whereas Chapter 3 describes the implemented methods. In Chapter 4 every part of the algorithm is evaluated. It is followed by the conclusion and the future plans.

Acknowledgements

The author wants to thank

Robert Sablatnig who reviewed this work and helped to reach the final result.

My Parents for supporting me during my studies.

Bernhard Koch for proof-reading this thesis.

Contents

1 Introduction and Overview 2

1.1 Objective of this thesis . . . 4

1.2 Overview . . . 5

2 Stereo Vision 6

2.1 Perspective Projection . . . 10

2.1.1 The Pinhole Camera . . . 11

2.1.2 Camera Model with Lens . . . 13

2.2 Camera Calibration . . . 16

2.3 Stereo Geometry . . . 18

2.3.1 Epipolar Geometry . . . 19

2.3.2 Fundamental Matrix . . . 20

2.3.3 Rectification . . . 21

2.3.4 Occlusion . . . 22

2.3.5 Constraints . . . 23

2.4 Correspondence Analysis . . . 24

2.4.1 Intensity-based correspondence analysis . . . 26

2.4.2 Feature-based Correspondence Analysis . . . 30

2.5 3D Reconstruction . . . 32

2.6 Summary . . . 34

3 Methods 36

3.1 Image Acquisition . . . 38

3.2 Calibration . . . 40

3.3 Edge detection . . . 43

3.4 Line Extraction . . . 47
3.5 Correspondence Analysis . . . 53
  3.5.1 Search for corresponding lines . . . 54
  3.5.2 Object detection . . . 56
3.6 3D Reconstruction and Visualization . . . 57
3.7 Robot Head . . . 59
3.8 Summary . . . 62

4 Results 63

4.1 Calibration . . . 64
4.2 Line Detection . . . 68
  4.2.1 Synthetic data . . . 69
  4.2.2 Real data . . . 71
4.3 Correspondence Analysis . . . 72
  4.3.1 Synthetic data . . . 72
  4.3.2 Real data . . . 73
4.4 3D Reconstruction . . . 75
  4.4.1 Synthetic data . . . 76
  4.4.2 Real data . . . 77
4.5 Summary . . . 79

5 Conclusion and Future Work 80

Bibliography 82

Chapter 1

Introduction and Overview

> Wie fang' ich nach der Regel an,
> Ihr stellt sie selbst, und folgt ihr dann.
>
> (Richard Wagner, "Die Meistersinger")

When a camera takes a shot from a scene, the light-rays reflected at the objects inside the scene form an object on the sensor chip of the camera. During this transformation the 3D information gets lost. However, the human visual system is equipped with two eyes, each eye taking a view of the same area from a slightly different angle. The benefit of a stereo system is best voiced by Rachel Cooper in the following lines:

> "Both images have plenty in common but contains information the other doesn't. The mind combines the two images, finds similarities and adds in the small differences. The small differences between the two images add up to a big difference in the final picture. The combined image is more than the sum of its parts. It is a three-dimensional stereo picture."[1]

Leonardo da Vinci (1452 - 1519) was the first to describe binocular parallaxes [GR76]. Charles Wheatstone was the first who used a two camera system to gain *stereo* images. These images could then be viewed through a stereoscope, which he invented in the year 1838. Figure 1.1 shows an illustration of his construction.

---

[1] http://www.vision3d.com/stereo.html

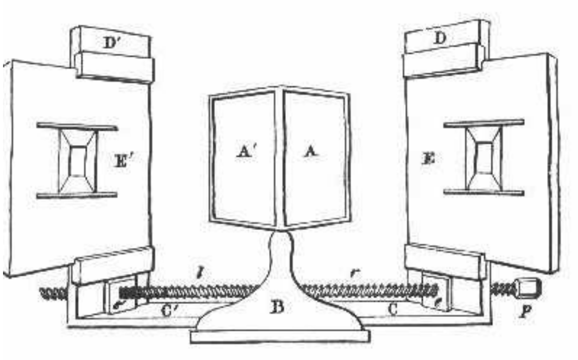


Figure 1.1: Charles Wheatstone invented 1838 the first stereoscope. It used angled mirrors A and A' to reflect the stereoscopic drawings E and E' towards the viewer. Picture is taken from "http://www.bitwise.net/ ken-bill/stereo.htm"

The occurring displacement between two corresponding image points seen by both eyes is used by the human brain to determine its 3D position relative to the viewer. This information is very useful, for example during interaction with objects of the real world. Especially when the actor has to deal with moving objects (e.g. soccer, baseball,..) 3D information is crucial.

So far a lot of sensors have been used to gain information about the periphery. In principle, one can distinguish between two groups of sensors: *active* and *passive* ones. Remote sensing systems which measure energy that is naturally available are called **passive sensors**. For example the sun provides a convenient source of energy for remote sensing. The emitted energy is either reflected, or absorbed and then re-emitted, as it is for thermal infrared wavelengths. Reflected energy is only available if a light source (or another wave emitting source) exists, thus can not be used during night (if no source is available). Energy that is naturally emitted (such as thermal infrared) can be detected at all time, as long as the amount of incoming energy is sufficient.

Active sensors, on the other hand, like infrared, ultrasonic, structured light or laser range scanners, provide their own source of energy. The sensor emits waves with a special wavelength in a given direction. The reflected waves are then investigated. The easiest form of an active sensor is an infrared bump sensor, which just measures if light is reflected or not, thus is not able to determine the distance of the object in relation to the sensor. An infrared distance sensor sends out infrared light and measures the position of the reflected light at a CCD chip and uses triangulation to get a 3D point, or the phase difference between the reflected light and the outgoing light is measured. Ultrasonic sensors send out ultrasonic signals and measure the time the signal needs to be reflected. This kind of information can be also used as distance information due to known speed of the signal. Regrettably are both methods unable to determine the exact position of the object which reflects the signal. Structured light uses the deformation of a projected pattern to gain information about the geometry or uses triangulation. It needs a sender which emits the light and a camera or other kind of receiver (depending on wavelength).

The eyes of a human being are **passive sensors**, like cameras. Cameras have lenses which project the light onto a sensor chip. The result is a 2D grayscale or color image. If two cameras are used which are positioned at different places but have an overlapping field of view, one can compute the 3D point of any point in one image if the corresponding point in the other image is found and the geometry between the cameras is known (i.e. they are *calibrated*). The search for corresponding points is one of the main problems in *stereo vision* and called *correspondence analysis*.

1.1 Objective of this thesis

The objective of this thesis is the implementation of a fast 3D reconstruction using a feature-based stereo approach. The field of application is reduced to a football field used by the FIRA[2]. Especially for an autonomous football playing robot, it is essential to have its own view of the environment. This information can be used for obstacle detection, object identification, localization in known or unknown environment and interaction. During a game of football, the actors have to face all these tasks, otherwise they would not be able to find the goal, the ball or other players. In addition, to have appropriate information, the computation of the 3D image of the environment has to be very fast. Otherwise, information will become more and more useless because when the computation is finished the image is obsolete.

---

[2] Federation of International Robosoccer Association

Beside the implementation of the stereo algorithm, a prototype of a robot head has been developed. The microcontroller (MC) inside the head is responsible for the movement of the stereo-head and some basic control cycles. It is able to send status information to the computer, and the computer can send commands to the MC in return. The circuit diagrams can be found in Section 3.7. The algorithm, or at least parts of it, should ultimately run on a DSP MC which is mounted on the *Tinyphoon* robot developed at the ICT institute[3].

1.2 Overview

In this chapter the principles of stereo vision and other sensors are presented. All sensors have in common that they can provide information about the environment. The next chapter discusses the state of the art in stereo vision and provides the mathematical background needed to understand stereo reconstruction. In Chapter 3 the implemented methods from calibration of the cameras to the reconstructed scene are described. The performance and accuracy is discussed in Chapter 4. My conclusion and future plans are presented in Chapter 5.

---

[3] see http://www.robotsoccer.at for more information on the *Tinyphoon* robot. Figure 3.11 shows a picture of the robot.

Chapter 2

Stereo Vision

Es genügt nicht, zum Fluß zu kommen
mit dem Wunsch Fische zu fangen,
man muß auch das Netz in der Hand mitnehmen.

[Chinesisches Sprichwort]

This chapter gives an overview of the mathematical background needed to understand 3D reconstruction using a stereo vision system. Section 2.1 contains information about the projection of light rays onto a sensor chip and shows the relation between the real world and the image generated at the sensor chip. An important section is on the calibration and the meaning of the intrinsic and extrinsic camera parameters. In order to gain 3D information a moving camera, a moving object or a multi-camera system is needed. To calculate 3D information in a dynamic environment, a synchronized multi-camera system is appropriate. In this thesis a two-camera system is used, the occurring geometry is explained in Section 2.3. If the geometry is known, corresponding points have to be found in order to calculate the 3D coordinates of the points. This topic is covered in Sections 2.4 and 2.5.

Stereo vision can be used in different field of applications (production line, autonomous navigating, gesture recognition [WO03] ...) where depth information is needed. It is computed through triangulation, which is the base of other 3D reconstruction techniques as well.

If objects are viewed from two different positions, it is possible to determine their position in 3D space as a result of their geometric relations (Equation 2.3). The process of the conversion of an image pair in a three-dimensional representation is called *stereo vision* according to the spatial vision of human beings. In a static environment it is possible to obtain stereo information from one camera if its position is changing and the geometric relation between two or more taken images are known. On a mobile football playing robot without a static environment, a stereo vision system with two or more cameras has to be used in order to obtain accurate stereo information. If the images are taken at the same time, a static environment is guaranteed. The goal of a stereo analysis method is the calculation of depth information due to geometric relations. In principal every method consists of these passes [MT79]:

Exposure The result of this process is mainly influenced by the resolution of the camera sensor, the used sampling frequency and the lighting conditions.

Camera calibration Determination of the intrinsic and extrinsic parameters of the camera.

Feature extraction Significant features are collected. Most important is the localisation of these elements because after solving the correspondence problem, disparity is calculated as the difference of the position.

Correspondence analysis Process of determining corresponding elements in the images. Different kinds of previous knowledge, constraints and plausibility considerations can be used to avoid false matches:

> **Search space** To an element in one image, the search for a corresponding element in the second image is restricted to a given window, normally the epipolar line (see Section 2.3.1 for more details about the epipolar line). Also it is assumed that the displacement of corresponding elements is limited.
>
> **Feature properties** If the features are distinguishable, only features with the same type (edges, corner) and the same properties (color, length) will be related to each other.

Order constraint If a correspondence is found, the plausibility of other correspondences changes. Rules for such interactions are:

- Uniqueness and Integrity: Every element in the left image, has exactly one corresponding element in the right image. A special case is monocular occlusion and transparency.
- Ordering: If a correspondence (l_i, r_j) is found, then elements which are situated on the right hand side of l_i should correspond to elements on the right hand side of r_j. The assumption in this case is that neighbouring elements are likely to have similar disparities.

Scale space The complexity of the correspondence analysis grows with the number of elements, thus often a image pyramid is used and the search starts with the highest level (lowest resolution) of the pyramid. Are the correspondences between the few elements found the search continuous with the next lower level.

Depth values calculation If the corresponding elements and the geometry of the camera is known, depth values for the extracted features can be found.

The difference in x and y direction between two corresponding elements is called *disparity*. There are different kinds of disparities that can occur, namely point disparity, point and attitude disparity, gray value disparity, photometric disparity and monocular occlusion. Figure 2.1 shows an example of each case. The different disparities can be described in more detail:

a. Point disparity A point which is shifted in vertical and/or horizontal direction. If parallel camera axes are used, only horizontal disparities occur. If camera axes converge points with horizontal and/or vertical disparities can occur, but also points without disparity. The set of points in space that are projected without disparity is called the *theoretical horopter* [Mal00].

b. Point and attitude disparities A beveled line in space is normally projected with different pitches.

c. Gray value disparities Often points can not be compared directly, because too many ambiguities would occur, thus sets of points are compared

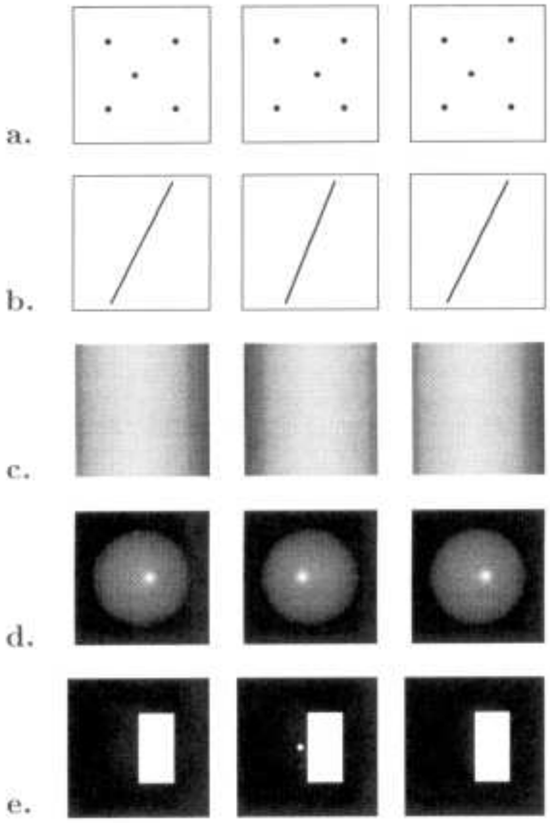


Figure 2.1: Different kinds of disparities that can occur. **(a)** point disparity **(b)** point and attitude disparity **(c)** gray value disparity **(d)** photometric disparity **(e)** monocular occlusion

d. Photometric disparities On glossy surfaces, a unique point can send out different amounts of light if the viewing position changes. This kind of disparity occurs due to the reflection properties of the surface and not due to the geometry. Equation 2.1 does only cover disparities which results from geometry (except occlusions).

e. Monocular occlusions Areas which are near an edge are often only seen by one camera and thus no corresponding element is available in the second image.

Let $I_l(\vec{x})$ and $I_r(\vec{x})$ be the left and right image. If there are no monocular occlusions the correspondence problem can be written as

$$I_l(\vec{x}) = I_r(\vec{x} - d(\vec{x})) \tag{2.1}$$

The components of the vector field $d(\vec{x})$ are the horizontal disparities. In the case of converge camera-axis $d(\vec{x})$ would be a two-dimensional vector containing the horizontal and vertical disparities. The equation above only includes a special case of stereo vision. Some more problems are:

Occlusion The constraint of a totally visible surface for both eyes is rather special hence $d(\vec{x})$ is not continuous.

Transparency If the surface is transparent one point of $I_l(\vec{x})$ corresponds with more points of $I_r(\vec{x})$ or vice versa.

Photometric effects Caused by reflections and not due to the geometry

Let us assume that x_l, x_r are corresponding points of the left and right image, respectively. Disparity d can be calculated as

$$d = x_l - x_r = -f\frac{b}{z} \tag{2.2}$$

This equation can be used to calculate the 3D position in the camera coordinate system of the left camera when the intrinsic parameters[1] known.

$$\begin{aligned} Z_c &= -f\frac{b}{d} \\ X_c &= \frac{-xz}{f} \\ Y_c &= \frac{-yz}{f} \end{aligned} \tag{2.3}$$

2.1 Perspective Projection

This section describes two different camera models. In the first part the relation between the world and the camera is explained by example. We use an ideal model of the camera, the so-called pinhole camera. It is the easiest model of a camera, but some basic relations between the world and the camera can be discussed. Unfortunately the pinhole through which the light enters is almost infinitely small in this model. But the smaller the hole, the lesser light enters the camera. Thus, in reality lenses are used to form the image on the sensor chip. The second model includes a

---

[1] the parameters of the inner camera geometry, see Section 2.2 for more details

lens. Regrettably, when using a lens it comes to a distortion effect whose properties have to be known in order to solve the corresponding problem and achieve accurate results.

2.1.1 The Pinhole Camera

An ideal model of a camera is the pinhole camera, as seen in Figure 2.2. This kind of camera can be imagined as a box with a pinhole, through which light enters and forms a two-dimensional image on the opposite site. A point $P = (X, Y, Z)$ in the three-dimensional XYZ-space is projected to an image-point $p = (x, y)$ in the two-dimensional xy-space (wall). If the coordinate system of the XYZ-space is aligned at the pinhole so that the Z-axis coincides with the optical axis and the image plane has its origin at $(0, 0, f)$, then the projection equations are given by

$$x = \frac{fX}{Z} \tag{2.4}$$

and

$$y = \frac{fY}{Z} \tag{2.5}$$

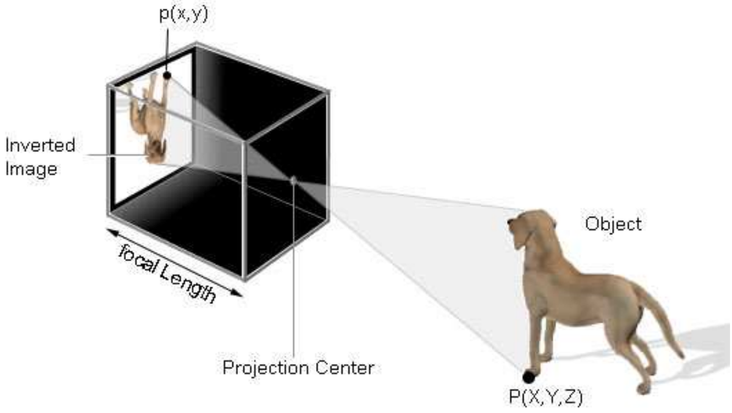


Figure 2.2: Pinhole camera model

To represent Equation 2.4 and Equation 2.5 in a linear way, we transform the point (x, y) in the Euclidean plane to a point $(x, y, 1)$ in the projective plane. This represents the same point, we simply added a new coordinate 1. Overall scaling is unimportant. The point $p = (\alpha x, \alpha y, \alpha)$ can be re-transformed by dividing through

α. Thus p is similar to $(\frac{x}{\alpha}, \frac{y}{\alpha})$. Because scaling is unimportant, the coordinate $(x, y, 1)$ is called *homogeneous coordinate.* Homogenous coordinates can also be used within a higher dimensional domain. Now we can combine Equation 2.4 and Equation 2.5 to

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{Z} & 0 & 0 & 0 \\ 0 & \frac{f}{Z} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.6}$$

If we want to know the image coordinates we have to take four more values into account. Namely

c_x, c_y image principal point, which is the intersection between the camera's optical axis and the image plane.

d_x, d_y distance between two sensor elements in x and y direction

d_x and d_y can normally be found in the datasheet[2] of the sensor chip. The image principal point has to be found by calibration (see Section 2.2 for more information). If the parameters and the point in camera coordinates are known, we can compute the image coordinates with the following formulation

$$x_i = \frac{x_c}{d_x} + c_x \tag{2.7}$$

and

$$y_i = \frac{y_c}{d_y} + c_y \tag{2.8}$$

where x_c and y_c are the coordinates of the point in the camera coordinate system. If we combine Equation 2.6 with Equation 2.7 and Equation 2.8 we can formulate the translation from Point $P(X, Y, Z)$ to the image coordinates $p(x_i, y_i)$

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{Zd_x} & 0 & 0 & C_x \\ 0 & \frac{f}{Zd_y} & 0 & C_y \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.9}$$

---

[2] technical description of a device

2.1.2 Camera Model with Lens

The pinhole camera has the problem that the pinhole should be almost infinitely small, but the smaller the pinhole, the fewer light enters the camera and the longer exposure time is needed. In a dynamic environment this is not possible because the objects in the scene are moving. As a result the pinhole camera is not used in real photography. In reality lenses are used to project the light onto the sensor chip. Figure 2.3 shows how the light-rays reflected by an object are projected through a lens. G is an object in the real world, d_{object} is the distance between G and the lens. f is the focal distance of the lens, f_k is the effective focal length (distance between the lens and the projected object B) and B is the projected object. Light is reflected at the object G and some of the light-rays enter the lens where they are refracted towards the sensor plane and form the object B. The relation between these parameters can be written in a mathematical form.

$$\frac{1}{d_{object}} + \frac{1}{f_k} = \frac{1}{f} \tag{2.10}$$

As a result of Equation 2.10, the *focal length* of the lens and the *effective focal length* are only the same if an object infinitely far away is focused. For reconstruction, d_{object} has to be approximated, thus the *effective focal length* has to be computed. As one can see in Equation 2.10, f_k strongly depends on the focused objects and should be chosen so that the objects of interest are focused. Objects that are closer or further away will not be mapped as points but as circles. This happens because the correct image point lies behind or in front of the image plane. Areas which are not focused

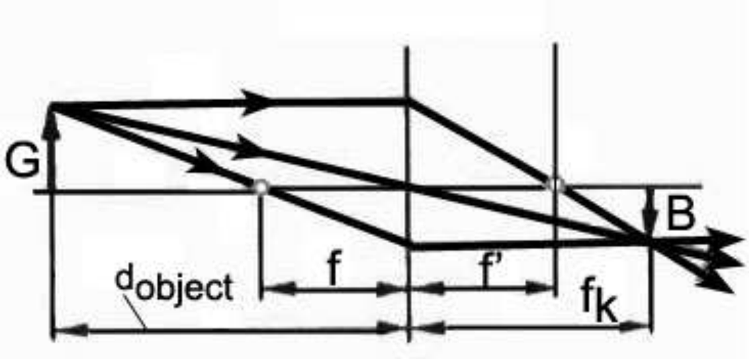


Figure 2.3: Lens Geometry

thus look blurred. There are some general laws which can help choosing the correct lens for a given problem.

- the bigger the aperture, the bigger are the circles
- the smaller the aperture, the sharper is the whole scene
- the bigger the depth sharpness, the lesser light arrives at the sensor chip
- the bigger the aperture, the smaller is the area where objects are mapped sharp

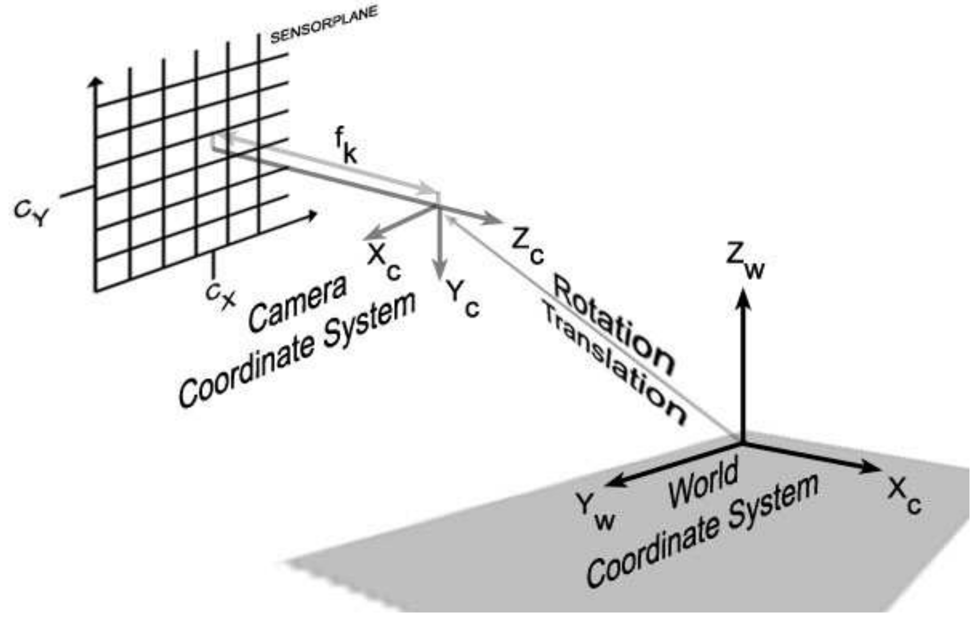


Figure 2.4: Relation between image-, camera- and world-coordinate system.

Camera parameters can be used to describe the camera geometry. It can be distinguished between two groups of parameters, namely the interior and exterior camera parameters. The interior parameters describe the camera geometry, independently from where the camera is situated. The exterior camera parameters depend only on the position where the camera is placed in relation to a given world coordinate system. Camera calibration is needed to compute both, the interior and the exterior camera parameters.

The **intrinsic parameters** are

- f_k: effective focal length

- κ_1, κ_2: lens distortion coefficients

- c_x, c_y: intersection of the optical axis with the image plane

and they determine the geometrical behavior of the projection caused by the lens used. The **extrinsic parameters** are

- R_x, R_y, R_z: rotation parameters

- T_x, T_y, T_z: transformation parameters

When a lens is used, it comes to geometric lens distortion which has no influence on the quality of the image, but it has a significant influence on the image geometry. There are two kinds of geometric distortion, *radial* and *tangential*, but radial distortion is the most general. Figure 2.5 shows the effects of radial distortion to the image geometry. It can be described as a radial outwards or inwards displacement of a point in reference to the midpoint (principle point) of the lens. Some radial distortion is inherent in most optical systems, but can be reduced by proper design. Figure 2.5a shows the effect of inwards displacement(barrel distortion). Figure 2.5b shows the effect of outwards displacement (pincushion distortion). The distortion coefficients κ_1, κ_2 describe the radial distortion and can be found using one of the calibration techniques presented in Section 2.2. Some implementations also deal with tangential distortion, even if it is often disregarded due to its negligible influence.

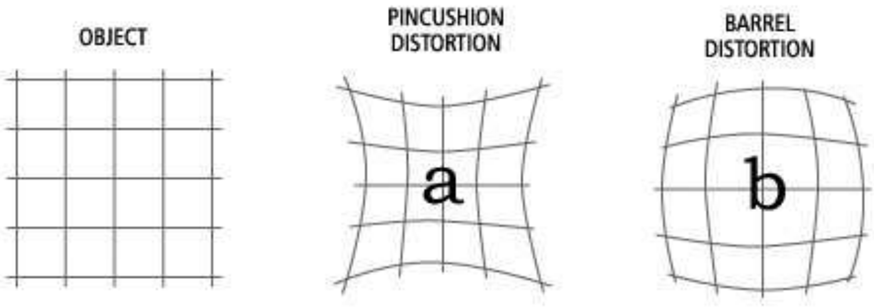


Figure 2.5: Effect of radial distortion to the geometry of an image a)barrel distortion b) pincushion distortion

2.2 Camera Calibration

Basic camera calibration is the recovery of the effective focal length f_k and the principle point c_x, c_y in the image plane or, equivalently, recovery of the position of the center of projection (x_0, y_0, f_k) in the image coordinate system. This is referred to as *interior orientation* in photogrammetry. Additionally, customary cameras have appreciable geometric distortions, but only nonlinear camera calibration techniques take them into account. If the distortion is modelled, it can be distinguished between distorted $p(x_d, y_d)$ and undistorted $p(x_u, y_u)$ image coordinates. The relation is given by

$$\begin{pmatrix} x_u \\ y_u \end{pmatrix} = \begin{pmatrix} x_d \\ y_d \end{pmatrix} + \begin{pmatrix} x_d(\kappa_1 r^2 + \kappa_2 r^4) \\ y_d(\kappa_1 r^2 + \kappa_2 r^4) \end{pmatrix} \tag{2.11}$$

where

$$r = sqrt(x_d^2 + y_d^2). \tag{2.12}$$

A calibration target is used to provide correspondences between points in the image and points in space. The relationship between the target coordinate system and the camera coordinate system is referred to as *exterior orientation.* The relation between the image- and the world-coordinate system can be written as

$$\begin{pmatrix} x_b - c_x \\ y_b - c_y \\ -f_k \end{pmatrix} = s_x \begin{pmatrix} r_1 r_2 r_3 \\ r_4 r_5 r_6 \\ r_7 r_8 r_9 \end{pmatrix} \begin{pmatrix} X_w - X_0 \\ Y_w - Y_0 \\ Z_w - Z_0 \end{pmatrix}, \tag{2.13}$$

where

- (x_b, y_b) image coordinate of the projected point (X_w, Y_w, Z_w)
- (c_x, c_y) coordinate of the intersection between the optical axis and the image plane
- s_x scale factor against the spatial sampling frequency
- f_k distance between the center of the lens and the image plane
- r_i coefficients of the rotation matrix $R = R_x * R_y * R_z$
- (X_w, Y_w, Z_w) world coordinates of the projected point

- (X_0, Y_0, Z_0) coordinates of the center of projection in world coordinates

To find a solution for Equation 2.13 we have to find corresponding image and world coordinates. Often this work is done by the calibration technique and need not be done by hand.

So far many calibration techniques have been published. According to [MK97], they can be categorized into four major techniques:

Techniques involving full scale nonlinear optimization [Tsa86, Fai75, Sob73]. The accuracy obtained by these methods is very high. But it is very time consuming due to the nonlinear optimization used.

Perspective transformation matrix using linear equation [DH73, AAK71] The benefit of these techniques is their linearity. The computation is fast, unfortunately lens distortion cannot be modelled because it is not linear (Equation 2.11). By solving an overdetermined system of linear equations the perspective camera matrix can be computed. The DLT (Direct Linear Transformation) is one example for this technique.

Two plane method [MBK71, IPS85] Only linear equations have to be solved, but the relation between the world and the camera coordinate system is assumed to be known.

Geometric techniques [FB86, LT88] Only linear equations have to be solved to get the outer orientation of the camera, but neither the focal length nor the distortion coefficients can be computed.

Normally, camera parameters have only to be calculated once, thus a nonlinear optimization techniques can be used. However, sometimes the parameters have to be calculated several times during operation, in this case a linear technique should be preferred. The two plane method makes only sense, when the relation between the world and the camera coordinate system is known and does not change during operation. In general, the decision which calibration technique to choose depends on time and accuracy constraints.

2.3 Stereo Geometry

The geometry of a monocular camera system can easily be extended to a stereo camera system. Let us assume that two cameras with an identical effective focal length f_k are used (Figure 2.6). The distance between their centers of projection is the baseline b. A point P in space is projected onto both sensor chips as x_l and x_r. As one can see, the projected points are slightly translated in x-direction. If camera axes are parallel, there are only horizontal disparities to deal with. This lets us distinguish between two different stereo geometries.

parallel camera-axes: only horizontal disparities, no points without disparity

converge camera-axes: vertical, horizontal and points without disparity can occur. The set of points without disparities is called the theoretical horopter. *Rectification* is used to transform the image so that only horizontal disparities appear. See Section 2.3.3 for more information.

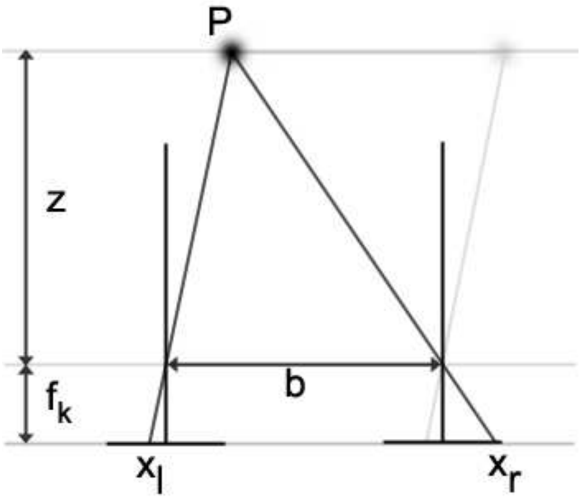


Figure 2.6: Stereo Geometry with parallel optical axes

The disparity d and the depth or z-coordinate in a camera oriented coordinate system are indirectly proportionally related. This coherence is shown in Equation 2.14.

$$d = f_k \frac{b}{z} \tag{2.14}$$

where $d = x_l - x_r$. As mentioned above, d is only one-dimensional if the camera axes are parallel. Otherwise d can be two-dimensional, i.e. vertical and horizontal disparities occur.

2.3.1 Epipolar Geometry

The epipolar geometry is the intrinsic projective geometry between two views. It is independent of scene structure, and only depends on the camera's internal parameters and relative pose. The *fundamental matrix* F encapsulates this intrinsic geometry. It is a $3x3$ matrix of rank 2. If a point X in space is mapped to one image as x_1, and to the second image as x_2, then the image points satisfy the relation $x_2^T F x_1 = 0$. F can be computed if both camera parameters are known. More about this in Section 2.3.2.

The relation between a projected point x_1 in one's view and its corresponding point x_2 in the other view can be described geometrically. In Figure 2.7, we can see that for every point x_1 a corresponding line, on which x_2 can be found, exists. These lines are called *epipolar lines.* The intersection between the baseline b and the image planes is called *epipole.* All epipolar lines intersect at the epipole.

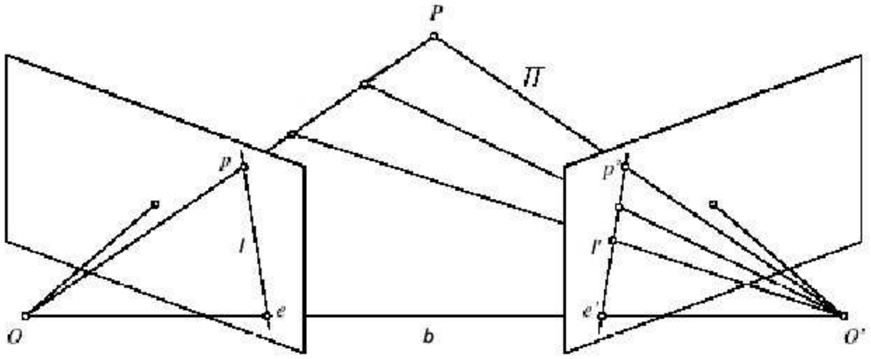


Figure 2.7: Epipolar Geometry, e and e' are the epipoles, l and l' are the epipolar lines, O and O' are the projection centers

If a point P in space is known, the corresponding epipolar lines can be found by computing the intersection between the two image planes, and the plane that is formed by the baseline b and the line from the optical center O to the point P. In Figure 2.7 the epipolar lines are denoted by l and l'. Supposing now that we know only p_1, we may ask how the corresponding point p_2 can be found. The plane Π is determined by the baseline and the ray defined by point p_1. From above we know that the point p_2 lies inside the line l' which is the intersection of plane Π with the second image plane. In terms of a stereo correspondence algorithm the benefit is that we do not have to search the entire image for the corresponding point p_2 but only along the line l'. The fundamental matrix F is the algebraic representation of the epipolar geometry and is discussed in the next section.

2.3.2 Fundamental Matrix

The fundamental matrix encodes all given geometrical constraints between a set of two stereo-images. Given a point x_1 in the first view and its corresponding point x_2 in the second view in homogenous coordinates, F fulfills the following equation

$$x_2^T F x_1 = 0 \tag{2.15}$$

F is a 3x3 matrix with rank 2. Details about the derivation can be found in [HZ00]. Some of the most important properties of F are:

Transpose If F is the fundamental matrix of a pair of cameras (C,C'), then F^T is the fundamental matrix of the pair in the opposite order (C',C).

Epipolar lines For any point x in the first image, we can find the corresponding epipolar line l' with $l' = Fx_1$ and also $l = F^T x_2$.

Epipole for any point x other than the epipole e, $l' = Fx_1$ contains the epipole e'. Thus e' satisfies $e'^T(Fx) = (e'^T F)x = 0$ for all x. It follows that $e'^T F = 0$, i.e. e' is the left null-space of F. Similarly $Fe = 0$, i.e. is the right null-space of F.

To compute F you can use corresponding points to solve the homogeneous linear Equation 2.15. The standard technique is to use pre-conditioning followed by symmetric matrix eigendecomposition to solve for the nine elements of F up to an undetermined scale factor.

In case of known camera calibration parameters the essential matrix E is the equivalent to the fundamental matrix F. In this case the rotation and translation between the cameras can be computed, up to an unknown scale factor. The images are now related by the following equation

$$x_2'^T E x_1' = 0 \tag{2.16}$$

where E is the essential matrix, x_1' and x_2' are ideal image coordinates. The ideal image coordinates x_1', x_2'have to be calculated from the original image coordinates x_1, x_2, so that x_1' and x_2' are the projected coordinates for an ideal camera. An ideal camera has focal distances $f_x = f_y = 1$, image center $x_0 = y_0 = 0$ and homogenous z-coordinate $= 1$. Thus the camera calibration matrix is the identical matrix I_{3x3}. The essential matrix can be written as

$$E = R[T]_x \tag{2.17}$$

where R is the rotation matrix, T is the translation vector between the two cameras and $[T]_x$ is defined as

$$[T]_x = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix} \tag{2.18}$$

For more information about the essential matrix have a look at [Fau93, HZ00].

2.3.3 Rectification

As said in the beginning of this chapter, using a standard geometry (parallel optical axes) has the benefit that only horizontal disparities can occur and thus the corresponding point for a point $p = (x, y)$ in the left image can only be found in the right image having the same y-coordinate as point p. Unfortunately it is almost impossible to mount the cameras in a perfectly parallel way. As we heard in the last section, the fundamental Matrix F describes the relation between the two image planes and the corresponding point p' of p can be found on the epipolar line l'. The process of *rectification* transforms the input image in a way that the epipolar lines are horizontal and thus no vertical disparities can occur. The search for a corresponding point is thus along the horizontal scan line in the other image. Figure 2.8 shows four images. The top left shows the left image with some points marked. The top right image shows the corresponding epipolar lines and the bottom images shows the result of rectification. In the bottom right image the epipolar line is plotted. One can see that it is horizontal now.

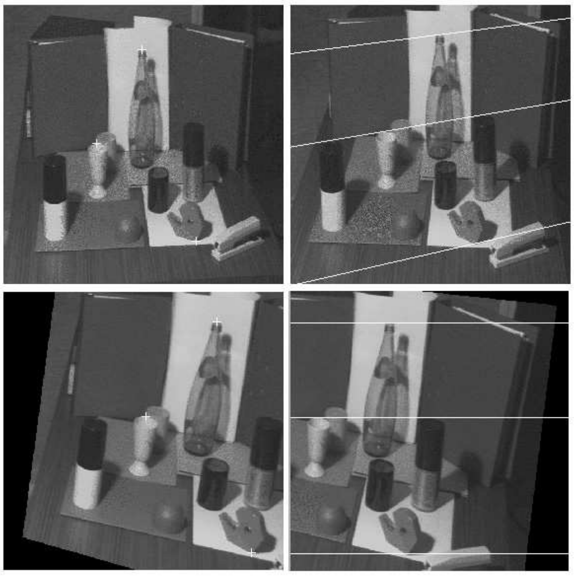


Figure 2.8: From left to right, top to bottom. 1.) left image with some points marked 2.) right image with corresponding epipolar lines of the marked points in the left image. 3.) rectified left image. 4.) rectified right image with plotted epipolar line.

2.3.4 Occlusion

When objects are viewed from two different positions, it is highly likely that some areas are occluded, thus these areas are only seen by one camera. The detection of occluded areas is an important problem in stereo vision and complicates the generation of 3D data. Figure 2.9 depicts this situation. The left camera is not able to see what is directly situated on the right side of the object. The same situation occurs for the right camera on the left side of the object. In Figure 2.9 only a small area is occluded, but when the object would be translated more into the view of the

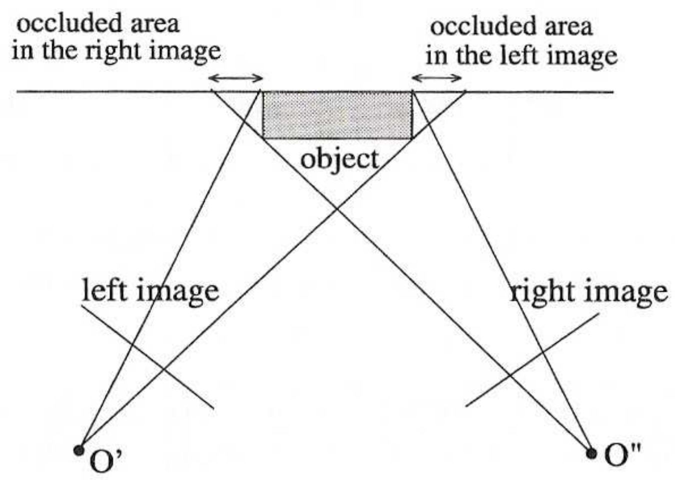


Figure 2.9: Multi-camera system with occluded areas shown.

right camera the left camera may not see the object at all. This would be the worst case scenario, but it illustrates the influence of occlusion, especially concerning the correspondence analysis. For occluded areas, no depth value can be computed.

2.3.5 Constraints

Stereo matching is a difficult problem, especially when the images are noisy. But also with perfect images, ambiguities occur during the matching process, thus for one point in the one image there may be more points in the other image. In order to minimize false matches, matching constraints must be imposed. The following list shows some commonly used constraints.

Similarity, Photometric constraint For an intensity-based approach (see Section 2.4.1 for more information), the matching pixels must have a similar intensity value. This constraint holds for an object with nearly lambertian surface and a parallel camera system. Also only an ambient light source should exist otherwise reflections and highlights would occur.

Epipolar constraint Generally, the search for a corresponding point has to be made over the whole image. This is not true if the epipolar constraint is used. For a point in one image, the corresponding point in the other image must lie on the epipolar line. Thus the search is $1D$ only.

Uniqueness[MT79] A given pixel or feature in one image can only correspond to at most one pixel or feature in the other image. This constraint does not hold for transparent or occluded areas.

Continuity[MT79] The disparity should vary smoothly almost everywhere

Ordering[BB81] If $m \leftrightarrow m'$ and $n \leftrightarrow n'$ and if n is on the right of m, then n' should also be on the right of m' and vice versa. The ordering fails at regions known as forbidden zone (Figure 2.10).

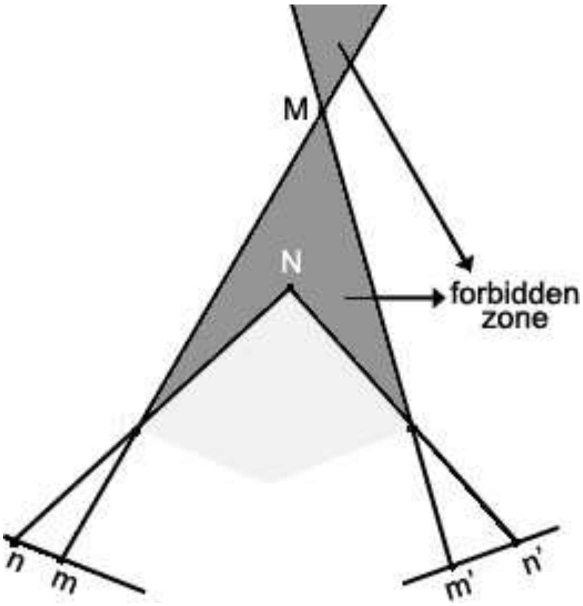


Figure 2.10: The ordering constraint fails, if a given 3D point (N here) falls onto the forbidden zone of another 3D point (M). The ordering in the right image is the reversed version of the ordering in the left image.

Disparity limit In most intensity-based approaches not the whole epipolar line is taken into account when searching for corresponding points, thus a limit for the disparity is set. For a point $p = (x, y)$ in the left image only points $p' = (x + d, y)$ in the right image are compared, where $d \in [1 \ldots maxdisp]$.

2.4 Correspondence Analysis

The previous section showed how to determine depth from disparity. But the calculation of corresponding features in both images is an additional problem which has to be solved. Corresponding features are projected points in both images that have the same object point in space. If the image features would be indistinguishable, then having n features would led to $n!$ correspondences which would all have different spatial interpretations. As soon as the correspondence problem is solved, the spatial representation of a feature can be computed through triangulation. The search for corresponding points in both images is an ill-posed problem[3].

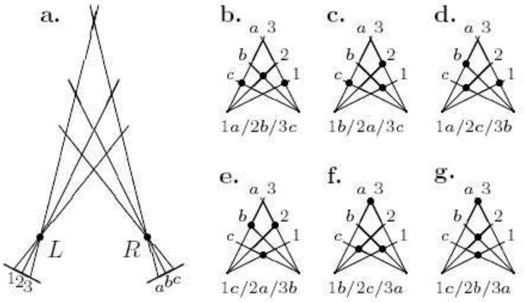


Figure 2.11: Correspondence problem of stereo vision **a.** In the left image 1,2,3 are features in the right image a,b,c. To these features there are $3! = 6$ possible correspondences. (b. - g.), which conform to the requirements of uniqueness and integrity. Every correspondence leads to another interpretation of the problem.

Figure 2.11 shows the famous double-nail illusion (Krol & van de Grind 1980, von Campenhausen 1993). Also the human visual system is not able to solve the correspondence problem always correctly, this is best understood by example. If one holds two needles in the centre plane of the eyes one behind the other and covers the ends with a bezel, the needles appear next to each other. In this case the image of the front needle in the left eye is merged with the needle in the back in the right eye. The corresponding problem is reasonable solved, even if it is physically false.

In principal there are two different approaches to solve the corresponding problem, namely *intensity-based* and *feature-based* correspondence analysis. The former approach compares the grayscale distribution between the two images, the latter extracted features. Both techniques have their advantages and disadvantages and

---

[3] Hadamard stated three criteria for a well-posed mathematical problem: 1.) a solution should exist; 2.) the solution should be unique 3.) the solution should depend continuously on the input data.

will be discussed in the next chapters.

2.4.1 Intensity-based correspondence analysis

Intensity-based approaches compare intensity distributions between two images. Figure 2.12 shows two images and four intensity profiles. The distributions correspond to the black lines marked in the images. One at row 80 and the other at row 230. In fact, a close look at the intensity profiles from the corresponding rows of the image pair reveals that the two intensity profiles differ only by a horizontal shift and a local foreshortening. Figure 2.12a and Figure 2.12b depict the images taken with a camera that undergoes a displacement in the horizontal direction, the image pair therefore corresponds to a parallel camera set up. The same situation can be reached with any stereo system using rectification (see Section 2.3.3 for more information). The result of an intensity-based method is a dense disparity map $d(x)$, thus for every pixel its disparity is known. Unfortunately there are also occlusions

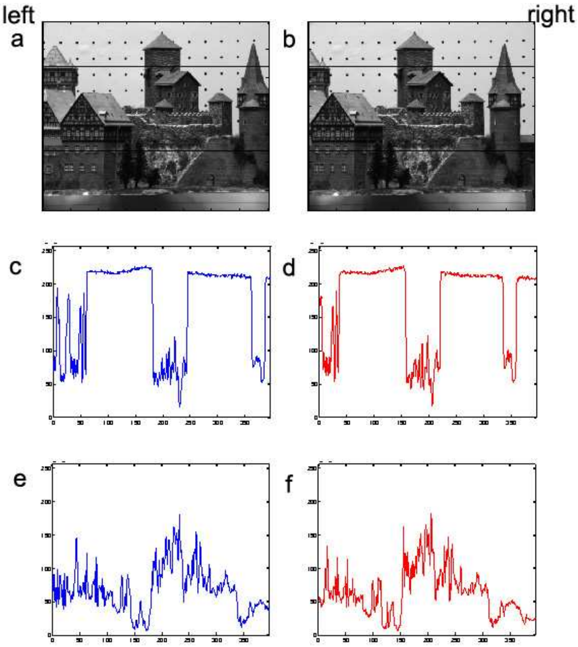


Figure 2.12: The image pair shown above is retrieved from the Calibrated Imaging Laboratory of CMU. They are taken from a parallel stereo configuration

in stereo images, for this parts of the image d(x) are undefined or set to a special value which represents an occluded area.

There are several ways to find a solution for the corresponding problem. Two approaches will be mentioned: correlation and phase difference.

Correlation

Disparity d, in this case, is the relative displacement between two grayscale distributions. For every pixel $p = (k, l)$ in the left image, a window with size mxn centered at the actual pixel is compared with a window which is centered at pixels $p' = (k + d, l + d)$ along the epipolar line. Figure 2.19 shows the shift of the window in the right image in case of a standard geometry. In this case $p' = (k + d, l)$.

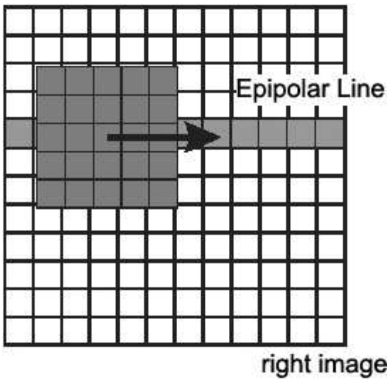


Figure 2.13: Correlation based similarity - the window is shifted along the epipolar line

The similarity with the highest correlation is chosen. The correlation is defined as

$$C(d) = \frac{\sigma_{lr}^2}{\sqrt{\sigma_l^2 + \sigma_r^2}} \tag{2.19}$$

where

$$\begin{aligned} \sigma_l^2 &= \sum_{i=k+d}^{k+m+d} \sum_{j=l}^{l+n} \frac{(I_l(i,j) - \mu)^2)}{mn} \\ \sigma_r^2 &= \sum_{i=k}^{k+m} \sum_{j=l}^{l+n} \frac{(I_r(i,j) - \mu)^2)}{mn} \end{aligned} \tag{2.20}$$

are the variances of the intensity values of the left and right image and

$$\sigma_{lr}^2 = \sum_{i=k}^{k+m} \sum_{j=l}^{l+n} \frac{(I_l(i+d,j) - \mu_l)(I_r(i,j) - \mu_r))}{mn} \tag{2.21}$$

is the covariance. The correlation is at its maximum if the variance is minimal and the covariance is maximal. A lower threshold Γ is chosen, which decides whether the similarity is strong enough or not. Thus

$$d(x) = \begin{cases} d & \text{if } C(d) > \Gamma \text{ and } d = \arg\max_d C(d) \\ \infty & \text{else} \end{cases} \tag{2.22}$$

Because correlation takes a lot of time to compute, the sum of squared difference (SSD) is often used instead. Equation 2.23 shows the equation of SSD.

$$SSD(d) = \sum_{i=k}^{k+m} \sum_{j=l}^{l+n} (I_l(i+d,j) - I_r(i,j))^2 \tag{2.23}$$

If the SSD is at its minimum, the best match has been found. In this case Γ is an upper threshold and defines whether the result is small enough or not.

$$d(x) = \begin{cases} d & \text{if } SSD(d) < \Gamma \text{ and } d = \arg\min_d SSD(d) \\ \infty & \text{else} \end{cases} \tag{2.24}$$

Intensity differences in high contrast areas are more reliable than in low contrast areas. A possible solution is to normalize Equation 2.23 with the local variance.

$$SSD(d) = \frac{\sum_{i=k}^{k+m} \sum_{j=l}^{l+n} (I_l(i+d,j) - I_r(i,j))^2}{\sigma_l^2 \sigma_r^2} \tag{2.25}$$

Another problem is that cameras often have different sensitivities. A solution for this problem is to normalize the image with variance and intensity. Equation 2.25 would look like

$$SSD(d) = \sum_{i=k}^{k+m} \sum_{j=l}^{l+n} [\frac{(I_l(i+d,j) - \mu_l)}{\sigma_l^2} - \frac{(I_r(i,j)^2 - \mu_r)}{\sigma_r^2}]^2 \tag{2.26}$$

So far we assumed that we have a fixed window size ω (m×n). The choice of ω influences the resulting disparity map. If ω is very small, many false matches can occur, especially if the images are noisy. If ω is very big, then the optimum is flattened and the computation time increases. Some approaches use adaptive window sizes to gain their results [KO94]. After calculating disparity values for all pixels, the resulting disparity map should be convolved with a median filter so that single very unrepresentative pixel in a neighbour hood are deleted.

Phase Difference

Another property to match is local frequency components [San88, FJJ91]. If a function $f(x)$ with fourier transform $F(u)$ is shifted by an amount of d then the resulting fourier transform of the shifted function $f(x+d)$ is $e^{jdu}F(u)$. The shift in the spatial domain is equivalent to a phase shift in the frequency domain. It is possible to determine the disparity if the phase differences are found. Since the shift in the spatial domain is not equal for different regions of the image, for example the disparity differs for different objects that are mapped onto the image plane, a local frequency filter is needed to determine the phase differences. The *Gabor filter* [Gab46], which is a bandpass filter with limited bandwidth, can be used for this. Equation 2.27 shows the filter

$$g_{w_0}(x) = gauss(x)e^{jw_0 x} \tag{2.27}$$

$$gauss(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \tag{2.28}$$

and Equation 2.29

$$G_{w_0}(w) = e^{-\frac{(w-w_0)^2}{2\tau^2}} \tag{2.29}$$

shows its fourier transform. The first part of the *Gabor filter* is the *Gaussian function*

σ is the filter width and w_0 is the filter frequency for solving the correspondence problem. The product $\tau\sigma$ is one, which is the theoretical minimum of any linear complex filter [Gab46]. Convolving g_{w_0} with the image intensities I_l, I_r yields a joint spatial and frequency representation of an image [Dau85]:

$$\begin{aligned} c_l(x_0) &= \int I_l(x_0) g_{w_0}(x_0 - x)dx = \rho_l(x_0)e^{j\phi_l(x_0)} \\ c_r(x_0) &= \int I_r(x_0) g_{w_0}(x_0 - x)dx = \rho_r(x_0)e^{j\phi_r(x_0)} \end{aligned} \tag{2.30}$$

As said before, a shift in the spatial domain is represented as a phase shift in the frequency domain, this gives an already estimation of the disparity d.

$$f(x+d) \rightarrow F(u)e^{jdu} \tag{2.31}$$

This theorem states that a spatial shift d corresponds to a frequency shift du. A suitable approximation of the local image shift is the normalized phase difference.

$$d(x) = \frac{\phi_r(x) - \phi_l(x)}{w_0} \tag{2.32}$$

The estimation can be further improved using local frequencies instead of the filter mid-frequency w_0 [FJJ91]. Advantages of the phase difference method are that it is not that sensitive to noise and that the correspondence analysis works with subpixel accuracy.

2.4.2 Feature-based Correspondence Analysis

In contrast to intensity-based methods, feature-based methods first search striking parts inside the image and use the features found for the correspondence analysis. Some advantages of this method are

faster computation because only features have to be compared. Using intensity-based methods for every pixel a depth value has to be computed, thus for every pixel a window has to be extracted and compared with a window which is centered at every pixel along the epipolar line. The costliest part of the feature-based correspondence analysis is normally the extraction of the features. The subsequent part of the comparison concerning needed computational power is mainly influenced by the amount of features found.

lesser ambiguities due to the smaller amount of possible matches, lesser ambiguities during the comparison can occur.

less sensitive comparisons between features are less sensitive to illumination changes than intensity based comparisons because intensity values are not used.

high accuracy the disparity between features can be computed using subpixel accuracy.

In contrast to intensity-based methods, a feature-based approach yields a sparse disparity map, thus depth values are only known for corresponding features found and not for every pixel. Interpolation can be used to get a denser disparity map. Typical features are

Edge points Edges are abrupt significant changes in the signal level and can be computed using convolution masks.

Line segments A line is a set of edge-points which are connected and form a straight line.

Corners Corners are local image features characterised by locations where variations of intensity values in x and y direction are high. In contrast edges are characterised by locations where variations of intensity values in a certain direction are high while variations in the orthogonal direction are low. Different corner detectors exist, most frequently used is the *Harris* [HS88] and the *KLT* (Kanade-Lucas-Tomasi) operator.

Regions The process of grouping pixels into regions of similarity is called segmentation. There are two main approaches used for segmentation, namely

Region Splitting The basic idea of region splitting is to break the image into a set of regions which are coherent within themselves. In the beginning the region is the image itself. If the whole region fulfills some similarity constraint the segmentation is done. If this is not true, the region is splitted into equal parts, usually four. This is done until the regions satisfy the prevailing circumstances. The worst case is that every region just contains one pixel. After the splitting process, it is highly likely that some neighbouring regions have similar properties. Thus, a merging process is used after each split. The algorithm is finished if no further splitting is needed.

Region Growing The main idea of the region growing algorithm is to take a pixel inside the image which is called the seed. If the neighbouring pixels fulfill a similarity constraints, the pixel is added to the region. If no pixels can be added any more, another pixel which is not yet in a region is chosen as the seed and the growing starts again. If every pixel is inside a region the process stops.

The similarity between two features can be defined as the weighted sum of the differences between their properties. For example the similarity of two lines l and r could be defined as

$$S(l,r) = \frac{1}{w_0(l_a - r_a) + w_1(l_b - r_b) + \ldots + w_n(l_z - r_z)} \tag{2.33}$$

where $w_0, \ldots, w_n$ are weights which determine the importance of the properties $a, \ldots, z$ (e.g. attitude, gradient direction, ...) for all features. More important properties are stronger weighted than unimportant ones. Determining the values of the weights that yield the best matches is a non-trivial problem. If a line l achieves the highest similarity

$$S(l,x) = \max_{x} S(l,x) \tag{2.34}$$

and $S(l,x)$ is below a given threshold, the disparity d, in a standard geometry, is defined as the horizontal distance between l and x.

2.5 3D Reconstruction

After the calibration is done and corresponding features in both images are found, the 3D position of the feature can be computed. With an intensity based approach a dense disparity map can be generated. A feature-based approach only provides disparity information for the features that have been extracted. Interpolation can be used to get more 3D points. The mathematical formulation to get the coordinates of an image point in the camera coordinate system[4] is given by

$$\begin{aligned} Z_c &= -f\frac{b}{d} \\ X_c &= \frac{-xz}{f} \\ Y_c &= \frac{-yz}{f} \end{aligned} \tag{2.35}$$

where f is the effective focal length and b is the baseline, both parameters can be computed using a calibration technique. If t_1 is the translation vector of the left camera and t_2 is the calibration vector of the second camera, the baseline b can be computed by subtracting the x coordinates of the translation vectors.

$$b = t_{2x} - t_{1x} \tag{2.36}$$

---

[4] z-coordinate coincides with the optical axis of the camera, origin is at $(0,0,f)$

For many applications it is more convenient to use coordinates of the object points given in another coordinate system. To get a rotation matrix and a translation vector, which are both needed to relate two coordinate system to each other, calibration is used. The extrinsic camera parameters define the orientation and translation of the camera coordinate system in reference to a given world coordinate system, which originates at the calibration pattern. To formulate the relation between the two camera coordinate system, let us introduce a new one, the so-called cyclopean view. This coordinate system has its origin in the middle of the baseline b, so that the z-axis bisects the angle between the left and right optical axes, as illustrated in Figure 2.14.

The angle between the optical axes is 2ϕ. The y-axis of the camera coordinate system has to be parallel. The relation between a point $P = (X, Y, Z)$ in the cyclopean coordinate system, and the same point $P_l(X_l, Y_l, Z_l), P_r(X_r, Y_r, Z_r)$ can be written as

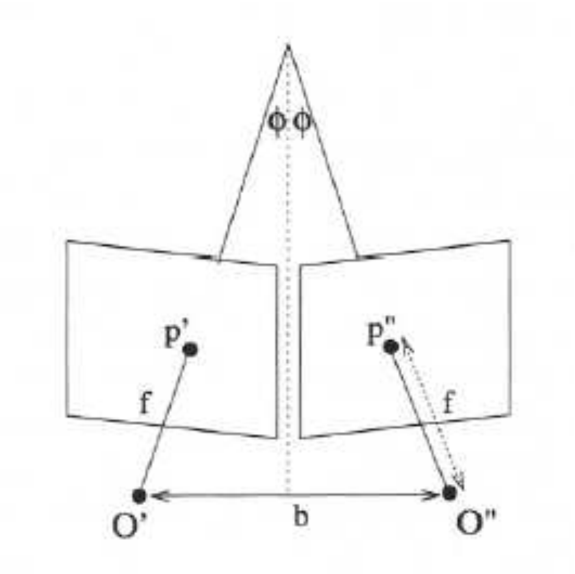


Figure 2.14: Cyclopean view bisects the angle between the left and right optical axes

$$\begin{pmatrix} X_l \\ Y_l \\ Z_l \end{pmatrix} = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \begin{pmatrix} X + \frac{b}{2} - f\sin\phi \\ Y \\ Z \end{pmatrix} \tag{2.37}$$

and

$$\begin{pmatrix} X_l \\ Y_l \\ Z_l \end{pmatrix} = \begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix} \begin{pmatrix} X + \frac{b}{2} + f\sin\phi \\ Y \\ Z \end{pmatrix} \tag{2.38}$$

To get the coordinates of point $P_w(X_w, Y_w, Z_w)$ in world coordinates, after the coordinates of point $P_c(X_C, Y_c, Z_c)$ has been computed, we can use Equation 2.13 to formulate the following equation

$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix}^{-1} \begin{pmatrix} X_c - T_x \\ Y_c - T_y \\ Z_c - T_c \end{pmatrix} \tag{2.39}$$

Transforming the camera coordinate system into another one is useful when the camera is not oriented in an appropriate manner. Additionally, the coordinate system can be shared amongst different applications with different camera systems.

2.6 Summary

This chapter contained the mathematical background needed to understand stereo 3D reconstruction. Every triangulation based reconstruction technique has a similar mathematical background. The introduced camera calibration techniques are essential to get an accurate result, without calibration a stereo algorithm cannot work properly. Without fail necessary are the projection equations that describe how light-rays are mapped onto the sensor chip using two different kinds of camera models, camera calibration, and the meaning of the camera parameters. This knowledge can be used for rectification, which is a transformation of an image into another one which is equivalent to an image that is the result of a parallel camera system in which only horizontal disparities occur. The benefit is a fast correspondence analysis. Finally, the 3D reconstruction from two corresponding image points was explained.

Although stereo techniques can achieve great results, there is a strong relation between quality of the result and computational power needed. The advantage of an intensity-based approach, the dense disparity map, is in concern of calculating time a disadvantage. Even if some real-time application exists [Ali02, HIG02], there is the need of another computational effort to classify the disparities, whereas features often present borders of objects. But also feature-based approaches suffer from time constraints. The more complex the description of the features becomes, the more calculation time is needed to find and describe the features, as well as to search correspondences between them. To conclude, if the goal is a fast algorithm, a feature based approach is appropriate. If a dense disparity map is wanted and time is negligible, an intensity based algorithm should be preferred. For those interested, in [SS02] you can read a taxonomy and evaluation of different intensity based stereo algorithms.

Chapter 3

Methods

So eine Arbeit wird eigentlich nie fertig,
man muß sie für fertig erklären,
wenn man nach Zeit und Umständen das Mögliche getan hat.

[Johann Wolfgang von Goethe, Italienische Reise].

This chapter explains the methods used to reach the final result. Figure 3.1 depicts the basic process cycle of stereo reconstruction and shows where a connection between the process and a logic module (LM) can be established. The chapter is built-on as the process cycle. First of all the acquisition system is presented. Because the inner- and outer camera geometry is a priori not known, the cameras have to be calibrated. Fortunately, this has to be done only once if the camera system remains untouched and the world coordinate system is always the same. For every coordinate system used which is different from the camera coordinate system, the extrinsic parameters have to be computed, in other words, the relation between the camera and the desired world coordinate system has to be described. The intrinsic parameters should be calculated, if an intrinsic property of the camera changes (e.g. zoom will result in a change of the effective focal length). When the images are in memory, they have to be rectified in order to get rid of epipolar geometry. After rectification the epipolar line of a point $p(x, y)$ corresponds to a horizontal line l_y, where y is the y coordinate of the line. After the feature extraction, corresponding elements are searched and for all accordances the 3D position is computed and visualized.

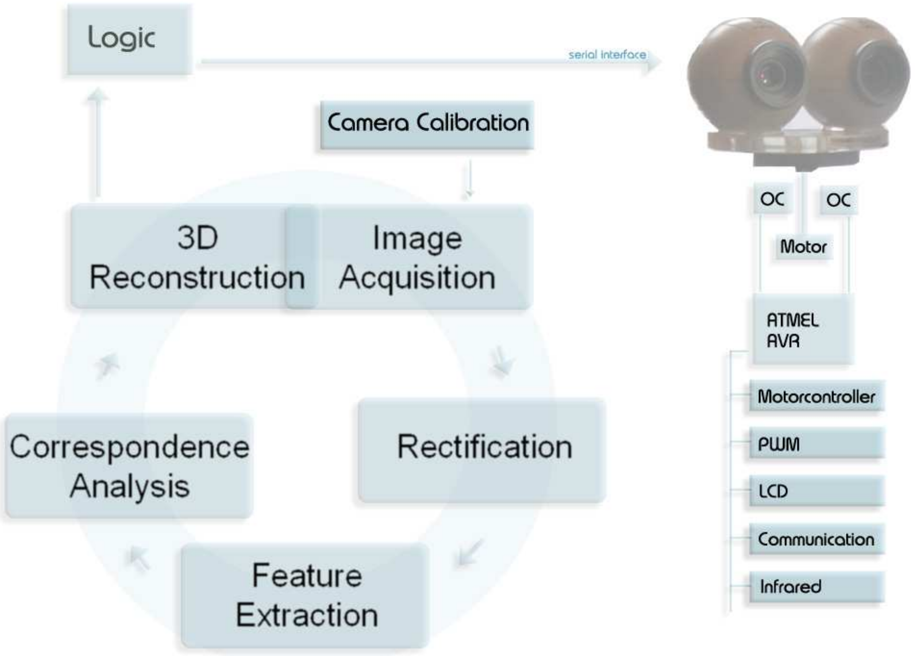


Figure 3.1: Basic process cycle

After the reconstruction, the robot knows the 3D position[1] of objects found. In relation to a dense disparity map, in which for every pixel its 3D information is stored, the 3D representation of straight lines (e.g. the start and end point) or objects is a memory efficient description. During this processing stage, the information should be transferred to the LM of the robot, where it can be used as knowledge base for basic movement decisions (e.g. as a result of obstacle avoidance [MCTM05], ball detection, goal detection,$\cdots$), as well as more complicated ones. For example information retrieval through self-localization [PVK05] or strategy considerations.

The acquisition system used consists of two customary webcams that are connected to a computer. The program that processes the stereo images is written using C and C++. The *open source computer vision library* [OCV] from INTEL provides a code library that is currently used to capture and rectify the images. Also, the Canny edge detector is implemented, as well as some useful data structures.

To analyse the demands and possibilities of a robot, especially in the electrical engineering domain, a connection between an ATMEL microcontroller and a computer,

---

[1] relative to its position

where the image processing is done, has been established. Via the serial (RS232) interface the computer can communicate with the controller and vice versa. A motor controller chip (L293D) is used to trigger a DC motor, which is responsible for the movement of the head. Actually, the head has only one degree of freedom, but the motor controller would be able to trigger two motors, thus the system could easily be extended to a robot-head with two degrees of freedom.

3.1 Image Acquisition

To capture the images, two *Logitech Quickcam Express* cameras are used. They have a resolution of 352 x 288 and use a CMOS sensor chip. Based on the curve seen in Figure 3.2, which shows the relation between disparity and depth in the case of a baseline b with length equal to 5cm, the sensor chips are approximately 5cm translated in x-direction. One can see, that the difference in disparity in a distance range from 25cm to 100cm is almost linear, thus accurate depth values can be computed inside this range. The stereo geometry is almost parallel to avoid occlusions. Other cameras could be used as well, but it must be verified that they can work simultaneously. Problems occurred with two cameras from *Trust inc.* Also

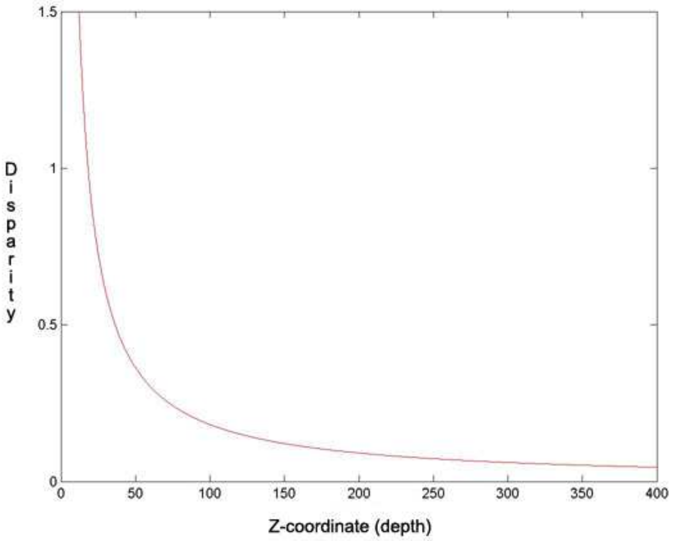


Figure 3.2: Relation between disparity and depth if the baseline $b = 5\text{cm}$ and the effective focal length $f = 3.5\text{mm}$

cameras from different manufacturers can be used, but they should have the same properties (lens, sensor chip). The quality of the device has a strong influence on the results, especially with an intensity based approach, where intensities are directly compared.

To get a stable system, within which the result of the calibration remains usable, even if they are moved, the cameras have been cast in eproxy resin. The focus of the camera can be manually adjusted, but seems not to be very accurate. The fine-tuning was done using the results of calibration to decide whether the focal length of both cameras is approximately the same. The connection between the cameras and the computer is established via USB. Both cameras have a maximum frame rate of 30Hz. Each frame is transferred sequentially, so a maximum frame rate of 15Hz can be reached. The sequential dataflow implicates that the images are not taken at exactly the same time. In a professional stereo system it should be proved, that the capturing of the images is done simultaneously. Otherwise errors in depth values can occur, due to the movement of objects or the camera system in between the acquisition of a stereo pair.

Figure 3.3 shows the field of view of two Omnivision cameras, which will be used

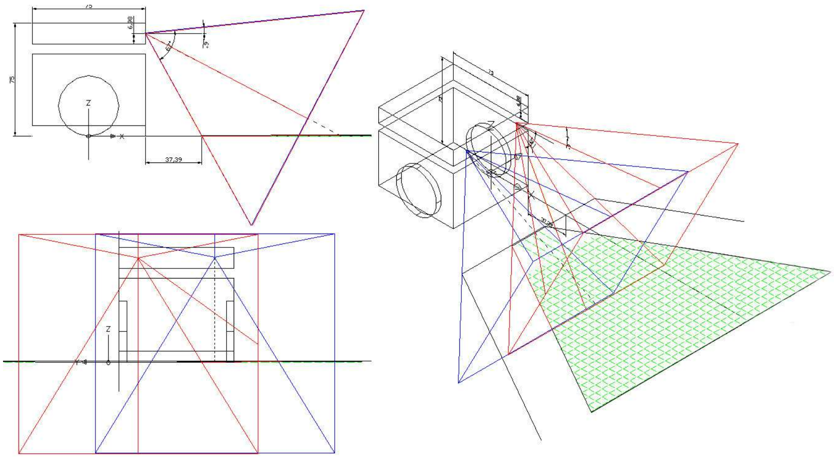


Figure 3.3: Illustration of the field of views of both cameras. The top left image shows the side view, the bottom left shows the front view and the third image shows the robot viewed from a 3D position. The green grid depicts the overlapping area of both cameras projected onto the ground plane.

on the Tinyphoon robot[2]. The baseline has a length of 5cm. They have a viewing angle of 67° and a focal length of 3.5mm. The angle between the horizontal and the optical axis of the camera is given by 27°. This is optimal in the case of a football playing robot, especially in the MiroSot league, because the robots are not allowed to have a height higher than 8cm. Also the football field is not much higher, so everything that should be seen by the robot is within its field of view. The robot will be able to see things which are more than 37.9mm[3] away from the camera.

3.2 Calibration

To gain information about the internal and external camera geometry, as well as the geometric relation between both cameras, the algorithm proposed by Zhengyou Zhang, which is described in [Zha00], was used. In order to calibrate the cameras, the calibration routine needs at least two observations of a planar pattern with different orientations. The OpenCV implementation uses a chessboard-pattern, as shown in Figure 3.4.

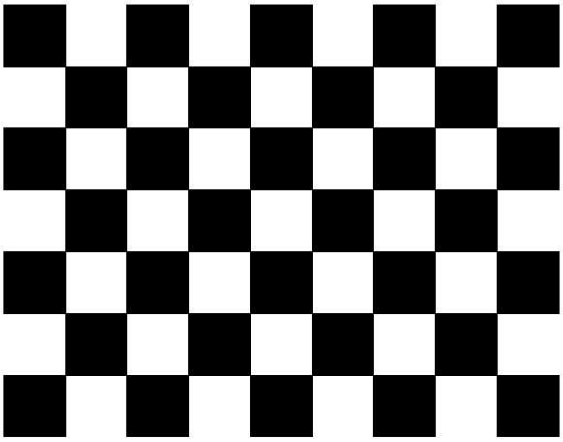


Figure 3.4: Calibration pattern used by the OpenCV calibration method

The benefit of this calibration technique is its simplicity, flexibility, robustness, low cost and of course it is already implemented. To calibrate a stereo system using

---

[2] one of the smallest autonomous football playing robot, developed at the ICT-Institute of Vienna. See Figure 3.11 for an image of the robot and further explanation

[3] measured in horizontal direction, difference between the robot and the first occurring object that is on the ground plane.

this technique, it is only needed to define the number of internal corners, the size of the squares in mm and the number of views to be taken into account. For example, the calibration pattern shown in Figure 3.4 contains 8 x 6 internal corners and when printed on a A4 paper the squares are 18mm x 18mm. For the calculation, ten different views of the calibration pattern, which has to be visible in both images, e.g. in the left and right view, are used. The first step of the implementation is to locate internal chessboard corners. For example, a simple chessboard has 8 x 8 squares and 7 x 7 internal corners, that is, points, where the squares are tangent. Initially the corners are approximated and if a corner is found, it is highlighted inside the image. The two images at the bottom of Figure 3.5 shows the result of this process. If all corners of the chessboard pattern have been found, the highlight changes it is color and connects the single corners. The two images at the top of Figure 3.5 depicts this situation. In this stage the algorithm tries to find the corners using subpixel accuracy [Res01]. If the pattern is found in both images, the coordinates of the corner candidates are stored for the succeeding calculation. If enough patterns have been found, the algorithm computes the intrinsic and extrinsic camera parameters. First it starts with an analytical solution which is followed by

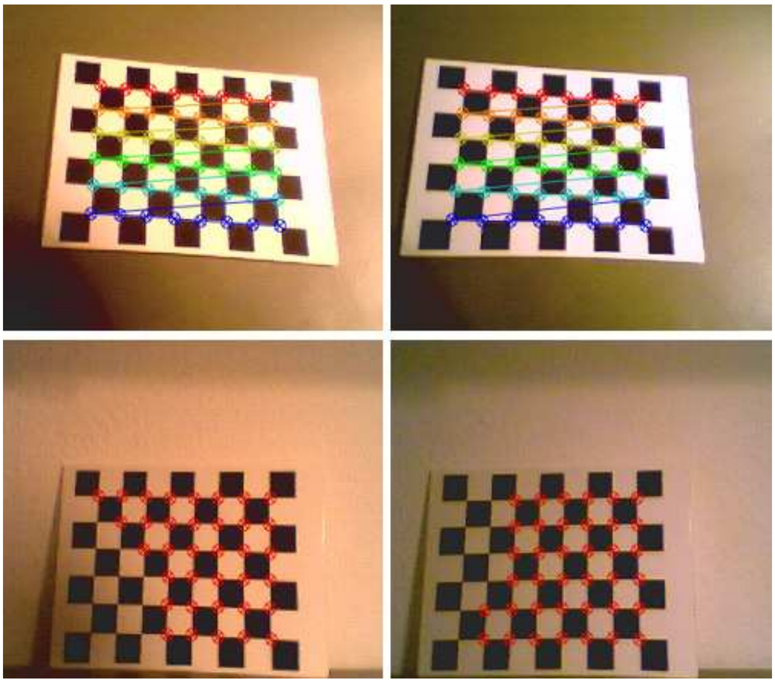


Figure 3.5: Highlighting of the images during calibration

a non-linear optimization technique based on the maximum likelihood criterion[4]. The resulting calibration file is shown in Table 3.1, where the first two diagonal elements of the camera matrix are the *effective focal lengths* in x and y direction given in pixels. The third and sixth value are the *camera center coordinates*, also in pixels. The *translation vector* is given in mm and describes the translation between the camera and the world coordinate system. The *rotation matrix* specifies the rotation between the camera and the world coordinate system and is defined as the composition of rotations (roll ψ, pitch ϕ, yaw θ) around the X, Y and Z axis.

Results of calibration for the left camera				
camera matrix	493.774	0	196.450	
	0.000	496.007	128.258	
	0.000	0.000	1.000	
rotation matrix	0.005	-1.000	0.011	
	1.000	0,005	-0.028	
	0.028	0.011	1.000	
translation vector	88.17	-79.706	611.79	
distortion	0.738	-1.494	-0.03	0.024

Results of calibration for the right camera				
camera matrix	497.525	0.000	172.205	
	0.000	500.180	128.569	
	0.000	0.000	1.000	
rotation matrix	0.034	-0.999	-0.009	
	0.997	0.034	-0.063	
	0.063	-0.007	0.998	
translation vector	141.617	-91.735	617.539	
distortion	0.668	-3.175	-0.004	0.022

Table 3.1: Results of calibration for both cameras

$$R = R_X \cdot R_Y \cdot R_Z \tag{3.1}$$

Matrix R_X describes the rotation around the X axis and is defined as

$$R_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{pmatrix} \tag{3.2}$$

---

[4] see [Zha00] for a close description

Matrix R_Y describes the rotation around the Y axis and is defined as

$$R_Y = \begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix} \tag{3.3}$$

Matrix R_Z describes the rotation around the Z axis and is defined as

$$R_Z = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.4}$$

Equation 3.1 combines the three matrices R_X,R_Y and R_Z, thus R can be written as

$$R = \begin{pmatrix} \cos\phi\cos\theta & -\cos\psi\sin\theta + \sin\psi\sin\phi\cos\theta & \sin\psi\sin\theta + \cos\psi\sin\phi\cos\theta \\ \cos\phi\sin\theta & \cos\psi\cos\theta + \sin\psi\sin\phi\sin\theta & -\sin\psi\cos\theta + \cos\psi\sin\phi\sin\theta \\ -\sin\phi & \sin\psi\cos\phi & \cos\psi\cos\phi \end{pmatrix} \tag{3.5}$$

As one can see, after calibration, the relation between the camera and world coordinate system is completely described. Many different methods for camera calibration have been proposed so far. [CF97] is a paper on the historical development of camera calibration techniques, which also covers an exhaustive literature survey.

After the images are transferred to memory, the result of the calibration is used to rectify the images. The benefit is that corresponding elements can be found on the horizontal scan line. Before the process of rectification, also a point to line correspondence could be established, but the epipolar line has to be taken into consideration. After rectification there is no further need to think about the epipolar geometry.

3.3 Edge detection

The next step is to transform the images into a representation in which only edge segments remain. Edges are declared as significant changes in the signal level, thus can be searched with the help of convolution masks which approximate the first or second derivation of the image. If the mask measures the first derivation of the image in x and y direction, denoted as $G_x(x, y)$ and $G_y(x, y)$, an edge point is found if the magnitude $m = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$ of the convolution at the measured point is above a threshold τ. Also the direction $\varrho = \arctan(\frac{G_y(x,y)}{G_x(x,y)})$ of the edge can be computed. A zero-crossing represents an edge in a mask which calculates the second derivation. The direction of the edge is stored in the values on the right and left side of the zero-crossing. Some popular convolution masks are:

Sobel The Sobel operator approximates the first derivation of the image. The operator mask of the Sobel operator is defined as

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{3.6}$$

Prewitt The Prewitt is similar to the Sobel operator, its operation mask is defined as

$$P_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \qquad P_y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{3.7}$$

Zero Crossing The core of the zero crossing operator is the LoG Filter (Laplacian of Gaussian). It is defined as

$$Log(x, y) = \frac{-1}{\pi\sigma^4}[1 - \frac{x^2 + y^2}{2\sigma^2}]e^{-\frac{x^2 + y^2}{2\sigma^2}} \tag{3.8}$$

and is shown in Figure 3.6 The LoG operator calculates the second derivation of an image, as a result in an area with no change in intensity values the response of the LoG operator will be zero. However, if there is a change in intensity the response will be positive on the darker side, zero in between (on the edge itself) and negative on the lighter side. Thus the direction of the edge is encoded in the resulting grayscale map after convolving with the LoG operator. The LoG operator is *isotropic*, so the quality does not depend on the attitude of the edge in relation to the image.

Canny The Canny edge detector can be roughly divided into six steps:

Step 1 During the first stage of the Canny algorithm, noise has to be filtered out. This is done by convolving the image with a Gaussian filter mask.

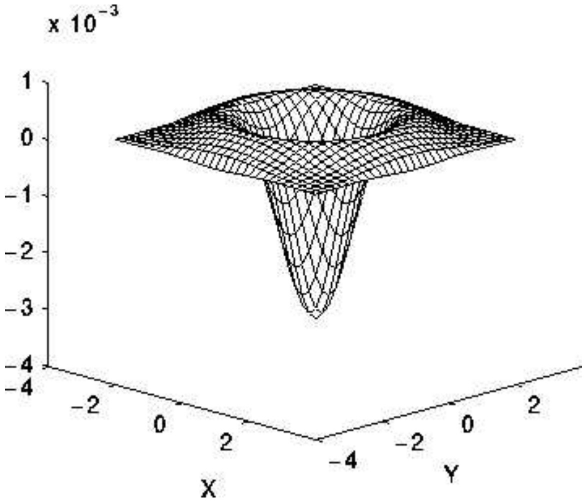


Figure 3.6: LoG Filter

The size of the mask is proportional to the sensitivity and indirectly proportional to the localization achieved by the detector.

Step 2 The strength of the edges is found by convolving the images with the Sobel operator (defined in Equation 3.6.), which performs a 2-D spatial gradient measurement on the image. The magnitude of the gradient can be found using the equation

$$M = \sqrt{G_x^2 + G_y^2} \tag{3.9}$$

A faster computation of the magnitude is given by

$$|M| = |G_x| + |G_y| \tag{3.10}$$

where G_x and G_y are the first derivations of the image in x and y direction. G_x is proportional to variations in the horizontal direction and G_y to variations in the vertical direction. If both derivates are combined, as it is in Equation 3.10 all directions are taken into consideration.

Step3 The direction of the edge can be computed as

$$\theta = \begin{cases} \arctan(\frac{G_y}{G_x}) & \text{if } G_x \neq 0 \\ 0^\circ & \text{if } G_y = 0 \ \& \ G_x = 0 \\ 90^\circ & \text{if } G_y \neq 0 \ \& \ G_x = 0 \end{cases} \tag{3.11}$$

Step 4 The direction has to be classified into five sectors. This process is similar to dividing a semicircle into five regions. Lets say the result of Equation 3.11 is θ and the new direction will be θ' , then θ' can be computed as:

$$\theta' = \begin{cases} 0^\circ & \text{if } \theta \geq 0 \text{ \& } \theta \leq 22.5 \text{ \& } \theta \geq 157.5 \text{ \& } \theta \leq 180 \\ 45^\circ & \text{if } \theta > 22.5 \text{ \& } \theta \leq 67.5 \\ 90^\circ & \text{if } \theta > 67.5 \text{ \& } \theta \leq 112.5 \\ 135^\circ & \text{if } \theta > 112.5 \text{ \& } \theta < 157.5 \end{cases} \tag{3.12}$$

According to [HB93], it could be better to calculate the discrete tangent directly, without the indirection of first computing the arctan and then classifying into five regions.

Step 5 After the edge directions are known, nonmaximum suppression is applied which traces along the edges and deletes every edge point (set it to 0) which is not considered to be an edge.

Step 6 Finally, hysteresis is used to eliminate the breaking up of an edge due to variations of the operator. Therefore two thresholds are defined. τ_H which is the higher threshold and τ_L which is the lower threshold. If the magnitude of a point is higher or equal to τ_H, it is immediately marked as an edge. Every pixel that is connected to such a pixel and has magnitude higher than τ_L is also marked as an edge. If you think of following an edge, the search starts at a pixel with a magnitude higher than τ_H and does not stop until a pixel with a magnitude lower than τ_L is reached.

The output of the Canny edge operator is an edge image with a low error rate, good localization and only one response to a single edge. Figure 3.7 shows the results of the Canny edge detector with two different thresholds. With a low threshold many artifacts remain whereas with a high threshold important information could be lost. The choice which thresholds to use depends on the demands of the continuative process as well as the environment conditions (natural light, artificial light, contrast, $\cdots$).

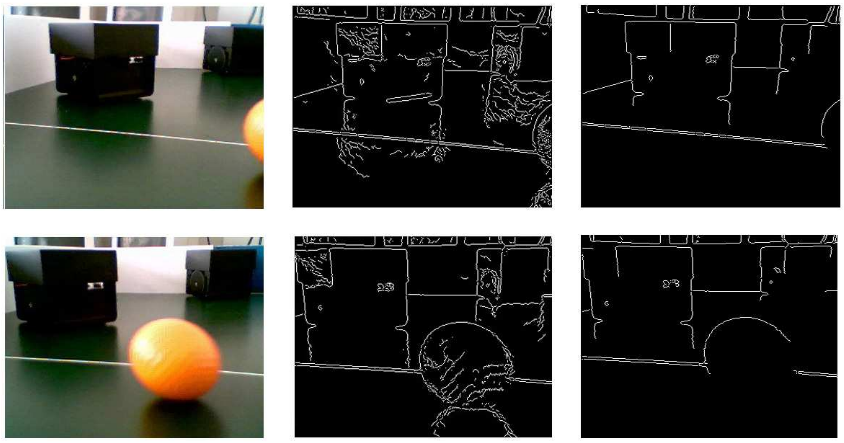


Figure 3.7: Result of the Canny edge detector, images are taken from a stereo camera system. The images are not rectified. The first column shows the original images, the pictures in the middle are produced by the Canny operator with $\tau_L = 3.2$ and $\tau_H = 8.0$, the thresholds of the operator producing the pictures on the right side are $\tau_L = 25.6$ and $\tau_H = 51.2$

The decision which detector to use depends on the problem which is to solve. Every edge detector has its special advantages and disadvantages. In order to find straight lines, the Canny edge detector was appropriate.

3.4 Line Extraction

A line is defined as a set of points which are connected and form a straight line. It can be completely described through the start point and end point. There are different techniques for finding straight lines. The most popular one is the *Hough transform*. The basic idea of this technique is to find curves that can be parameterized, like lines, polynomials, circles, etc. in a suitable parameter space. The detection of straight lines in images will be shortly discussed because it is the main feature used in the proposed approach. Let us assume that the line is described in parameterized form $r = x\cos\theta + y\sin\theta$, where r is the perpendicular distance from the origin and θ the angle of the x-coordinate with the normal. For any point $p = (x_i, y_i)$ on a straight line, r and θ are constant. Points in the cartesian image space maps to curves in the polar Hough parameter space. This point-to-curve transformation is called Hough transform. Points which are collinear in the image space intersects at a common point (r, θ). Figure 3.8 shows how three lines are mapped into the parameter space. The transformation is implemented by quantizing the Hough parameter space into finite intervals, called accumulator cells. For every $(x_i, y_i,$ the parameters (r, θ) are computed and the accumulator at position (r, θ) is incremented. Peaks in the parameter space represent a straight line in the image space. The main disadvantage is that it needs a lot of computational effort to transform the image into the hough-space and the output is a parametric description of a line, thus the starting and end point are not known and it needs some comparisons to find the starting and end point. Another way is to define a convolution mask which detects lines. The main disadvantage with this method is that every mask only finds lines that have a certain direction.

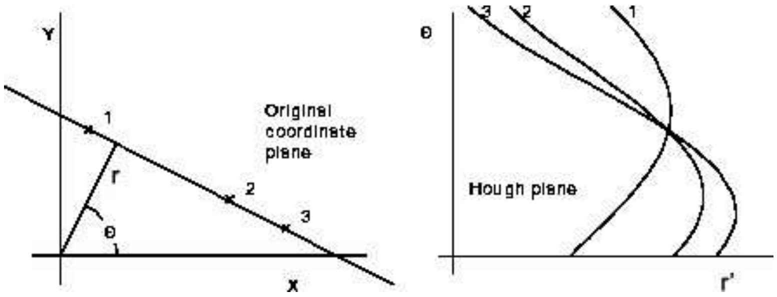


Figure 3.8: Hough transform, the left image shows three lines in the image space and the right image shows the corresponding lines in the parameter space

Another approach is to follow edge points iteratively and use the attitude of the found edge points as a reference to decide whether the following set of edge points seems to be in a line with the same attitude or not. The proposed work uses such a method.

A linear list is used to store the lines. A list element (line) consists of the following attributes

Point start Stores the image coordinates of the point, where the line has its origin.

Point end Stores the image coordinates of the point, where the line ends.

Point cp The compare-point stores the image coordinates of the point, that is used to calculate the disparity between this line and the corresponding line in the other image. In other words, if l is the line in the left image and l' its corresponding line in the right image, then the cp for l is defined as

$$\begin{aligned} l \rightarrow cp_x &= \frac{l \rightarrow start_x + (l \rightarrow end_x - l \rightarrow start_x)}{2} \\ l \rightarrow cp_y &= \frac{l \rightarrow start_y + (l \rightarrow end_y - l \rightarrow start_y)}{2}. \end{aligned} \tag{3.13}$$

The cp for l' is defined as the point of intersection between a horizontal line h, that has its origin in $l- > cp$, and l'.

Real direction Stores the direction of the line, which is computed by adding up the directions of the edge pixels that are considered to lie on the line. The resulting direction is mainly used to find corresponding lines in other images, as well as finding corresponding object lines.

Digit length Stores the length of the line. It is increased for every pixel that is added to the line.

Digit counterX Counter of steps in x direction. It is increased or decreased in every iteration, dependently on the chosen direction.

Digit counterY Counter of steps in y direction. It is increased or decreased in every iteration, dependently on the chosen direction.

Digit disparity Disparity, ∞ if no disparity is found, or the disparity in pixels if a corresponding line has been found.

Real hitRateDisp Stores the rating of the correspondence between this line and a line in the other image. The line with the best rating is chosen, otherwise the rating is set to a value which is equivalent to ∞.

Real hitRateObj is the quality of the found object line or ∞ if no line is found.

Real xw Stores the x-coordinate of the line in the world coordinate system.

Real yw Stores the y-coordinate of the line in the world coordinate system.

Real zw Stores the z-coordinate of the line in the world coordinate system.

LineList* next Pointer to the next element of the line list or NULL at the end of the list.

LineList* corrEdge Pointer to the corresponding line in the other image or NULL if no line is found.

EdgeList* corrObj Pointer to a line that seems to belong to the other side of this line i.e. the two lines of a goal or a robot.

First of all everything is set to 0, except the disparity and both hit rates, which are all set to ∞. A LIFO (last in, first out) list ϑ of length ς is created, which stores the chosen directions. The algorithm controls every pixel in the image whether it is an edge pixel. The search starts at the top-left of the image and searches from left to right and top to bottom. For vertical lines it is proven that when an edge pixel is found and it belongs to a line, the pixel is the first point of the line. This is not true for lines that are almost horizontal, but it can be ignored because horizontal lines cannot be used to calculate disparity and thus they are not searched for.

If an edge pixel is found and it is not already processed, the start and end points of the line list are set to its coordinate. Then the algorithm tries to follow the edge pixels, starting from the actual point, until the length of the line has reached a given threshold ς. During the process of following a line, pixels that seem to be more likely to lie on a straight line are preferred. This is necessary, otherwise the algorithm would always prefer a given direction.

If a connection between a line point and one of its neighbour points, that can be used to continue the search, exists, the direction in which the new point can be found is stored. See Table 3.2 for possible directions and their encodings. If no connection exists the algorithm stops and the pixels that are already in ϑ will be marked as processed. Then it continues searching edge pixels that are not processed yet. But if the length of the line reaches the threshold ς, the attitude α of the first part of the line is computed. ϑ is now totally filled, that means if we add another direction, the oldest element will be shifted out of the list. Let us assume that c_x is the counter in x direction, c_y is the counter in y direction and β is a boolean variable, then α can

left, up 7	up 8	right, up 9
left 4	Center	right 6
left, down 1		right, down 3

Table 3.2: Encoding of directions

be computed as

$$\alpha = \begin{cases} \frac{c_x}{c_y} & \beta = false \quad \text{if } |c_y| > 2 \\ \frac{c_x}{c_y} & \beta = true \quad \text{if } |c_y| > 0 \ \& \ |c_y| \leq 2 \\ 10 & \beta = true \quad \text{otherwise, ie. } |c_y| = 0,\ \beta = \text{true} \end{cases} \tag{3.14}$$

α is used as a reference to decide whether the following pixels belong to a straight line or not. We now have a line l_{ref} with length ς and attitude α. A second *EdgeList* l_h is generated, whose starting point is equal to the end point of l_{ref}. The algorithm does the same as before, it tries to follow the edge. If this is possible, l_h stores the new end coordinate, its counters are increased and the new point is marked as processed. The chosen direction, in which the new edge pixel is situated is shifted into the ϑ list, thus the oldest element is lost. Until now, it is not clear if the new added point belongs to the line l_{ref}, thus the points added are stored in a separate list. This is needed, because otherwise a lot of points would be lost during the search process[5]. With the aid of ϑ we calculate a new attitude α'. The computation is the same as shown in Equation 3.14. One could ask why not compare the attitude of the whole line with the reference line. If the whole line would be compared, the computation of α' would not be that significant. By computing the new attitude with ϑ it is proved that only the last part of the line is incorporated. If the absolute difference of both attitudes $|\alpha - \alpha'|$ is below a lower threshold τ_l, the points that are not on the list are added to the list and marked as processed. The information needed is stored in l_h. The list which stores the points is emptied, because now it is clear that these points belong to the line. Otherwise, if the difference is below a

---

[5] in order to circumvent that points which do not belong to the line are marked as processed, we have to store them and after the search is finished mark them as not processed.

high threshold τ_h the search continuous.

τ_h is used to handle the given noise. When τ_l is very low, the distance between a point and the line is only allowed to be very small thus noise can lead to the end of the search. With the use of τ_h a higher variance is allowed, but only to follow the edge. To gain access to the line, in sum the change in direction has to be low. Only pixels that belong to the line are marked as processed. If the difference is higher as τ_h the search is finished. The line is kept, if its length is higher than the minimum length accepted. If the line is accepted, a new *LineList* is generated and the old one points to the new one.

A linear list of lines is the result of this procedure. Figure 3.9 shows an image in which straight lines are searched. Unfortunately, edge pixels are not always connected. Thus, after the line list has been generated it tries to connect lines if they have approximately the same attitude, e.g. the absolute difference of two lines is below a threshold, and the end point of one line is near the start point of the other or vice versa.

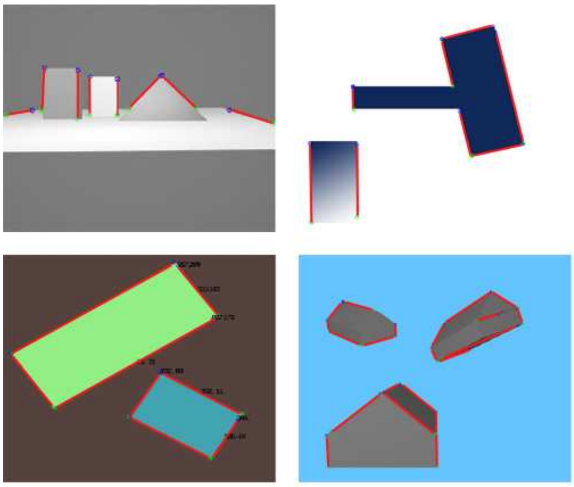


Figure 3.9: Results of the line extraction, horizontal lines are dismissed. Found lines are plotted in red, start points are green and the end points are blue (colors can only be seen in the electronic version of this thesis).

3.5 Correspondence Analysis

According to a recent taxonomy [SS02], methods creating a dense disparity map can be roughly divided into two groups.

Global algorithms [KZ02], which tries to assign disparities in order to minimize a global cost function. They yield very accurate and dense disparity maps but at the expense of highly computational efforts, thus are not applicable to real-time environments.

Local algorithms [HIG02, DSMMN02, KKK⁺95], also referred to as area-based algorithms, compare the photometric properties of neighbouring pixels in order to determine disparity. They yield significantly less accurate results compared to global algorithms, but may run in real-time. This depends heavily on the maximum disparity allowed.

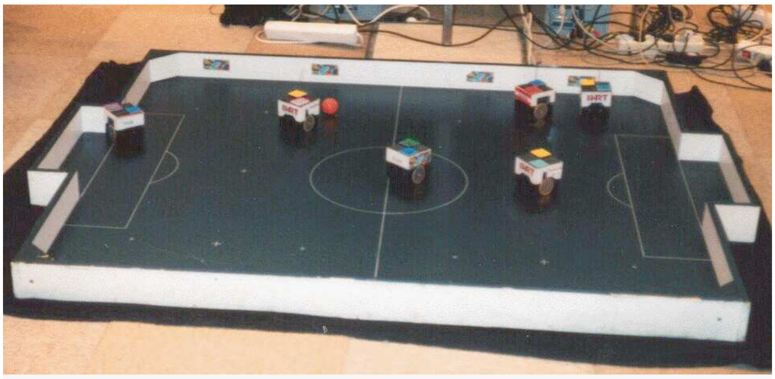


Figure 3.10: The football field of the FIRA MiroSOT football league

The fastest method to compute disparity is a *feature-based* approach with the constraint that the feature extraction is fast - which is the first problem that has to be solved. To find a feature which is easy to extract and easy to compare. Features are striking parts of an image, thus have a strong relation to the environment and the objects that are recorded. In the case of a football playing robot, the football field and the robots. The size of the robots in the MiroSOT football league is limited to 8cm x 8xm x 8cm. Figure 3.10 shows the official football field of the MiroSOT football league. Viewed from the camera of the Tinyphoon robot (Figure 3.11), the goal, the markers on the ground and the field are , when projected to the image plane, represented by lines. Thus, lines are chosen as features for the corresponding analysis. The benefit of a straight line is that it has a memory efficient description and that a fast line extraction can be implemented. In Section 3.4 an iterative line detector has been presented. Besides the Canny edge detector it has a time complexity (TC) of $O(n)$, where n is the number of image points. To find corresponding lines, every line has to be compared with each other, thus the TC of the correspondence analysis is $O(n^2)$.

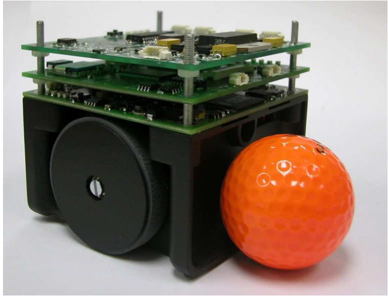


Figure 3.11: The Tinyphoon robot

3.5.1 Search for corresponding lines

After the line lists for both images are generated, the *correspondence problem* needs to be solved. In contrast to an intensity-based correspondence analysis, there is no need to have a maximum disparity limit because all the lines are compared and thus all possible disparities are taken into consideration. But a constraint, that a line which is nearer than another is more likely to be chosen than a line which is more far away, has to be inserted. This constraint is needed, because if there are two objects mapped with approximately the same y-coordinate and approximately the same line properties, e.g. the same length, attitude and midpoint, as it is if two robots are next to each other, the left border of the robot could be matched with the left border of the other robot. Figure 3.12 depicts this situation. To avoid matching of two opposite lines, the gradient direction is taken into consideration. Figure 3.12 shows the gradient direction for the left edge image (depicted as red arrows in the electronic version of this thesis at the top and bottom). To measure the similarity between two lines, the weighted sum of the difference between the attitude α , the direction θ, the length g and the difference in the y-coordinate between the start

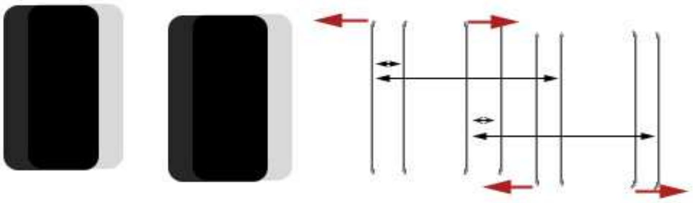


Figure 3.12: Line ambiguity, the left image shows the merged original images (the alpha value of the right image has been decreased) and the right image shows the line ambiguity if no constraint is inserted

and end point p_s, p_e of the line is computed. Let $'$ denote a line inside the right image, then the rating r of a line l can be formulated as

$$r(l, l') = w_{pos}|p_s - p'_s| + w_{pos}|p_e - p'_e| + w_\alpha(\alpha - \alpha') + w_g(g - g') + w_\theta(\theta - \theta') \quad (3.15)$$

where w_{pos}, w_α, w_g and w_θ are weights, that define the importance of the line property. The currently implemented method uses $(0.5, 200, 0.1, 0)$ as $\vec{w}$. A line l corresponds to a line l', if

- the rating $r(l, l')$ is smaller than the best rating achieved, thus $(\forall x | r(l, l') \leq r(l, x))$ must hold.
- the x-coordinate of the compare point in the left image is greater than the x-coordinate of the compare point in the right image
- the y-coordinate of the intersection between the horizontal line which has its origin in the left compare point and the line in the right image lies inside this line.
- the rating r is below a given threshold
- the calculated disparity between the two lines is smaller than the maximum disparity allowed
- the difference of the gradient direction $\theta - \theta'$ is below a given threshold

During the search for corresponding lines, every line inside the left image is compared with every line inside the right image. The lines with the best rating are considered as corresponding lines. If two lines fulfill the constraints mentioned above, the lines are removed from the list of possible candidates. To avoid false matches, the rating has to be smaller than a given threshold. If two corresponding lines with disparity d have been found, a horizontal line of length d is inserted in the left image. It has its origin in the midpoint of the left line and if the right line would be inserted into the left image, the compare point would be the end point of the line.

3.5.2 Object detection

If corresponding edges have been found, the 3D position of these lines can be computed. But so far it is not clear which lines correspond to each other, thus which lines belong to objects. According to [SB97], objects can be detected searching for lines with approximately the same properties and opposite directions. Also intensity values could be used to detect objects. If two possible candidates of edges belong to an object, the intensity values to the right of the left edge are often similar to the intensities on the left of the opposite edge. Due to the given time constraints, this technique is not used, but for other applications it could be a good attempt. Another given fact is that the disparities of the borders of objects are often very similar. The dissimilarity between the disparities is proportional to the spatial difference in x-direction thus the allowed threshold between the difference in disparity can be related to the difference in x-direction. In a mathematical manner, the object detection technique can be defined as follows:

$$r(l, l') = w_{disp} \frac{(d - d')}{(cp.x - cp'.x)} + w_{ydif}(cp.y - cp'.y) + \frac{w_{xdif}(cp.x - cp'.x)}{l} \tag{3.16}$$

where d and d' are the disparities of the left and right edge, respectively. cp and cp' are the compare points, $.x$ and $.y$ denotes that the x alternatively y coordinate is used. l is the length of the left line. The weights are set to

- $w_{disp} = 10$
- $w_{ydif} = 2$
- $w_{xdif} = 50$

It is not trivial to decide which weights yield the best result. So far the values have been empirically chosen, but sometimes false matches occur. A good attempt could be to let a neural network solve this problem. In the end it is just a linear classification problem in which the weights have to be chosen so that if a line l corresponds to a line l', the rating $r(l, l')$ should be maximal. In a mathematical formulation this would look like: $l \leftrightarrow l' \rightarrow r(l, l')$ is maximal $\forall$ lines.

3.6 3D Reconstruction and Visualization

For each pair of corresponding lines, the 3D position can be computed. OpenGL has been used to visualize the result of the reconstruction. The origin of the left camera is positioned at the origin of the OpenGL coordinate system, but can be moved along the y axis if desired. A 10cm x 10cm grid can be displayed to have an idea where the objects are placed in reference to the real world. Also the view of the camera can be displayed. Figure 3.13 shows the result of the reconstruction using a synthetic image pair. The coordinate system is off.

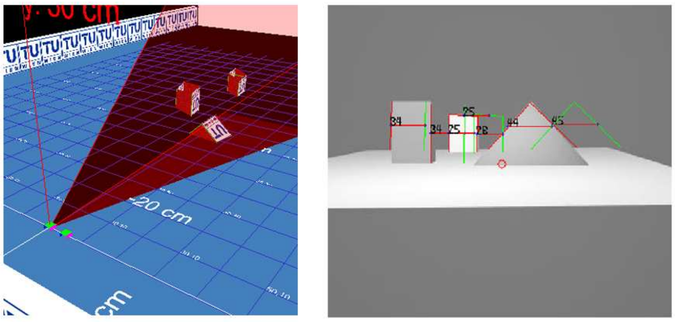


Figure 3.13: Results of the reconstruction using synthetic data.

If the right mouse is clicked inside the application, the program switches to the 3D window. It is now possible to fly around in the reconstructed world. The commands are given by

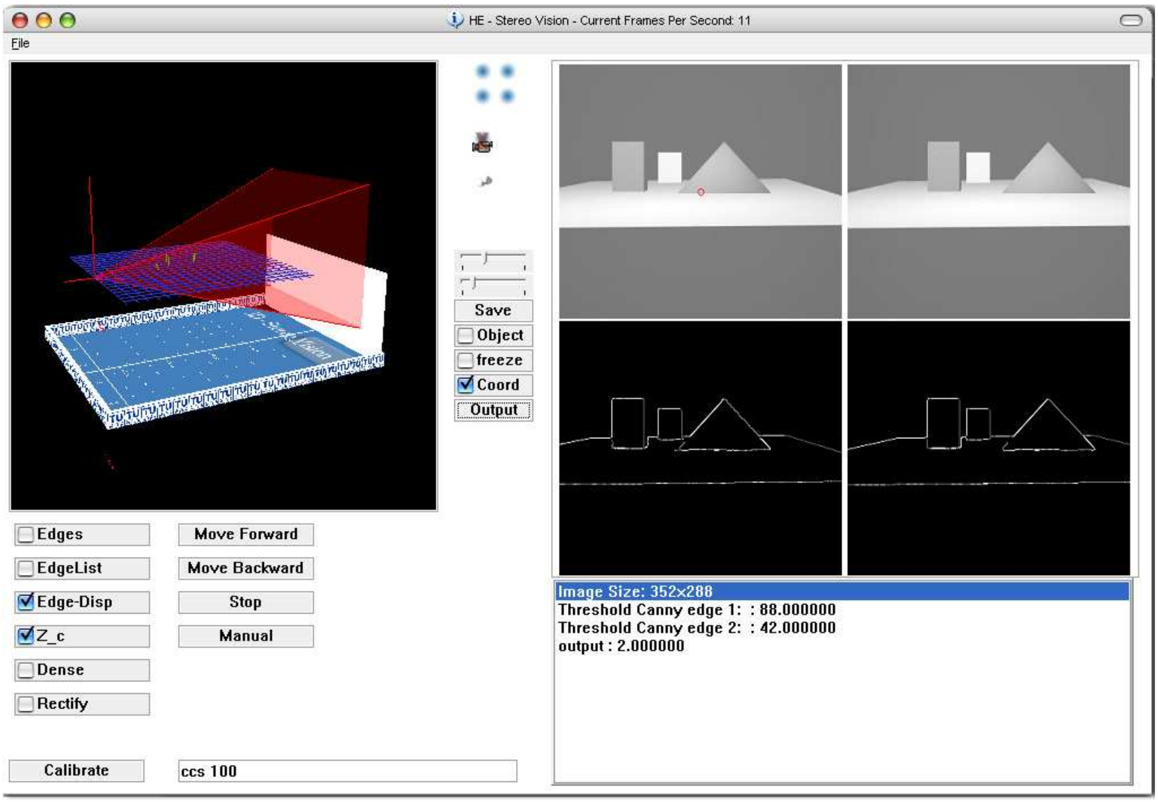
w move forward

a strafe left

d strafe right

s move backward

The mouse can be used to change the viewing direction. Figure 3.14 shows an image of the whole application. The view of the camera is displayed, as well as the coordinate system that is 50cm translated in y direction. The input field, next to the calibrate button can be used to load image pairs into the program by entering their filename. If a image pair is found, it is loaded automatically. The prompt "*ccs* digit" moves the camera coordinate system, which has its origin in the OpenGL coordinate system, along the y axis. The value of digit represents the amount of the movement in cm.



HE - Stereo Vision - Current Frames Per Second: 11

ccs 100

```
Image Size: 352x288
Threshold Canny edge 1: : 88.000000
Threshold Canny edge 2: : 42.000000
output : 2.000000
```

Figure 3.14: Win32 application

The buttons on the left are used to control the program, e.g. show the edge list, rectify the images or calibrate the camera. The buttons next to them can be used to control the robot head. If *Manual* is pressed, the opto-couplers are deactivated, thus the head could turn in. The four circles in the middle, represents the images on the right side. If one is clicked, the image is displayed using the whole image area. The camera button can be used to switch between file or camera input. *Freeze* stops the calculation, and *output* can be used to change the output content.

3.7 Robot Head

Because the system was built with regard to use it on a mobile robot with a rotating head, a prototype, which can be seen in Figure 3.15, was developed. The electronics and mechanics are inside a wooden box, on which the cameras are mounted. The cameras can be rotated at a maximum of 270°. Two optoelectronic couplers are responsible that the head does not turn in while receiving commands from the server. The MC does all the calculations needed, except the image processing, which is done on the controlling computer. It is also responsible to guide the head. According to Figure 3.1 at the beginning of this chapter, the LM would be concerned with this assignment (e.g. move forward and rotate the head clockwise in order to find the ball, that just moved out of the field of view). The controller as a standalone machine is not very intelligent because it is not able to process the images yet. Also the connection between the head and the server is wired, which is a big problem when it comes to a mobile robot which is capable of moving around. This will be the major concern in my future work. The wire diagram of the robot-head is shown

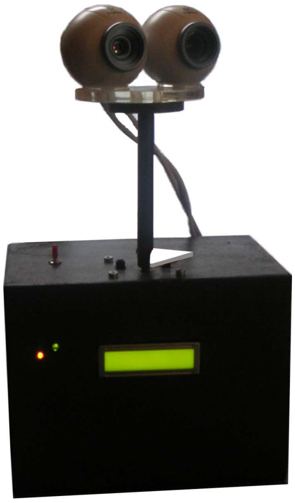


Figure 3.15: Image from the rotating Head, the figure shows a front view of the construction. The orange led indicates that the head is turned on and the green led changes its state (on/off) every time a command arrives from the computer. The LCD can be used to display information.

in Figure 3.16. The main parts of the diagram are

The voltage regulator is responsible to transform the input voltage to constant $5V$. The two capacitors suppress possible oscillations. The rotating head can be used within the range from $6V$ to $12V$. If a higher voltage is used, the system still works, but the regulator will become very hot and thus a lot of energy is lost.

The tuned circuit generates a $4MHz$ frequency, which is used as an external clock generator. Although there is an internal clock generator, the external is used because the other only oscillates with $1MHz$.The main part of this circuit is a $4MHz$ crystal.

The RS232 interface is responsible for the communication between the robot and the computer and vice versa. The Max233 chip does the conversion from $[0,5]V$ to $[-9,+9]V$. Three pins of the serial interface of the computer are used: RxD,Txd and GND.

The motor controller is able to trigger two DC-motors. So far only one is connected, but it would be able to trigger two motors. The input voltage of the whole circuit[6] is used as current supply of the motor. So far software PWM (Pulse Width Modulation) is used to trigger the motors, but it is planned to use hardware PWM.

The parallel interface is where the controller can be programmed. I use the avr-gcc compiler to generate programs that can be run on the microcontroller.

The LCD display is used to communicate with the user and connected to pins 0 to 5 at port E. It uses 4-Bit mode to transfer the data.

---

[6] not the constant 5V that are used for the rest of the circuit.

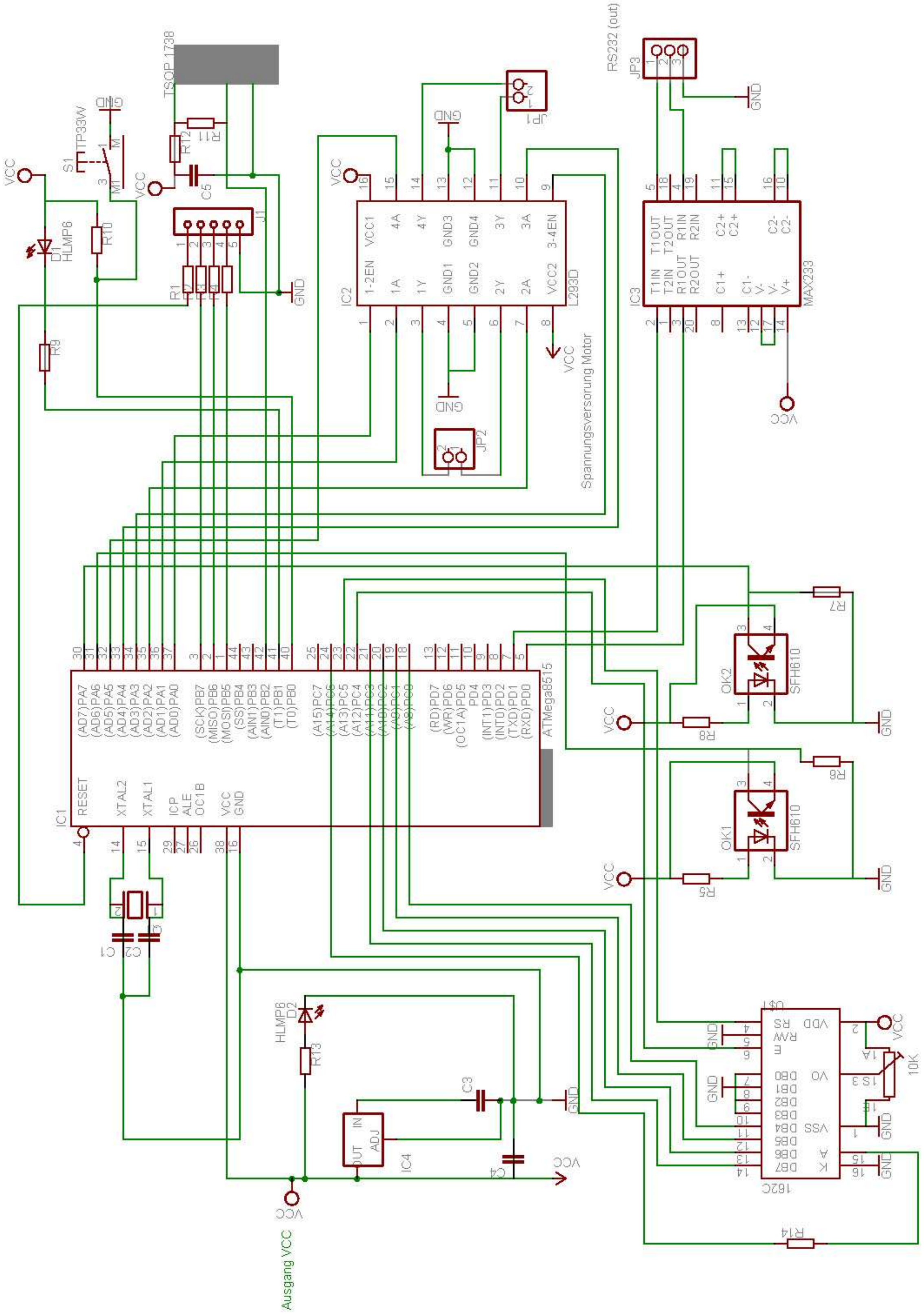


Figure 3.16: Wire diagram of the rotating head

3.8 Summary

This chapter explained the methods used to solve the reconstruction problem. The first part describes the acquisition system and the geometry used. It has been shown that an almost parallel camera system which is 5cm translated in x direction is appropriate. The calibration method was used to describe the parameters of the camera. After the system is modelled, features have to be extracted. Thus, the edge detection technique used and a new method to find straight lines in edge images is presented. Straight lines have been considered as a convenient feature to describe the environment of a football playing robot. When features are correctly extracted and described, the correspondence problem can be solved. The results are used to calculate the 3D position of features found, and the OpenGL library is used to display them. The last part covers the electronic diagram of the *rotating head* and the describes the main parts.

Chapter 4

Results

> *Man darf nicht glauben*
> *eine Idee könne durch den Beweis ihrer Richtigkeit*
> *selbst bei gebildeten Geistern Wirkungen erzielen.*
> *Man wird davon überzeugt, wenn man sieht,*
> *wie wenig Einfluß die klarste Beweisführung*
> *auf die Mehrzahl der Menschen hat.*
> *Der unumstößliche Beweis kann von seinem*
> *gebildeten Zuhörer angenommen worden sein,*
> *aber das Unbewußte in ihm wird ihn schnell zu*
> *seinen ursprünglichen Anschauungen zurückführen.*
>
> [George B. Shaw].

In this chapter, the accuracy of the implemented methods, which yield to a 3D reconstruction of the recorded environment, is investigated. First of all the cameras have to be calibrated. Every calibration technique formulates a mathematical model of a camera which is used to solve the equations that relate the known world coordinates and the projected points on the sensor plane. The model of the camera can also be used to re-project the world coordinates to image coordinates. For an estimation of the accuracy, the distance between re-projected coordinates and the coordinates used for calibration is measured.

The next step in the reconstruction is the line detection. If it worked perfectly it would cover all pixels of a straight line with a length more than the minimum length allowed. The results of the iterative line detector using synthetic and real data is presented in Section 4.2. Synthetic data has the benefit that an ideal environment and all possible special cases can be modeled. Using real data, the accuracy of the method in a real environment can be investigated.

In Section 4.3 the correspondence analysis is controlled, also using synthetic and real data. When corresponding lines are found, it can be compared with true values, both in a synthetic and a real world. The synthetic data is modeled using MAYA[1]. This can be done very easily, because it is possible to let the intrinsic geometry of the real camera and the synthetic camera in maya match. Unfortunately, lens distortion cannot be modeled thus cannot be investigated with synthetic data. The last part of this chapter reviews the 3D reconstruction itself.

4.1 Calibration

To measure the accuracy, the results of calibration (inner and outer geometry) are used to re-project the world coordinate points to image coordinates. In an ideal model, the image points and the re-projected points would perfectly match. Unfortunately, this does not happen. Figure 4.1 shows the re-projection using maya. The translation vector and the inverse rotation matrix was used to move and rotate the camera. The intrinsic parameters of the real camera and the camera used for re-projection in MAYA match. Unfortunately, distortion can not be modeled in MAYA, however the calibration pattern fits almost perfectly. The result can be used as a visualization to demonstrate the potentials but cannot be used to measure the accuracy of the calibration. The world coordinates have to be re-projected using a mathematical formulation in which lens distortion is modeled, i.e. the same model as used for calibration. The transformation that projects the world coordinates to image coordinates passes through the following stages.

$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} \rightarrow \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \rightarrow \begin{pmatrix} x_u \\ y_u \end{pmatrix} \rightarrow \begin{pmatrix} x_d \\ y_d \end{pmatrix} \tag{4.1}$$

---

[1] 3D program from Aliaswavefront

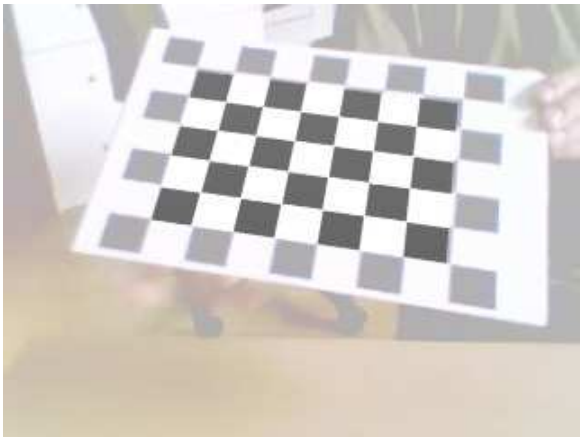


Figure 4.1: Re-projection of the calibration pattern in MAYA. Fog has been added to see the difference between the re-projected pattern and the original image. The bottom right rect fits best. The top right is slightly translated in plus y direction. The top right rect is translated in minus x and y direction and the bottom right is translated in minus x and plus y direction.

In other words, the 3D world coordinates are transformed (translated and rotated) to 3D camera coordinates. The intrinsic parameters of the camera are used to project the camera coordinates onto the 2D image plane. A 2D → 2D transformation models the distortion affect.

To calibrate the cameras, at least two images of the calibration pattern are needed. Figure 4.2 shows ten images recorded with the left camera. The patterns inside the images have different attitudes and positions in every image. This is needed,

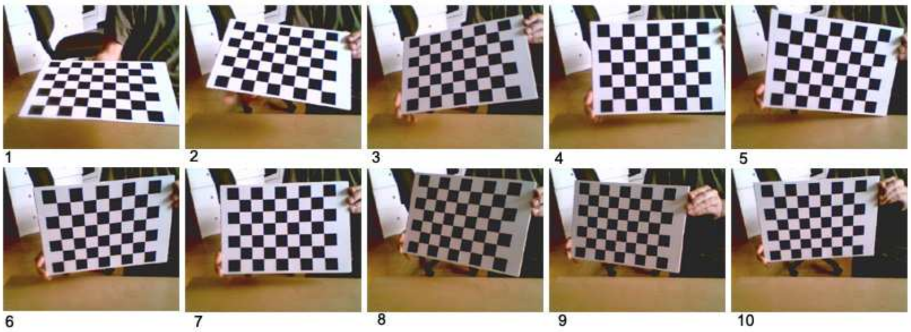


Figure 4.2: Images of the calibration pattern, recorded with the left camera of the stereo camera system

otherwise the calibration method would be unable to solve the calibration problem correctly. Once the calibration is done, the attitude of the calibration patterns, and more precisely, the translation and rotation in reference to the camera, is known. Figure 4.3 shows the reconstructed calibration patterns. We can use this information to re-project the calibration patterns onto the images. Figure 4.4 shows the result of re-projection of the upper left corner for each image. It is a zoomed view to see the deviation. The depicted cut-out has a size of 5x5px. Calculated edge points are marked as circles and re-projected points are marked as crosses. The estimator for the standard deviation of the difference between the original point and the re-projected one is called *pixel error*. It can be used as an estimation of the accuracy of the re-projection. Table 4.1 shows the re-projection error for every calibration image. It can be seen that the re-projection is very accurate. The calibration works

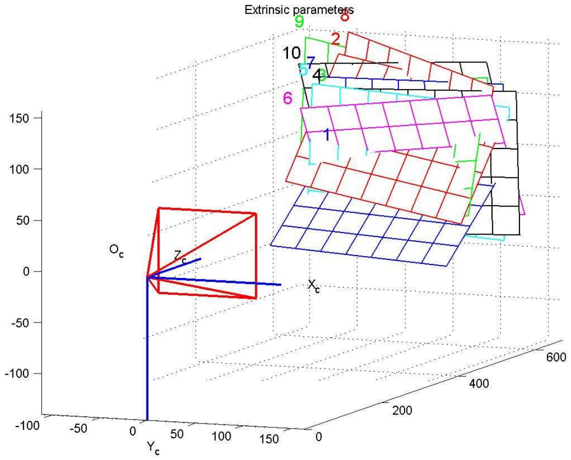


Figure 4.3: Reconstructed calibration pattern

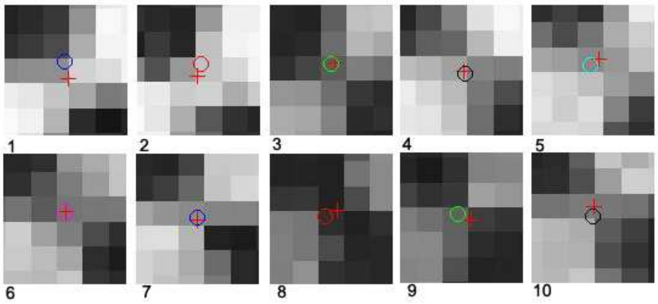


Figure 4.4: Calculated edge point ○ and re-projected point +

with subpixel accuracy, the mean pixel error is below 0.17 pixels. The mean and median of x and y are very small, and can be further decreased by adding more calibration images. Figure 4.5 shows the re-projection errors of the 10 calibration

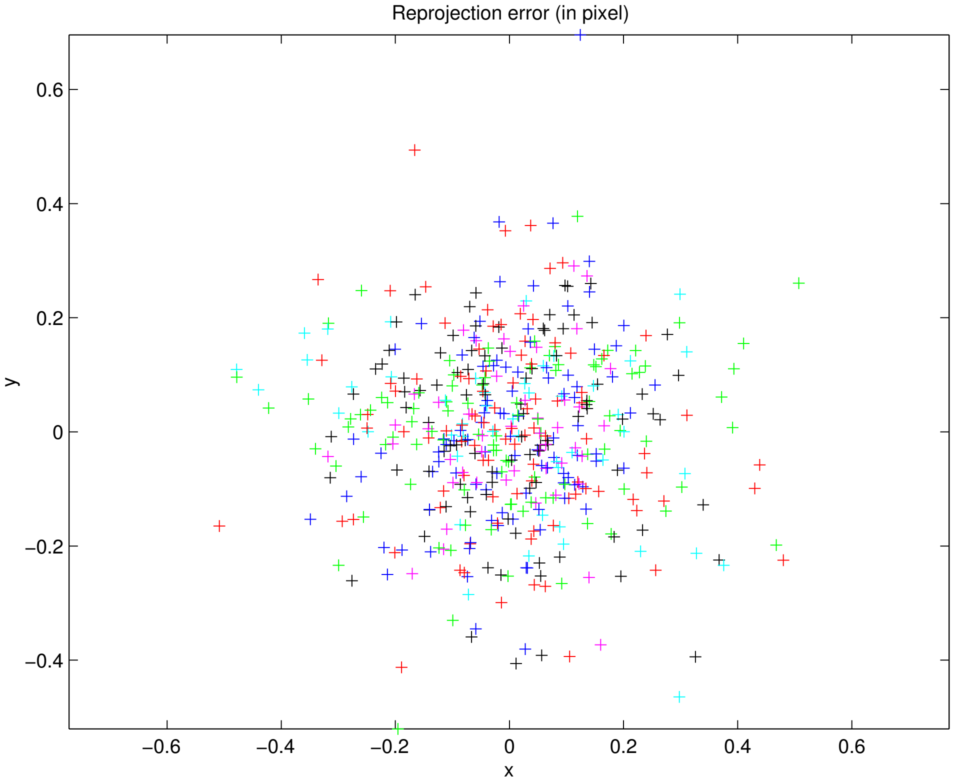


Figure 4.5: Re-projection error

images for the left camera. The calibration pattern has 48 internal corners, thus 48 points are plotted in an own color for every image. All deviations from the center are below 0.6 px.

Equation 4.2 shows the pixel error in closed form

$$\hat{\sigma}_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} X_i - \hat{\mu}} \tag{4.2}$$

In order to ensure unbiasedness, note that the divisor in Equation 4.2 is $N-1$ and not N, as it would be with knowledge about the true mean μ. The calculation of the estimator $\hat{\mu}$ is similar to the calculation of the mean μ.

The quality of the calibration has no influence to the quality of the line detection,

	Mean (x)	Mean (y)	Median (x)	Median (y)	$\hat{\sigma}_x$	$\hat{\sigma}_y$
Image 1	-0.0019	0.0009102	0.0039	-0.0382	0.1118	0.2048
Image 2	-0.0011	-0.0004358	-0.0106	-0.0368	0.1197	0.1943
Image 3	-0.0007	0.000634	-0.0018	0.000143	0.1935	0.1122
Image 4	-0.00004	0.000355	0.0034	0.0041	0.1570	0.1431
Image 5	-0.00056	0.000945	0.0282	0.0265	0.2096	0.1450
Image 6	0.0012	-0.000878	0.0167	-0.0045	0.1143	0.1328
Image 7	0.00025	0.000406	-0.00061	-0.0131	0.1306	0.1125
Image 8	0.0018	0.000240	0.0052	0.0067	0.2016	0.1349
Image 9	-0.00071	-0.0015	-0.0152	0.0261	0.2121	0.1571
Image 10	0.0018	0.0011	-0.0162	0.0293	0.1429	0.1750
$\forall$	0.00000	0.00000	0.0031	0.0020	**0.16248**	**0.15269**

Table 4.1: Re-projection error

which is investigated in the next section, but it has a strong influence on the correspondence analysis and the 3D reconstruction. Because the line detection works on rectified images, which are computed using the fundamental matrix F and F results of the prior calibration. If the rectification does not work properly, the attitudes of the lines are adulterated and thus correspondences may not be established. In addition, the disparity between two lines is also falsified. This has a direct influence to the 3D reconstruction, as well as other inner camera parameters have an influence to the final result (e.g. depth depends on effective focal length f_k, X_c and Y_c depends on principal point). If the camera coordinates are transformed into world coordinates, the error in the extrinsic camera parameters is reflected in the world coordinates.

4.2 Line Detection

The accuracy of the line detection is measured with *synthetic* and *real* data and defined by the difference of the manually and the automatic detection of line properties. The properties that are evaluated are

line origin The coordinate of the first point

line end The coordinate of the last point

attitude The attitude of the line

The attitude α is the most important property when it comes to correspondence analysis and is defined as

$$\alpha = \frac{x_l}{y_l} \tag{4.3}$$

where $x_l = l_{x_e} - l_{x_s}$ and $y_l = l_{y_e} - l_{y_s}$. l_{C_T} are the coordinates of the line, C defines if x or y is used and T defines if it is the start or end point. Note that α is not defined when y_l is 0. In this special case α is set to x_l. But y_l is only 0 in the case of a horizontal lines, which cannot be used for the further computation. Due to the importance of the attitude in the following correspondence analysis the images have to be rectified or be the result of a parallel camera system[2] in order to solve the corresponding problem correctly.

4.2.1 Synthetic data

Due to the high number of lines ($8 * 4 = 32$) in Figure 4.6, only the vertical lines on the left side of the rectangles inside the image are investigated and shown in Table 4.2. As one can see in Figure 4.6, all lines have been found. The two horizontal lines at the bottom of object 1 and 5 have been rejected because they are perfectly horizontal and thus cannot be used for correspondence analysis. Normally the value, which classifies the lines into two groups to decide whether correspondence analysis make sense or not, is increased in order to get less outliers, as it is often with horizontal lines. In Table 4.2 you can see the properties (2D image coordinates of the start and end point, attitude) of the lines shown in Figure 4.6. The result, especially of the start and end points is very accurate. The maximum deviation is +/- 1px, but for example the line at the bottom of object 4 only covers about 85 % of the line. The error in the attitude increases if the contingent of the counter in x direction decreases. Although one has to keep in mind that it is an ideal situation.

In real world the contrast is lower and the edges are noisy. As a result often not the whole line is detected and thus the line detection extracts short lines. This is not a big problem when it happens in both images, the reconstructed line is just shorter. Obviously, it is a problem when it happens just in one image. The correspondence

---

[2] manually adjusted stereo systems suffer from positioning and alignment errors

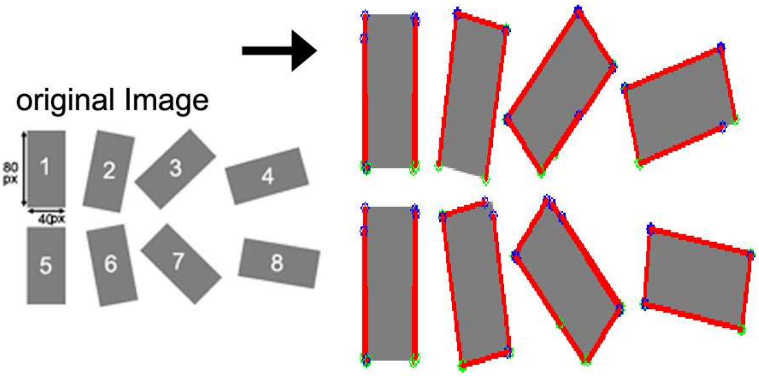


Figure 4.6: The image on the left side shows a synthetic image containing eight rectangles with different attitudes. On the right side the result of the line detection is displayed. The detected lines are colored red. The starting and end points are colored blue and green, respectively. In Table 4.2 you can see the result of the line detection.

Description	x_{end}	y_{end}	x_{start}	y_{start}	α	x'_{end}	y'_{end}	x'_{start}	y'_{start}	α'
Rect1	30	90	30	10	0	29	89	31	9	0
Rect2	87	87	103	9	-0.205	88	90	102	9	-0.177
Rect3	141	63	198	9	-1.055	141	64	197	8	-1.000
Rect4	245	87	234	49	0.289	245	86	234	48	0.289
Rect5	30	190	30	110	0	31	190	29	109	-0.167
Rect6	106	192	91	115	0.195	106	192	91	115	0.195
Rect7	203	190	147	134	1.000	203	189	148	135	1.000
Rect8	248	160	255	121	-0.179	249	160	254	121	-0.128
Error (μ)	-	-	-	-	manually	0.5	0.875	0.75	0.5	0.0376

Table 4.2: Real and detected coordinates, results are from synthetic data.

analysis compares different kinds of properties, amongst other things also the length and the similarity between the start and end point. If the difference between both lines is big enough, no correspondence can be established.

To get an idea how the line detection works in a more natural environment, the next section uses real data to gain its results. The coordinates of the lines are manually extracted in this case.

4.2.2 Real data

Different kinds of real objects can be used to evaluate the quality of an algorithm. But it must be proved, that the object used can be correctly and totally described. In the actual case the objects are also limited to those that can be detected thus produce straight lines when mapped onto the image plane. A ball for example will not be detected by the line detector. The objects used can be categorized into

Cube A metallic cube with dimensions (70x100x60,h-w-d,mm).

Cone A wooden cone with dimensions (79x76.5,r-h,mm)

Plane A plastic CD-case with dimensions (124x142,w-h,mm)

The reference values are manually extracted, the worst case is a displacement of +/- 1px. The values of the line detection result from four different images of the same objects. All object lines are investigated. The result is shown in Table 4.3. It can be

Description	#	x_{end}	y_{end}	x_{start}	y_{start}	α	x'_{end}	y'_{end}	x'_{start}	y'_{start}	α'
Cube 1	1	78	135	76	202	-0,030	78	136	75	206	-0,043
Cube 1	2	128	131	124	204	-0,055	127	129	122	210	-0,062
Cone 1	3	202	160	148	201	-1,317	200	162	149	202	-1,275
Cone 1	4	202	160	250	216	0,857	202	161	244	210	0,857
difference	-	-	-	-	-	-	-0,75	-0,5	2	-1,25	-0,006
Cube 2	1	78	135	76	202	-0,030	77	147	77	203	0,000
Cube 2	2	128	131	124	204	-0,055	128	144	122	208	-0,094
Cone 2	3	202	160	148	201	-1,317	199	163	159	194	-1,290
Cone 2	4	202	160	250	216	0,857	202	161	248	214	0,868
difference							-1	-7,25	-2	1	-0,007
Cube 3	1	78	135	76	202	-0,030	79	136	76	198	-0,048
Cube 3	2	128	131	124	204	-0,055	127	129	124	205	-0,039
Cone 3	3	202	160	148	201	-1,317	200	163	149	201	-1,342
Cone 3	4	202	160	250	216	0,857	202	161	245	211	0,860
difference							-0,5	-0,75	1	2	0,006
Cube 3	1	78	135	76	202	-0,030	79	136	76	198	-0,048
Cube 3	2	128	131	124	204	-0,055	128	144	122	212	-0,088
Cone 3	3	202	160	148	201	-1,317	201	161	149	202	-1,268
Cone 3	4	202	160	250	216	0,857	203	161	252	222	0,803
difference							0,250	-4,000	-0,250	-2,750	0,014

Table 4.3: Manually extracted and automatically extracted line properties. Results are derived from real data.

seen, that the differences between the real and the computed start and end points have remarkable increased in comparison to the line detection using synthetic data. Also the differences between the start and end points of the same object between different images varies with an approximate maximum of about 10% of the line length. Thus the points should not be taken as an important attitude when it comes to the correspondence analysis. In comparison, the difference of the attitude and the attitude itself varies only about 0.1. As a result, the attitude can be seen as a strong property, as well as the gradient direction of the line.

4.3 Correspondence Analysis

In order to construct a 3D image out of two 2D images, correspondences between both images have to be found, i.e. corresponding lines in both images. To get an estimation of the similarity between two lines the properties of the lines are used. The accuracy of these properties was shown in the last section.

4.3.1 Synthetic data

The synthetic scenes are built using *MAYA* software. They contain the same objects as described in Section 4.2.2. The cameras have their origin in $(0, 850, 0)[mm]$ and heading in $-y$ direction. The focal length of the cameras is $480px$, the baseline is $60mm$. The aperture is given by $(13.858, 11.266)[inch]$. The aperture values have to be in inches[3], no matter what global dimension is used. The cameras are totally parallel, thus no rectification is needed. The position of the objects in space is known and used to calculate the true disparities[4]. Figure 4.7 shows the synthetic left and right image. Also a perspective view of the maya environment is shown. Table 4.4 shows the true disparities and the calculated ones. Actually the disparity is calculated using pixel accuracy, but it could be easily extended to subpixel accuracy. In this case not only one disparity per line would be calculated. The maximum difference between the true and the calculated disparities is 1.7px. The mean disparity difference is **0.605** px, which is actually quite good for a pixel

---

[3] this is MAYA specific

[4] for the calculation of the disparity see Equation 2.2 but note that the camera is heading in $-y$ direction

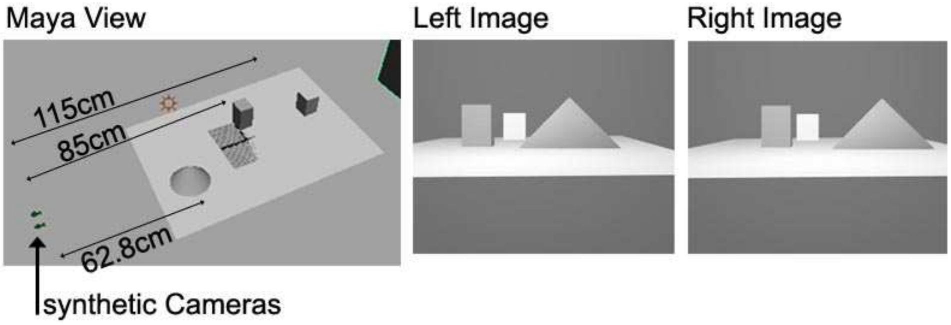


Figure 4.7: The figure shows a pair of synthetic images and a perspective view of the scene build in MAYA. The values are the distances between the cameras and the objects inside the scene.

Object	distance	disparity	calculated disparity	error[px]	error[cm]
Rect 1,l	85	33.88	34	0.12	0.3
Rect 1,r	85	33.88	34	0.12	0.3
Rect 2,l	115	25.04	24	1.04	3
Rect 2,r	115	25.04	26	0.86	4.3
Cone l	55	52.36	52	0.36	0.38
Cone r	55	52.36	52	0.36	0.38
Rect 1,l	75	38.4	38	0.4	0.7
Rect 1,r	75	38.4	39	0.6	1.2
Rect 2,l	95	30.3	31	0.7	2.1
Rect 2,l	95	30.3	30	0.3	1.0
Cone l	65	44.3	46	1.7	2.4
Cone r	65	44.3	45	0.7	1.0

Table 4.4: Results of correspondence analysis using synthetic data.

based disparity calculation. In cm it is a deviation of **1.421**. The accurate disparities results due to the accurate attitude of the detected line. The next section evaluates the correspondence analysis using real data.

4.3.2 Real data

The acquisition system used was already presented in Section 3.1, now it is used to measure the accuracy of the correspondence analysis using real data. The images have to be rectified to achieve correct results. The position and size of the objects in space is known and, as in the last chapter, used to calculate the true disparities.

The translation vectors are given by

$$T_l = \begin{pmatrix} 88.1702346802 \\ -79.7057495117 \\ 611.7904663086 \end{pmatrix} \qquad T_r = \begin{pmatrix} 141.6166229248 \\ -91.7347183228 \\ 617.5389404297 \end{pmatrix} \tag{4.4}$$

Using Equation 2.36, the baseline is given by $53.4464mm$. The focal length of the left camera is $[493.773, 496.006]$px. For the computation the bigger value of the focal lengths is used. Figure 4.8 shows the left and right image. The objects that are inside the view of the cameras are a wooden cone and a metallic cube. The properties of these objects have already been described in Section 4.2.2. Table 4.5 shows the true and the calculated disparities. The distance between the camera and

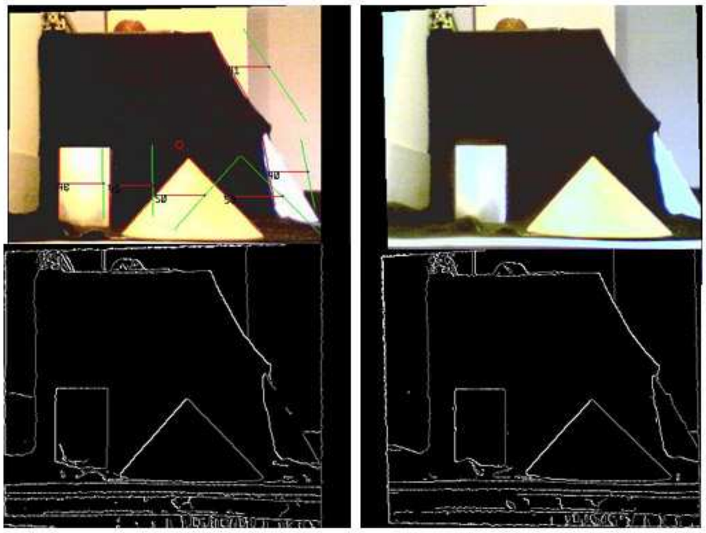


Figure 4.8: The figure shows a pair of real images and their corresponding edge images. The top left image shows also the result of the correspondence analysis. The red and green lines corresponds to each other.

Object	distance	disparity	calculated disparity	error[px]	error[cm]
Rect 1,l	60	44.1828	44 - 46	0.128,1.872	-0.2,2.4
Rect 1,r	60	44.1828	45	0.8171	1.1
Cone l	50	53.01	50-51	3.01-2.01	3,1.9
Cone r	50	53.01	58	4.99	4.3

Table 4.5: Results of correspondence analysis using real data.

the objects have been manually measured, thus are affected by an error which is about +/- 2cm. The maximum disparity error rises up to 5px. Which, in the case of Table 4.5, results in a deviation of 4.3 cm in the camera coordinate system. The mean disparity error is given by **2.12** px.

4.4 3D Reconstruction

After corresponding elements have been found the 3D coordinates of the features can be computed, i.e. the two dimensional coordinates of the start and end point of a line are transformed into three dimensional space coordinates. Using Equation 2.3, the coordinates are given in reference to the camera coordinate system of the right camera. In this case the optical axis and the z-coordinate of the coordinate system coincide. As in the previous section synthetic and real data is used to verify the accuracy of the 3D reconstruction.

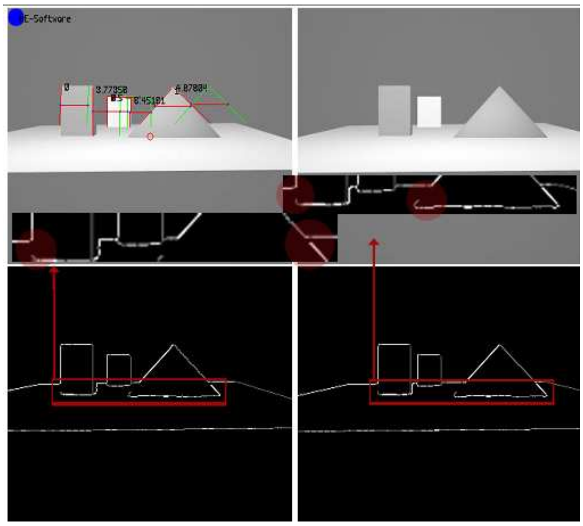


Figure 4.9: The zoomed views show the break up of lines. The red circles show the problematic spots. The top left image shows the image of the right camera, the found lines and the corresponding ones, the lower pictures shows the result of the Canny edge detector.

4.4.1 Synthetic data

The image that is investigated in Table 4.6 is shown in Figure 4.7. Several properties of the lines are investigated, amongst others the length of the reconstructed lines and the position. Table 4.6 shows the real properties and the result of the reconstruction. First the real values are shown, followed by the results of the reconstruction which are labelled by an apostrophe. The error in length results from the wrong starting points of all three objects. The correspondence analysis has already shown, that it works accurate with synthetic data. The reconstruction only depends on the results of the correspondence analysis as well as the calibration. Using synthetic data the calibration parameters are known, thus the result depends only on the correspondence analysis. Despite the y coordinate of the start point, all properties are accurately calculated. As depicted in Figure 4.9, the breaking up of lines is the problem that all y coordinates have falsified values. The edge detector, as well as the line detector has problems with the crossing from the background to the ground plane.

Object	Start	End	disparity	length
Rect 1,l	(-19.5,0,85)	(-19.5,10,85)	33.88	10
Rect 1,l'	(-19.6,2.3,84.7)	(-19.6,9.7,84.7)	34	7.4
Rect 1,r	(-12.5,0,85)	(-12.5,10,85)	33.8	10
Rect 1,r'	(-12.5,2.0,84.7)	(-12.5,9.5,84.7)	34	7.5
Cone 1,l	(-3.8,0,65)	(6,7.9,65)	44.31	11.1
Cone 1,l'	(-1.8,1.9,65.5)	(3.5,7.2,65.5)	44	7.49
Cone 1,r	(11.8,0,65)	(6,7.9,65)	44.31	11.1
Cone 1,r'	(10,2,64)	(3.7,7.5,64)	45	8.36
Rect 2,l	(-12.5,0,115)	(-12.5,10,115)	25.04	10
Rect 2,l'	(-12.7,2.6,115.2)	(-12.5,10.8,115.2)	25	8.2
Rect 2,r	(-5.5,0,115)	(-5.5,10,115)	25.04	10
Rect 2,r'	(-5.8,2.5,110.8)	(-5.5,9.7,110.8)	26	7.2

Table 4.6: Results of the reconstruction using synthetic data.

4.4.2 Real data

To test the accuracy with real data the objects presented in Section 4.2.2 are used. Especially the reconstructed size and position of the objects is investigated. The objects are centered at the optical axis of the camera. Thus the position can be accurately estimated. A second technique would be to transform the coordinates to a world coordinate system and measure the difference in the world coordinate system. However, this is another possible source of error, thus the camera coordinate system is used. The coordinates of the objects are shown in Table 4.7 and given in cm. The notation is the same as used in Table 4.6. Due to the noise in the images, the coordinates vary with +/- 1 pixel. The maximum deviation of the x and z coordinates are 0.6cm and 5.4cm. The mean deviation of both coordinates

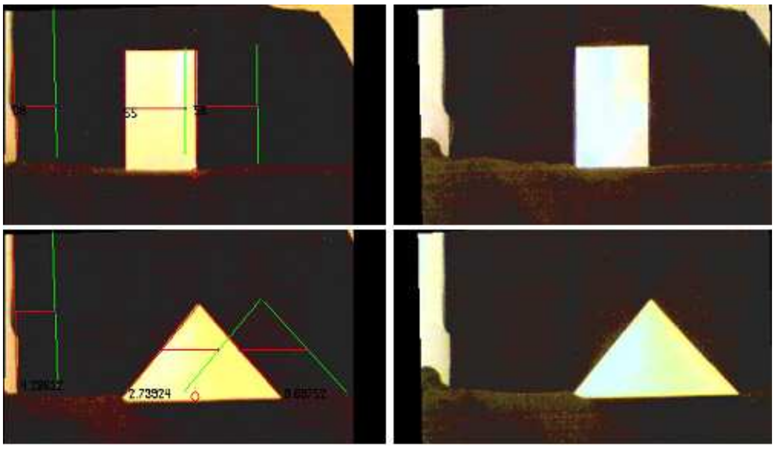


Figure 4.10: Real images of the cube and cone used for the accuracy measurement.

Object	Start	End	disparity	length
Cone l	(-7.8,0,50)	(0,7.9,50)	53.02	11.1
Cone l’	(-7.5,0.1,55.4)	(0.5,5.2,55.4)	48	8.66
Cone r	(7.8,0,50)	(0,7.9,50)	53.02	11.1
Cone r’	(7.2-7.4,1,45.7)	(0.4-0.5,4.3,45.7)	58	8.16
Cube l	(-7,0,50)	(-7,10,50)	53.02	10
Cube l’	(-6.9,0.2-1.3,51.4-52.4)	(-6.9,7.5-7.6,51.4-52.4)	55-56	7.3
Cube r	(0,0,50)	(0,10,50)	53.02	10
Cube r’	(0.1,0.5,50.5)	(0-0.1,7.3-7.5,50.5)	57-58	6.8-7.0

Table 4.7: Results of the reconstruction using real data.

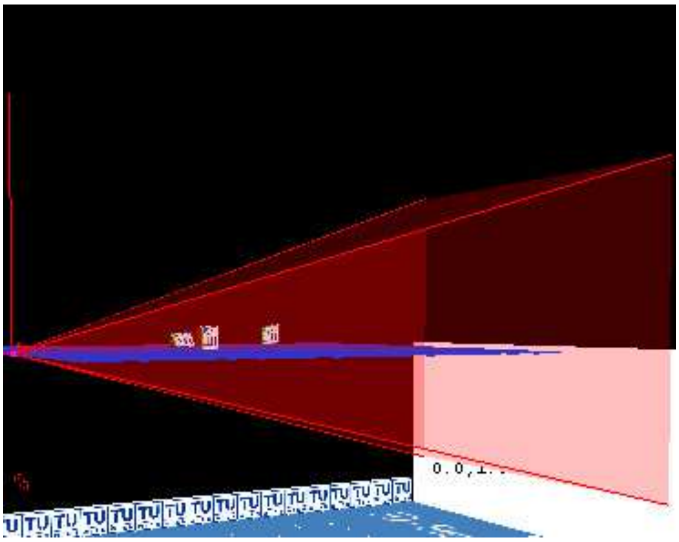


Figure 4.11: Result of the object detection. The original image is shown in Figure 4.7

is 0.2875cm and 3.025 which is accurate enough for autonomous football playing robots, only the y coordinate is not that accurate. The maximum deviation is 3.6cm, which means that only 54.4% of the line are detected. The mean deviation is 1.7125cm. The error results of the line detection. Sometimes pixels of a line are marked as processed, if the line started before a corner and stops not exactly at the corner. It would be a further improvement to deal with this special case. The disparity, as before, a maximum error of 5px. Figure 4.10 shows the images of the cone and the cube used for the accuracy measurement. After the 3D reconstruction of the lines, the algorithm tries to detect objects, in other words lines that belong together. Figure 4.11 shows the result using the synthetic image shown in Figure 4.7.

To conclude, the reconstruction works accurate on synthetic data, but the quality significantly decreases in reality. Nevertheless the algorithm reaches 48 frames per second with an image of size 352px x 288px (Figure 4.7) on a Intel Celeron processor with 1.4 GHz. Unfortunately, the cameras are only able to transfer 30 frames per seconds which results in 15 frames per second, if real data is used.

4.5 Summary

This section described the accuracy of all stages that are needed to build a sparse 3D model out of two 2D images. The algorithm shows good results using synthetic data. In a real environment the results are less accurate but are still close enough to guide a robot through a football field without bumping into other robots. Especially with synthetic data the speed of the algorithm can be tested because the bottleneck of transferring the images via USB to memory cease to apply.

Chapter 5

Conclusion and Future Work

It has been shown, that straight lines are appropriate for modeling the football field and the objects (except the ball) inside the view of an autonomous football playing robot. Additionally, it has been shown that a real-time algorithm can be implemented which produces a model that is close enough to the real world, in order to guide a robot through it.

Due to the fact that most engineers want to build small robots, two customary webcams are used to accomplish this goal. The sensors used in webcams are very similar to common small cameras. Unfortunately, they provided very noisy data, especially under bad lighting conditions which result in edges that often brake apart. In order to suppress this effect, the line detection tries to connect parts of lines that seem to correspond to each other.

The second part of my thesis was the construction of a robot head which is able to communicate with a computer via the serial interface or a LCD-display. The eyes can be rotated but have only one degree of freedom so far, even if the motor controller would be able to control two motors. The next step will be to mount the head onto a mobile robot which requires a wireless communication which has enough bandwidth to transmit at least 20 images per second which will result in a new 3D image every tenth part of a second. An alternative would be to do the calculations directly on the robot. A faster controller as well as more memory would be needed to accomplish this goal. Also a second motor controller will be needed to trigger two motors which will be responsible for the movement of the robot. The stereo head could also be mounted on a flying object. I considered to mount the cameras on a flying zeppelin. But there's still the need of a wireless communication.

The algorithm could be improved, especially the line detection should additionally control the direction of the edge gradient in order to decide if the pixel belongs to a line or not. The edge detector and the line detection could be combined to reach a higher frame rate. Another improvement would be to track the objects and predict the future position. This would lead to a more stable system. Especially when few frames are reached, a movement vector (result of the prediction) can be used to interpolate between the positions. This could be useful for aiming or hitting the ball.

It was interesting to see how our visual system and, as shown, a program can derive 3D information from two views of the same area. Hopefully, the results in this area, especially in the appliance of 3D information, will lead to more intelligent robots.

Bibliography

[AAK71] Y.I. Abdel-Aziz and H.M. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry . In *ASP Symposium on Close Range Photogrammetry*, pages 1–18, Falls Church, 1971. American Society of Photogrammetry (ASP).

[Ali02] Louis-Philippe Morency Ali. Fast 3d model acquisition from stereo images. In *Proceedings of 3DPVT*, 2002.

[BB81] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *Proc. of the 7th IJCAI*, pages 631–636, Vancouver, Canada, 1981.

[CF97] T.A. Clarke and J.G. Fryer. The development of camera calibration methods and models. In *Photogrammetric Record*, pages 51–66, April 1997.

[Dau85] J.G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by twodimensional visual cortical filters. *Journal of the Optical Society of America*, 2(A):1160–1169, 1985.

[DH73] R.O. Duda and P.E. Hart. *Pattern Recognition and Scene Analysis.* John Wiley and Sons, 1973.

[DSMMN02] L Di Stefano, M Marchionni, S Mattoccia, and G Neri. Dense stereo based on the uniqueness constraint. In *16th IEEE-/IAPR Interna-*

tional Conference on Pattern Recognition (ICPR'2002), pages III: 657–661, Quebec City, Canada, Aug 2002.

[Fai75] W. Faig. Calibration of close-range photogrammetry systems: Mathematical formulation. In *Photogrammetric Eng. Remote Sensing*, pages 1479–1486, 1975.

[Fau93] O.D. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, 1993.

[FB86] M A Fischler and R C Bolles. Perceptual organization and curve partitioning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):100–105, 1986.

[FJJ91] D. J. Fleet, A. D. Jepson, and M. R. M. Jenkin. Phase-based disparity measurement. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):198–210, march 1991.

[Gab46] D. Gabor. Theory of communication. *J. I.E.E. (London)*, 93:429–457, 1946.

[GR76] W.L. Gulick and B. Lawson R. *Human Steropsis.* Oxford University Press, 1976.

[HB93] Ph. W. Besslich H. Bässmann. *Bildverarbeitung Ad Oculus.* Springer Verlag, 1993.

[HIG02] Heiko Hirschmüller, Peter R. Innocent, and Jon Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *Int. J. Comput. Vision*, 47(1-3):229–246, 2002.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference (Manchester University, 31st August–2nd September)*, pages 147–152. The University of Sheffield Printing Unit, 1988.

[HZ00] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000.

[IPS85] A. Isaguirre, P.Pu, and J. Summers. A new development in camera calibration: Calibrating a pair of mobile cameras. In *Proc. IEEE Int. Conf. on Robotics and Automat*, pages 74–79, 1985.

[KKK$^+$95] Takeo Kanade, H. Kano, S. Kimura, E. Kawamura, A. Yoshida, and K. Oda. CMU video-rate stereo machine. In *Mobile Mapping Symposium*, pages 549–557, May 1995.

[KO94] Takeo Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, September 1994.

[KZ02] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV (3)*, pages 82–96, 2002.

[LT88] R. K. Lenz and R. Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):713–720, 1988.

[Mal00] H.A. Mallot. *Sehen und die Verarbeitung visueller Information - Eine Einführung.* Vieweg, Wiesbaden, 2000.

[MBK71] H.A. Martins, J.R. Birk, and R.B. Kelley. Camera models based on data from two calibration planes. In *Computer Graphics and Image Processing*, pages 17:173–180, 1971.

[MCTM05] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Auton. Robots*, 18(1):81–102, 2005.

[MK97] Christian Menard and Walter G. Kropatsch. Adaptive stereo matching in correlation scale-space. In *ICIAP (1)*, pages 677–684, 1997.

[MT79] D. Marr and T.Poggio. A computational theory of human stereo vision. In *Proceedings of Royal Society of London*, pages 301–328, 1979.

[OCV] Intel open source computer vision library, beta 3.1.

[PVK05] J. M. Porta, J. J. Verbeek, and B. J. A. Kröse. Active appearance-based robot localization using stereo vision. *Auton. Robots*, 18(1):59–80, 2005.

[Res01] Intel Research. OpenCV manual. Technical report, Intel Microcomputers, 1999 - 2001.

[San88] T. Sanger. Stereo disparity computation using Gabor filters. *Biological Cybernetics*, 59:405–418, 1988.

[SB97] Stephen Se and Michael Brady. Stereo vision-based obstacle detection for partially sighted people. In *Proceedings of the Third Asian Conference on Computer Vision-Volume I*, pages 152–159. Springer-Verlag, 1997.

[Sob73] I. Sobel. On calibration computer controlled cameras for perceiving 3-d scenes. In *Artificial Intelligence*, volume 5, pages 185–198, 1973.

[SS02] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[Tsa86] R.Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'86)*, pages 364–374, 1986.

[WO03] Andrew Wilson and Nuria Oliver. Gwindows: robust stereo vision for gesture-based control of windows. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 211–218. ACM Press, 2003.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.