TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DIPLOMARBEIT

Optical 3D Measurement using Calibrated Projector-Camera-Systems

Ausgeführt am Institut für
Diskrete Mathematik und Geometrie
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dr. Martin Peternell

durch
Maria Satzinger
Abraham a Sancta Clara-Straße 6
1140 Wien

Wien, Dezember 2012 _____

Contents

List of Figures

List of Tables

1 Abstract

Stereo vision is a part of the broad field Computer vision. It deals with the three-dimensional reconstruction of a scene. Therefore usually two cameras are mounted in parallel and capturing simultaneously. Furthermore, correspondences are searched between the different views using a stereo matching method in order to calculate depth by triangulation.

This thesis is about a special stereo vision system, a projector-camera sensor. Unlike classical stereo vision sensors where two cameras are used, one camera is replaced by a pattern projector. The three dimensional reconstruction is based on the same algorithms and processes. The only difference is, that not two camera images are used for the calculations but only one where the projected pattern is visible and in addition the pattern which is projected.

First, such a sensor has to be calibrated accurately. Since only one camera is available, the calibration method differs slightly from the classical stereo calibration procedure. In this thesis a projector-camera calibration method is presented where a precalibrated camera is needed to calculate all the intrinsic and extrinsic calibration parameters. If this information is available the rectification and stereo matching procedure can be performed and furthermore, depth can be computed by triangulation. The stereo matching method that is discussed here, is a pixel-based approach called census.

Furthermore, an extension of a classical two-camera stereo system by a pattern projector is presented. This special sensor, consisting of two cameras and a projector is therefor a combination of a two-camera system and a projector-camera system. The purpose of this combined sensor is to reduce the reconstruction difficulties of both individual systems to result in a denser combined depth map.

Next, the implemented software, including the projector-camera calibration, the rectification and stereo matching in MATLAB as well as the extensions of the existing High Speed Stereo Engine S3E developed by the Austrian Institute of Technology, is documented. Especially for the projector-camera calibration an automatic approach is developed and presented in this thesis which is based on the feature detection algorithm SIFT.

Finally, the projector-camera system is tested and verified. For that purpose, real-world scenes are measured and compared with the calculated results. Additionally the matching quality is analysed and it is shown that the best results are achieved by using a blurred version of the projected pattern for the stereo matching calculations. Furthermore, the depth maps of the projector-camera system are compared with the maps of the two-camera system as well as with the maps of the combined sensor.

2 Kurzfassung

Stereo Vision ist ein Teilgebiet von Computer Vision, das sich mit der dreidimensionalen Rekonstruktion einer Szene beschäftigt. Dies geschieht üblicherweise mit Hilfe von zwei Kameras, welche parallel zueinander montiert sind und zeitgleich aufnehmen. Des weiteren werden dann Punktkorrespondenzen zwischen diesen beiden Ansichten gesucht um Tiefenwerte durch Triangulierung zu berechnen.

In dieser Arbeit wird ein ganz spezielles Stereo Vision System vorgestellt, nämlich ein Projektor-Kamera Sensor. Im Gegensatz zu klassischen Stereo Vision Sensoren, bei denen zwei Kameras verwendet werden, wird hierbei eine der zwei Kameras durch einen Musterprojektor ersetzt. Die dreidimensionale Rekonstruktion basiert jedoch auf den selben Algorithmen und Prozessen, der einzige Unterschied besteht darin, dass nicht die zwei Kamerabilder einer Szene für die Berechnungen verwendet werden, sondern lediglich eines, in dem das projizierte Muster sichtbar ist. Statt dem zweiten Kamerabild wird das Muster selbst verwendet.

Zu aller erst muss dieser Stereo Vision Sensor genau kalibriert werden. Da nur eine Kamera im Einsatz ist, unterscheidet sich die Projektor-Kamera Kalibrierung ein wenig von der üblichen Stereo-Kamera Kalibrierung. In dieser Arbeit wird eine Projektor-Kamera Kalibriermethode vorgestellt, welche eine bereits kalibrierte Kamera benötigt um die entsprechenden intrinsischen und extrinsischen Parameter zu berechnen. Wenn diese Informationen vorhanden sind, kann die Rektifizierung und anschließend das Stereo Matching durchgeführt werden. Die Stereo Matching Methode, welche hier verwendet wird, Census genannt, ist ein Pixel-basierter Ansatz.

Weiters wird in dieser Arbeit eine Erweiterung eines klassischen Zwei-Kamera Systems mit einem Musterprojektor vorgestellt. Dieser spezielle Sensor, bestehend aus zwei Kameras und einem Projektor, ist daher eine Kombination eines Zwei-Kamera Sensors mit einem Projektor-Kamera Sensor. Der Grund, warum man solche Sensoren betrachtet, ist, dass Schwierigkeiten bei der Rekonstruktion beider Systeme reduziert werden können, um somit ein dichteres Tiefenbild zu erhalten, welches aus der Kombination der beiden Tiefenbilder berechnet wird.

Anschließend wird die implementierte Software, welche aus der Projektor-Kamera Kalibrierung, Rektifizierung und Stereo Matching in MATLAB als auch aus den Erweiterungen der bestehenden High Speed Stereo Engine S3E, welche vom Austrian Institute of Technology entwickelt wurde, dokumentiert. Im Detail wird in dieser Arbeit eine automatische Projektor-Kamera Kalibriermethode vorgestellt, welche auf dem Algorithmus SIFT, der zur Extraktion von Bildmerkmalen verwendet wird, basiert.

Zuletzt wird der Projektor-Kamera Sensor getestet und verifiziert. Zu diesem Zweck, werden reale Szenen vermessen und mit den berechneten Ergebnissen verglichen. Zusätzlich wird die Matching Qualität untersucht, wobei gezeigt wird, dass die besten Resultate erzielt werden, wenn eine verschwommene Version des projizierten Musters für die Stereo Matching Berechnungen verwendet wird. Weiters werden die Tiefenbilder des Projektor-Kamera Systems mit jenen des Zwei-Kamera Systems verglichen, als auch mit den Tiefenbildern des kombinierten Sensors.

3 Introduction

Three-dimensional reconstruction of a scene is a non-trivial problem. Human beings achieve this, by viewing their environment with two eyes, hence two different views of the same scene are available. The brain processes these impressions in order to obtain a three-dimensional image. But how does a computer reconstruct a three-dimensional scene, since it is not able to see what we see? One possible approach is to use optical measuring methods. Capturing a scene with a camera, means projecting the 3D points onto a two-dimensional image plane. This implies, that one dimension gets lost, a 3D point can be located only along a ray. Therefore additional information is necessary to recover the missing dimension. Today many different methods exist to deal with that problem. High reconstruction accuracy for example is provided by laser scanners, but they are expensive and mostly not usable in dynamic scenes. A modern and relatively cheap approach of optical 3D measurement is stereo vision or in other words the binocular viewing of a scene. If the visual perception of human beings is taken as an example, a second camera, which captures the scene simultaneously, has to be added. This view also provides the rays, where the 3D points are located. To reconstruct the coordinates of one such point, the two rays, that correspond to that point in each view, have to be intersected. This process is called triangulation. The problem here, is the identification of one and the same point within the different views, which is also called the *correspondence problem*. There exist different approaches to solve this problem. During this work, the rectification and stereo matching process will be discussed in detail. Rectification is the procedure that rotates the two images in such a way that corresponding image lines are aligned on the same row which simplifies the stereo matching procedure. Classical stereo vision sensors consist of two cameras. To perform the stereo algorithms, they have to be calibrated accurately. Camera calibration is the process of calculating the camera specific parameters which are necessary for 3D reconstruction. Stereo camera calibration is the procedure of computing the geometrical relationship between the two cameras, which is also needed to computationally measure a scene. In section 4, all the necessary information and details concerning camera calibration, the geometry, on which stereo vision is based, and the reconstruction procedure will be explained.

Classical two-camera stereo vision sensors suffer from different conditions. First of all the so-called *occlusions*, which are areas that are only visible in one of the two images, cannot be matched. Furthermore, textureless or repetitive areas make it difficult to achieve accurate results. One approach to overcome this, is to use structured light. Here, one camera is replaced by a pattern projector which illuminates the scene to be measured. Now correspondences are not searched between the two camera images, but between the projected pattern and the camera's image of it. Therefore some areas, which were not visible by the other camera can also be reconstructed as well as textureless parts of the scene due to the structure that comes from the projected pattern. In section 5 different structured light methods are presented as well as the most common procedures, called stereo matching algorithms, to find corresponding points between two images.

Since structured light sensors usually consist of a projector and a camera, they are also called projector-camera sensor. As with classical stereo vision systems, it has to

be calibrated before stereo vision algorithms, such as rectification, stereo matching and reconstruction, can be performed. A projector can be modeled as a camera with the only difference, that it is not able to capture the scene, but to illuminate it. Therefore the classical camera calibration procedure has to be modified slightly. The whole process is explained in section 6. Furthermore, a special stereo matching algorithm is depicted in detail and improvements are introduced.

What has not been taken into account up to now, is that projector-camera systems also have measuring difficulties. They differ from the ones of two-camera sensors, but does not have to be ignored. One of these problems is the ambient light. If it is too bright and the luminosity of the projector too low, the projected pattern is hardly visible in the camera's image and correspondences are difficult to find. A possible solution to overcome this, is to combine a two-camera sensor with a projector. This creates two or even three separate stereo vision systems, the two-camera system and the two projector-camera systems, one with the left camera and the second with the right one. Since the improvements, that are achieved by using two systems, are adequate and another additional system requires another calibration procedure and stereo matching stage, which also generates additional error sources, usually only the two-camera sensor is combined with one projector-camera system. This combined setup reduces the measuring problems from both sensors and leads to denser reconstruction results. The required modifications and algorithms are explained in section 6.

In section 7 all the implementations of the previously described algorithms are introduced. This includes the individual calibration procedures, the rectification and stereo matching algorithms. Furthermore, the existing High Speed Stereo Engine S3E developed by the Austrian Institute of Technology is depicted and the modifications and additional functions that were implemented during this work are explained.

Since the projector calibration method, described in section 6.1, is a semi-automatic procedure, which needs a lot of input from the user, a completely automatic procedure was developed during this work as well. It is based on the feature extracting algorithm SIFT and requires a precalibrated two-camera sensor, whereby one of the cameras is used for the projector-camera sensor as well. The details of this method are also explained in section 7.

To prove that the implemented algorithms provide accurate and correct results some evaluation setups are created and verification tests are performed. First of all the reconstruction results from the projector-camera system are evaluated. This includes the correctness, the measuring quality and the accuracy. Further the measuring outputs from the two-camera sensor and the projector-camera sensor are compared an their reconstruction difficulties are shown. Last some results of the combined stereo vision sensor are introduced. The whole evaluation and verification can be found in section 8.

4 Fundamentals of Stereo Vision

Stereo vision or stereo imaging describes the geometry and the relationships between two different views of one scene. It makes it possible to reconstruct the three-dimensional scene like our eyes do. Basically this is achieved by solving the so-called *correspondence problem*, which is the challenge to find corresponding points in both images. Once they are found, the depth of the scene can be calculated through triangulation. This stereo matching procedure can be simplified by rectifying both images, which means that the image planes are rotated as long as they get parallel. The main advantage of this procedure is, that the correspondence search only has to be performed along the image rows. To perform rectification, the stereo system has to be calibrated to get to know about the geometrical relationship between the two cameras and also the internal geometry of each one. So firstly the process of camera calibration is described before going into detail about the underlying geometry and 3D reconstruction.

4.1 Camera Calibration

Camera calibration is an important task in mathematically modeling a camera and it is used to calculate the camera's parameters. There are two types of parameters that should be known, the intrinsic and the extrinsic. The intrinsic parameters describe the internal camera geometry and the extrinsic the geometrical relationship between the camera coordinate system and the world coordinate system. These values will be calculated during the calibration process, but first it is important to understand how a camera is being modeled.

4.1.1 Camera Model

The simplest model of a camera is the pinhole camera model. Here, it is assumed that every point in space produces a single ray, which is projected on the image plane. This is illustrated in Figure 1. All these rays intersect in one point, called the *optical center* of the camera. This point is part of the focal plane. The distance between the image plane and the focal plane is called *focal length*. The distance Z between the camera and one object point can be easily calculated by similar triangles:

$$\frac{-x}{f} = \frac{X}{Z} \Rightarrow Z = f\frac{X}{-x}$$

where f is the focal length.

To make illustration and calculation easier the image plane and the focal plane are swapped. This doesn't affect the mathematical model, only the negative sign of x disappears so Z can be calculated as

$$Z = f\frac{X}{x}.$$

Now the projection of a point on the image plane is the intersection of the ray between that point and the optical center and the image plane.

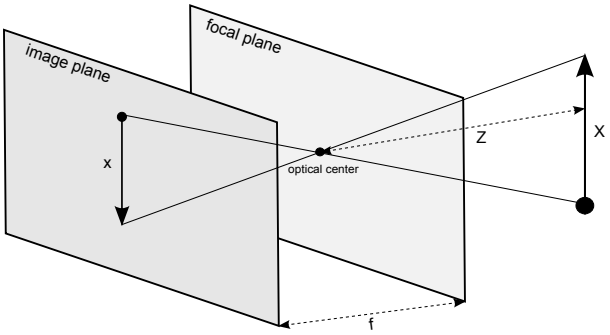


Figure 1: Pinhole camera

The camera coordinate system consists of the three orthogonal axes X_C, Y_C and Z_C and the origin C, which is the optical center of the camera. Z_C is pointing in the viewing direction and is called *optical axis*. Its intersection with the image plane is called *principle point*. The whole setup is shown in Figure 2.

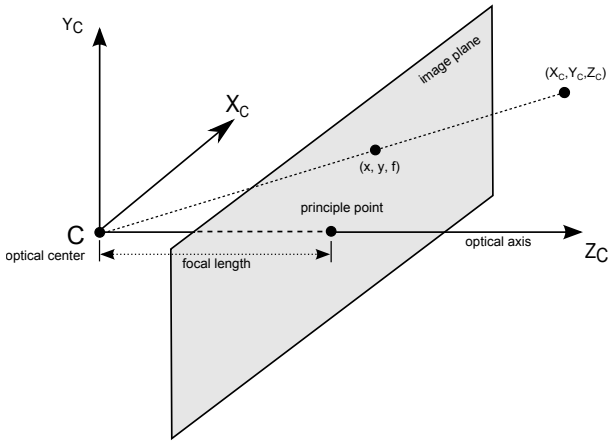


Figure 2: Camera model

The transformation between the world coordinate system, with axes X_w, Y_w and Z_w and origin O_w, and the camera coordinate system can be modeled by a rotation matrix R and a translation vector T and these two parameters are the so-called *extrinsic parameters* of the camera. The matrix R is defined by three normalized row vectors r_x, r_y and r_z, which represent the orientation of the three axis of the camera coordinate system with respect to the world coordinate system.

Given an object point P and its coordinates (X_w, Y_w, Z_w) relative to the world coordinate system, the coordinates (X_c, Y_c, Z_c) can be calculated as

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + T$$

or in homogeneous coordinates

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Next, the *intrinsic parameters* of the camera, which will model the projection of the point on the image plane onto the screen, will be described. One of these parameters has already been mentioned, the focal length, the distance from the optical center to the image plane. As explained above the 2D coordinates (x, y) of the point P on the image plane are

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z_c} \begin{pmatrix} X_c \\ Y_c \end{pmatrix}$$

which can be summarized to

$$\begin{pmatrix} U \\ V \\ S \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix},$$

where $S = Z_c$, hence $x = \frac{U}{S}$ and $y = \frac{V}{S}$.

Since the coordinate origin of the image plane is set at the principle point (u_0, v_0) and not at the corner of the screen, (u_0, v_0) has to be added. Another fact to mention is, that pixels often are not square as it would be thought, but rectangular. Therefore the physical focal length f is split into f_x and f_y, the focal lengths in units of pixels. So the matrix above changes to

$$A := \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

All together there are four intrinsic parameters, f_x, f_y, u_0 and v_0. So the projection of a 3D point (X_w, Y_w, Z_w) onto the screen can be summarized to

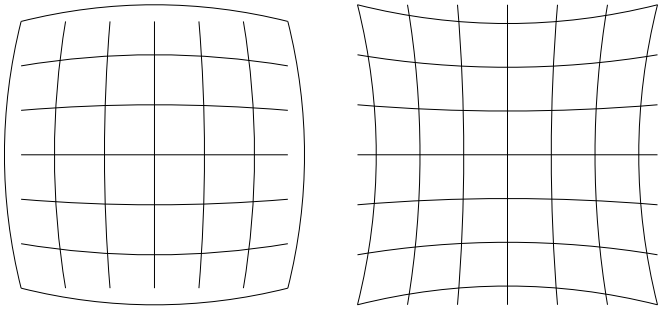
$$S \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = A \begin{pmatrix} R & T \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix},$$

where $A \begin{pmatrix} R & T \end{pmatrix}$ is the corresponding projection matrix.

What has not been taken into account up to now, is, that all cameras have a lens and so lens distortion has to be considered. There are two types of distortion, the radial and the tangential distortion. The radial distortion is caused by the curvature of the lens and causes for example that straight lines are curved on the image plane. We use a 6^{th} order polynomial to model the distortion as it is proposed by [5]. The corrected point on the image plane has the coordinates

$$\begin{pmatrix} x_{corr} \\ y_{corr} \end{pmatrix} = \begin{pmatrix} x + xk_1r^2 + xk_2r^4 + xk_3r^6 \\ y + yk_1r^2 + yk_2r^4 + yk_3r^6 \end{pmatrix},$$

where r is the distance of $\begin{pmatrix} x \\ y \end{pmatrix}$ from the origin, so $r = \sqrt{x^2 + y^2}$ and k_1, k_2 and k_3 are the radial distortion parameters. If k_1 is positive the distortion is called *barrel distortion* and otherwise it is called *pincushion distortion*. The effects are shown in Figure 3.



(a) Barrel (b) Pincushion

Figure 3: Different, distortion models (a) Barrel distortion and (b) Pincushion distortion.

The tangential distortion is an affect of the design of the lens, for example that the lens is not exactly parallel to the image plane. Here there are two parameters, p_1 and p_2 which model the distortion and the corrected coordinates are

$$\begin{pmatrix} x_{corr} \\ y_{corr} \end{pmatrix} = \begin{pmatrix} x + 2p_1xy + p_2(r^2 + 2x^2) \\ y + p_1(r^2 + 2y^2) + 2p_2xy \end{pmatrix}$$

4.1.2 Single Calibration

When calibrating a single camera the two extrinsic parameters, R and T, the four intrinsic parameters, f_x, f_y, u_0 and v_0, and the five distortion parameters, k_1, k_2, k_3, p_1 and p_2, have to be calculated. This is usually achieved with the help of a calibration object, for example a chessboard, which is a plane object and part of the model plane. It is assumed, that the plane's equation is $Z_w = 0$. So the origin of the world coordinate system is also part of this plane. Usually it is equivalent to one of the outer chessboard corners. The 3D coordinates of the other chessboard corners are easy to calculate, when the size in millimeters or centimeters of one square within the chessboard is known as well as the number of squares within the chessboard. During the calibration procedure the chessboard is viewed from different angles, hence the orientation of the world coordinate system changes with every different view. Each time all of the corners have to be detected in the camera's image. Afterwards the correspondences between the 3D-chessboard corners and the chessboard corners on the camera image plane are known, the camera's parameters can be estimated as it is proposed in [6].

First it is pretended, that there is no radial and tangential distortion so the image coordinates are

$$\begin{pmatrix} U \\ V \\ 1 \end{pmatrix} = s \cdot A \begin{pmatrix} r_1 & r_2 & r_3 & T \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ 0 \\ 1 \end{pmatrix} = s \cdot A \begin{pmatrix} r_1 & r_2 & T \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ 1 \end{pmatrix}$$

Set $H = [h_1\ h_2\ h_3] := s \cdot A \begin{pmatrix} r_1 & r_2 & T \end{pmatrix}$, where s is an arbitrary scale factor. Then the three 3×1 vectors $h_1 = sAr_1, h_2 = sAr_2$ and $h_3 = sAT$ have to be calculated. There are two constraints that have to be taken into account. We can use the fact that r_1 and r_2 are rotation vectors, so their dot product is zero, $r_1^{\mathrm{T}} r_2 = 0$, and their magnitudes are equal, $r_1^{\mathrm{T}} r_1 = r_2^{\mathrm{T}} r_2$. In both equations the scale factor $\lambda = \frac{1}{s}$ is extracted. Substituting r_1 and r_2 gives the two constraints

$$h_1^{\mathrm{T}} A^{-T} A^{-1} h_2 = 0 \text{ and } h_1^{\mathrm{T}} A^{-T} A^{-1} h_1 = h_2^{\mathrm{T}} A^{-T} A^{-1} h_2$$

Set $B = A^{-T} A^{-1}$. It follows that

$$B = \begin{pmatrix} \frac{1}{f_x^2} & 0 & \frac{-u_0}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-v_0}{f_y^2} \\ \frac{-u_0}{f_x^2} & \frac{-v_0}{f_y^2} & \frac{u_0^2}{f_x^2} + \frac{v_0^2}{f_y^2} + 1 \end{pmatrix}$$

Combined with the two constraints it gives

$$h_i^{\mathrm{T}} B h_j = \underbrace{\begin{pmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{pmatrix}^{\mathrm{T}}}_{:=v_{ij}^{\mathrm{T}}} \underbrace{\begin{pmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{pmatrix}}_{:=b} \qquad i, j \in \{1, 2\}$$

and so

$$\begin{pmatrix} v_{12}^{\mathrm{T}} \\ (v_{11} - v_{22})^{\mathrm{T}} \end{pmatrix} b = 0.$$

So when the calibration object is viewed from n different angles, we get n such equations and can then build the $2n \times 6$ matrix V by putting them together and having the following equation

$$Vb = 0$$

which can be solved for b. The solution is the eigenvector of $V^{\mathrm{T}}V$ associated with the smallest eigenvalue. Now it is easy to calculate the intrinsic parameters:

$$
\begin{aligned}
v_0 &= \frac{b_2 b_4 - b_1 b_5}{b_1 b_3 - b_2^2} \\
\lambda &= b_6 - \frac{b_4^2 + v_0(b_2 b_4 - b_1 b_5)}{b_1} \\
f_x &= \sqrt{\frac{\lambda}{b_1}} \\
f_y &= \sqrt{\frac{\lambda b_1}{b_1 b_3 - b_2^2}} \\
u_0 &= \frac{-b_4 f_x^2}{\lambda}
\end{aligned}
$$

The extrinsic parameters can also be computed:

$$
\begin{aligned}
r_1 &= \lambda A^{-1} h_1 \\
r_2 &= \lambda A^{-1} h_2 \\
r_3 &= r_1 \times r_2 \\
T &= \lambda A^{-1} h_3
\end{aligned}
$$

where $\lambda = \frac{1}{||A^{-1}h_1||}$. Here, it should be noticed that these parameters are different for every different view of the chessboard because the world coordinate system changes.

The next step during the calibration process is the calculation of the lens distortions. Once we know the intrinsic and extrinsic parameters, the pixel coordinates for every 3D point can be computed, but this calculated 2D point does not conform to the actual corresponding 2D point because of distortion. As explained above the relationship between the calculated point and the correct point is

$$
\begin{pmatrix} x_{corr} \\ y_{corr} \end{pmatrix} = \begin{pmatrix} x + xk_1r^2 + xk_2r^4 + xk_3r^6 \\ y + yk_1r^2 + yk_2r^4 + yk_3r^6 \end{pmatrix} + \begin{pmatrix} 2p_1xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2xy \end{pmatrix}
$$

Many of these equations together form a system of equations which can be solved in order to obtain the distortion parameters k_1, k_2, k_3, p_1 and p_2.

4.1.3 Stereo Calibration

Stereo systems usually consist of two cameras which are used to reconstruct a 3D scene. For this task it is very important to know the geometrical relationship between the two cameras. Stereo calibration is the process which handles this. In fact the goal is to calculate the rotation R and the translation T between the two different coordinate systems.

First, both cameras have to be calibrated on their own, with the only constraint that they are viewing the calibration object simultaneously, so that the rotations R_l and R_r and the translations T_l and T_r for each camera and every different view refer to the same world coordinate system. Given a 3D point P in the world coordinate system its corresponding point P_l in the left camera coordinate system and P_r in the right camera coordinate system can be calculated as

$$
P_l = R_l P + T_l \quad \text{and} \quad P_r = R_r P + T_r
$$

This implies

$$R_l^{\mathrm{T}}(P_l - T_l) = R_r^{\mathrm{T}}(P_r - T_r) \quad \Rightarrow P_r = R_r R_l^{\mathrm{T}} P_l - R_r R_l^{\mathrm{T}} T_l + T_r$$

$$\Rightarrow R = R_r R_l^{\mathrm{T}} \quad \text{and} \quad T = T_r - R_r R_l^{\mathrm{T}} T_l$$

So the rotation and translation that maps the right camera coordinate system to the left camera coordinate system can easily be calculated when the two cameras are calibrated. As explained in the previous section R_l and R_r are different for every different view during the calibration process of the left camera and the right camera, hence R and T also differ slightly. To obtain the final extrinsic parameters, the median of all resulting $R's$ and $T's$ could be taken.

The reference coordinate system here is fixed at the left camera coordinate system. So the left projection matrix is $A_l \begin{pmatrix} I & 0 \end{pmatrix}$ and the right is $A_r \begin{pmatrix} R & T \end{pmatrix}$.

4.2 Epipolar Geometry

The epipolar geometry is the basic geometry when working with stereo systems. It combines the geometry of the two cameras and makes 3D reconstruction easier. The matrix, which describes the whole epipolar geometry, is called fundamental matrix and will be explained in this section. Also a special stereo camera setup where the image planes of the cameras are parallel will be described.

4.2.1 Definitions

Before going into detail on the mathematical issues of the epipolar geometry, some terms have to be defined. As explained in the previous section each camera has an optical center, let C_l be the optical center of the left camera and C_r the one of the right. The line b, that connects these points is called *baseline*. This line intersects the image plane of the left camera as well as the image plane of the right camera. These intersection points, e_l and e_r, are called *epipoles* and the plane formed by them and a point P in space is called *epipolar plane*. This plane intersects the two image planes in the *epipolar lines*, which also contain the epipoles. So all the epipolar lines of one image plane intersect in this special point. The whole geometry is illustrated in Figure 4.

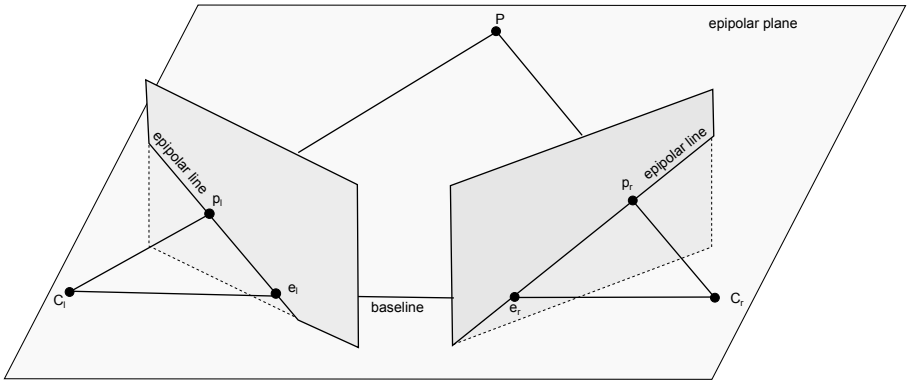


Figure 4: Epipolar geometry

A very useful fact concerning epipolar geometry is, that it makes it easier to find point correspondences in the left and right image. Let p_l be the projection of the point P on the left image plane. What we are searching for is the corresponding projection p_r on the right image plane. The epipolar geometry makes this search easy, because of the so called *epipolar constraint*, which says that p_r must lie along the corresponding epipolar line on the right image plane. Therefore the correspondence search becomes a one-dimensional search once the epipolar geometry of the stereo system is known.

4.2.2 The Essential and Fundamental Matrix

To solve the correspondence problem for a 3D point P considering the epipolar constraint, the epipolar lines have to be calculated. For this purpose two matrices, the *essential matrix* E and the *fundamental matrix* F are introduced.

Given an object point P, we already know that the points

$$P_l = \begin{pmatrix} X_l \\ Y_l \\ Z_l \end{pmatrix} \text{ and } P_r = \begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix}$$

in the corresponding camera coordinate systems are related according to a rotation R and a translation T, $P_r = RP_l + T$. The coordinates of the appropriate points on the image planes are

$$p_l = S_l \begin{pmatrix} x_l \\ y_l \\ 1 \end{pmatrix} = \frac{f_l}{Z_l} \begin{pmatrix} X_l \\ Y_l \\ Z_l \end{pmatrix} \text{ and } p_r = S_r \begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix} = \frac{f_r}{Z_r} \begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix}$$

Using the relationship between left and right coordinate system it follows that

$$S_r p_r = S_l R p_l + T \text{ or } p_r = \frac{f_r}{Z_r}(\frac{Z_l}{f_l} R p_l + T)$$

$$\Rightarrow T \times p_r = \frac{f_r}{Z_r}\frac{Z_l}{f_l} T \times R p_l \;\Rightarrow\; 0 = p_r^{\mathrm{T}}(T \times p_r) = p_r^{\mathrm{T}}(\frac{f_r}{Z_r}\frac{Z_l}{f_l} T \times R p_l)$$

and finally

$$p_r^{\mathrm{T}}(T \times R p_l) = 0$$

The cross product $T \times R$ can be rewritten as $E := [T]_x R$ with

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

leading to the final *epipolar equation*

$$p_r^{\mathrm{T}} E p_l = 0$$

where E is the previous mentioned essential matrix. It contains all the information about the geometrical relationship between corresponding image points. So with this equation

it is possible to calculate the epipolar line on the right image plane of a given point p_l on the left image plane, which contains all possible corresponding points p_r on the right image plane. It is simple $l_l = Ep_l$ and the one on the left image plane is $l_r = E^{\mathrm{T}}p_r$. Since all the epipolar lines l_l contain the epipole e_l, it follows that $0 = e_l^{\mathrm{T}}(E^{\mathrm{T}}p_r) = (e_l^{\mathrm{T}}E^{\mathrm{T}})p_r$ for all p_r and further $Ee_l = 0$. The equation for the right epipole can be computed analog, $e_r^{\mathrm{T}}E = 0$. If we use the fact, that one epipole is the corresponding point of the optical center of the other camera, the right epipole can be computed easily. The optical center of the left camera has the coordinates $(\ 0 \quad 0 \quad 0\ )^{\mathrm{T}}$, so the epipole e_r can be computed as follows

$$p_l = S_l \begin{pmatrix} x_l \\ y_l \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow S_l = 0 \Rightarrow S_r e_r = 0 \cdot R + T$$

$$\Rightarrow e_r = \lambda T$$

To calculate the left epipole we take a closer look at the equation $Ee_l = 0$. Let us rewrite it to $0 = SRe_l = T \times Re_l$. What follows is that T and Re_l are parallel, so we can set $T = Re_l$ and therefore $e_l = R^{\mathrm{T}}T$.

In practice we are interested in pixel coordinates, so some information about the camera's geometries have to be added. Let A_l be the left camera's intrinsic matrix and A_r the one of the right. The screen coordinates, q_l and q_r, can be calculated as $A_l p_l$ and $A_r p_r$. Now the epipolar equation can be rewritten as

$$q_r^{\mathrm{T}}(A_r^{-1})^{\mathrm{T}}EA_l^{-1}q_l = 0.$$

The fundamental matrix F is defined as $(A_r^{-1})^{\mathrm{T}}EA_l^{-1}$ and therefore we have the relationship for the pixel coordinates:

$$q_r^{\mathrm{T}}Fq_l = 0.$$

The epipolar lines in pixel coordinates can be computed as Fq_l and $F^{\mathrm{T}}q_r$ and the epipoles are A_rT and $A_lR^{\mathrm{T}}T$.

If we use the equality $(A_r^{-1})^{\mathrm{T}}[T]_x = \frac{1}{det(A_r)}[A_rT]_xA_r$, we get another form for F:

$$0 = q_r^{\mathrm{T}}(A_r^{-1})^{\mathrm{T}}[T]_xRA_l^{-1}q_l = \frac{1}{det(A_r)}q_r[A_rT]_xA_rRA_l^{-1}q_l$$

$$\Rightarrow q_r[A_rT]_xA_rRA_l^{-1}q_l = 0 \ \Rightarrow \ F = [A_rT]_xA_rRA_l^{-1}$$

$$F = [e_r]_xA_rRA_l^{-1}$$

4.2.3 Parallel Camera Setup

When the image planes of the two cameras are parallel, we are talking about a parallel camera setup. It is a special configuration and makes some calculations easier. Since the image planes are parallel, the rotation matrix R is equal to the identity matrix and the optical centers are horizontal translated according to the translation vector T, which is

therefore parallel to the x-axis. So its coordinates are $(\ b \quad 0 \quad 0\ )^{\mathrm{T}}$, where b is the length of the baseline. It follows, that the epipole e_r is $(\ 1 \quad 0 \quad 0\ )^{\mathrm{T}}$. Further it is assumed, that the two intrinsic matrices, A_l and A_r are equal. Then the left projection matrix is $A_l(\ I \quad 0\ )$ and the right is $A_l\begin{pmatrix} I & T \end{pmatrix}$.

Now the fundamental matrix can be computed using all of this information:

$$F = [e_r]_x A_l R A_l^{-1} = [e_r]_x A_l A_l^{-1} = [e_r]_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

Let $p_l = (\ x_l \quad y_l \quad 1\ )^{\mathrm{T}}$ be a point on the left image plane. Then the corresponding epipolar line on the other image is

$$Fp_l = \begin{pmatrix} 0 \\ -1 \\ y_l \end{pmatrix}$$

which can be written in implicit form $ax + by + c = 0$ with $a = 0, b = -1$ and $c = y_l$, leading to the final explicit form $y = y_l$. A consequence is, that the corresponding epipolar line has the same y-coordinate as the given image point and that every epipolar line is parallel to the x-axis. Since the epipole is the intersection of all the epipolar lines of one image, it follows, that this point is at infinity. The epipole is also the intersection of the baseline and one image plane and therefore the baseline is parallel to the x-axis as well. As a consequence the correspondence problem is easier to solve, because the epipolar lines doesn't have to be calculated explicitly, we only have to search along the line with the same y-coordinate parallel to the x-axis. So it would be desirable, that every camera setup can be transformed to this parallel setup and this will be explained in the following section.

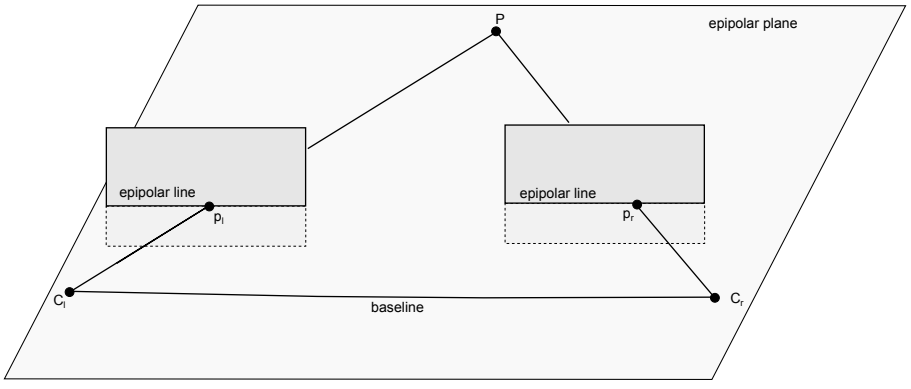


Figure 5: Parallel camera setup

4.3 Rectification

Rectification is the process of transforming the two image planes of any stereo system to a parallel setup, where corresponding epipolar lines are aligned on the same row and all of

them are parallel to the x-axis. It is assumed, that the stereo rig is calibrated, hence the intrinsic parameters of both cameras are known, as well as the rotation and translation between them. The two projection matrices have the form $A_l(\ I \quad 0\ )$ and $A_r(\ R \quad T\ )$. To perform the rectification procedure, two new projection matrices $A_1(\ R_1 \quad T_1\ )$ and $A_2(\ R_2 \quad T_2\ )$ have to be calculated in order to get epipolar lines parallel to the x-axis and further to align corresponding epipolar lines on the same row. The new intrinsic matrices are the same for both cameras, since the parallel image planes imply same focal lengths and principle points. There exist different methods to calculate the new parameters. Some set the new intrinsic matrices equal to one of the old ones, while others take care about the visible area after rectification to be maximal. The second approach will be described in section 7.2.3.

The rotation matrices R_1 and R_2 are composed of two rotation matrices, $R_1 = R_{new}R_l$ and $R_2 = R_{new}R_r$. R_l and R_r are the matrices that rotate left and right image planes to be parallel. Here, we fix the left camera, hence R_l is equal to the identity matrix and R_r is the inverse of the rotation matrix R between the two image planes.

R_{new} represents the rotation matrix, that turns the image lines to be parallel to the x-axis. The three normalized row vectors of R_{new} represent the orientations of the three axes. Because we want the new x-axis to be parallel to the baseline, the first row of R_{new} is $r_x = \frac{(C_r - C_l)^{\mathrm{T}}}{||C_r - C_l||} = \frac{T^{\mathrm{T}}}{||T||}$. The second vector has to be orthogonal to the first one, but has otherwise no constraints, so set $r_y = \frac{(\ -T_y \quad T_x \quad 0\ )^{\mathrm{T}}}{\sqrt{T_x^2 + T_y^2}}$. The third row is then $r_z = r_x \times r_y$.

Another approach to calculate the rotation matrix, takes the angle between the baseline, i.e. $C_r - C_l = T$, and the x-axis into account. The rotation axis is equal to the unit normal vector of the plane, which is spanned though the x-axis and T. In general the matrix for a rotation by an angle α around an axis $\begin{pmatrix} x & y & z \end{pmatrix}$, which is represented by an unit vector, is given by

$$R = \begin{pmatrix} (1-\cos(\alpha))x^2 + \cos(\alpha) & (1-\cos(\alpha))xy - \sin(\alpha)z & (1-\cos(\alpha))xz + \sin(\alpha)y \\ (1-\cos(\alpha))xy + \sin(\alpha)z & (1-\cos(\alpha))y^2 + \cos(\alpha) & (1-\cos(\alpha))yz - \sin(\alpha)x \\ (1-\cos(\alpha))xz - \sin(\alpha)y & (1-\cos(\alpha))yz + \sin(\alpha)x & (1-\cos(\alpha))z^2 + \cos(\alpha) \end{pmatrix}$$

Here, the rotation axis is equal to $\frac{T \times \begin{pmatrix} -1 & 0 & 0 \end{pmatrix}^{\mathrm{T}}}{||T||} = \begin{pmatrix} 0 & \frac{-T_z}{\sqrt{T_y^2 + T_z^2}} & \frac{T_y}{\sqrt{T_y^2 + T_z^2}} \end{pmatrix}^{\mathrm{T}}$ and the angle $\alpha = \arccos\left(\frac{T \cdot \begin{pmatrix} -1 & 0 & 0 \end{pmatrix}^{\mathrm{T}}}{||T||}\right) = \arccos\left(\frac{-T_x}{||T||}\right)$.

It follows that $\cos(\alpha) = -\frac{T_x}{||T||}$ and

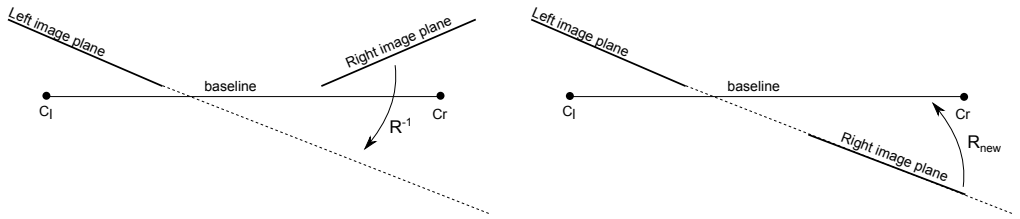
$$\sin(\alpha)z = \sqrt{1 - \frac{T_x^2}{||T||^2}}\ \frac{T_y}{\sqrt{T_y^2 + T_z^2}} = \frac{\sqrt{T_y^2 + T_z^2}}{||T||}\ \frac{T_y}{\sqrt{T_y^2 + T_z^2}} = \frac{T_y}{||T||}$$

and

$$\sin(\alpha)y = \sqrt{1 - \frac{T_x^2}{||T||^2}}\ \frac{-T_z}{\sqrt{T_y^2 + T_z^2}} = \frac{\sqrt{T_y^2 + T_z^2}}{||T||}\ \frac{-T_z}{\sqrt{T_y^2 + T_z^2}} = \frac{-T_z}{||T||}$$

Therefore the first row of the corresponding rotation matrix R is given by $\begin{pmatrix} -\frac{T_x}{||T||} & -\frac{T_y}{||T||} & -\frac{T_z}{||T||} \end{pmatrix}$ which is equal, up to the factor -1, to the first row of the rotation matrix using the first calculation approach, but this is not surprising, since the first row of R represents the orientation of the x-axis.

The last step of the rectificaton procedure is the calculation of the two translation vectors T_1 and T_2. The optical center C of one camera satisfies the equation $A(\ R \quad T\ )\begin{pmatrix} C \\ 1 \end{pmatrix} = (\ 0 \quad 0 \quad 0 \quad 0\ )^{\mathrm{T}}$. Rewriting this equation it follows that $T = -RC$. Because the optical center of the left camera is $(\ 0 \quad 0 \quad 0\ )^{\mathrm{T}}$, T_1 has the same coordinates. C_r is equal to T and therefore $T_2 = -R_{new}R^{-1}T$. The final new projection matrices are $A(\ R_{new} \quad 0\ )$ and $A(\ R_{new}R^{-1} \quad -R_{new}R^{-1}T\ )$. The two rectification steps are illustrated in Figure 6.



(a) Rotation of right image plane (b) Rotation of left and right image plane

Figure 6: Rectification process, (a) Rotation of the right image plane to be parallel to the left image plane, (b) Rotation of the image planes to be parallel to the baseline

Let us check, if the camera setup is parallel now. There are two facts to verify, the rotation matrix between the two camera coordinate systems has to be equal to the identity matrix and the translation vector should be parallel to the x-axis. If we look at the new projection matrices, the rotation matrix can easily be estimated. Because both coordinate systems are rotated according to the matrix R_{new}, the image planes are parallel now, thus the matrix, which rotates one coordinate system to the other, is the desired identity matrix. The new translation vector is $-R_{new}T$. It is easy to proof, that this vector is parallel to the x-axis because of the construction of R_{new}. The first row vector is $\frac{T^{\mathrm{T}}}{||T||}$ and both of the other rows are orthogonal to it. What follows is that $-R_{new}T$ is equal to $(\ \frac{-T^{\mathrm{T}}T}{||T||} \quad 0 \quad 0\ )^{\mathrm{T}}$.

Since the distortion parameters of the two cameras are usually different, it is recommended to undistort the rectified images according to the individual values. This improves the stereo matching results and further the 3D reconstruction.

4.4 Stereo Matching

To reconstruct a three-dimensional scene using stereo vision, point correspondences between the two different views are required. This is the goal of stereo matching. As

explained in the previous section, the corresponding point of one image point has to be part of the appropriate epipolar line in the other image. Therefore the correspondence search is reduced to a search along the epipolar line. Are the images in addition also rectified, the corresponding point only has to be searched along the same row in the other image. If the left image is the reference image, the corresponding pixel to a pixel (x, y) in the left image has to be searched in the right image and has the coordinates $(x - d, y)$. If the right image is the reference image, the corresponding pixel in the left image has the coordinates $(x + d, y)$. The associated value of d is defined as the *disparity* of (x, y). With this information a so-called *disparity map* or *disparity image* can be calculated, where the pixel at position (x, y) has the value of the appropriate disparity d. There are numerous stereo matching methods, but basically they can be classified into two categories: pixel-based and feature-based techniques.

Pixel-based methods compare intensity or color of pixels to find corresponding points. Mostly not only single pixels are considered, but small blocks around one pixel. That is the reason why these approaches are often called block-matching methods. The structure of one pixel block in the first image is searched in the other one. For the position (x, y) in the first image a block with size (m, n) around this point is compared with a block the same size around the position $(x - d, y)$ or $(x + d, y)$, depending which image is the reference image, in the second image, where d varies in a predefined searching range $[d_{min}, d_{max}]$. The point $(x - d, y)$ resp. $(x + d, y)$, where the block surrounding it is most similar to the reference block, according to some predefined similarity measure, is referred to as the corresponding one. So dense disparity maps are computed, which takes much time.

In feature-based approaches certain features are extracted in both images to find corresponding ones. Therefore both images have to be preprocessed to extract these features. This is an additional effort, but afterwards correspondence search is much easier and faster. The calculated disparity maps are usually more sparse than in pixel-based methods. Possible features could be edges, corners, lines, curves or others.

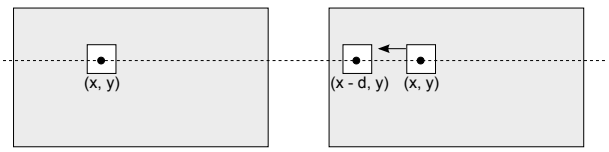


Figure 7: Stereo matching

Nevertheless some cases may arise, where it is very difficult or impossible to find the appropriate point. First of all there are areas, which cannot be viewed by both cameras or are hidden behind any object. These areas are called *occlusions*. Here, point correspondences are impossible to find as it is illustrated in Figure 8. Furthermore, the structure of the scene can cause problems. For example if there is a low textured area like a white wall, every point in the surrounding area could be a matching candidate. Another example is repetitive texture. The problem with such areas is, that multiple

correspondences exist and therefore wrong matches occur. Furthermore, photometric variation causes problems during the stereo matching procedure. This means, that if two cameras are viewing the scene from different angles, the captured light and intensity are different, hence correspondences are difficult to find.

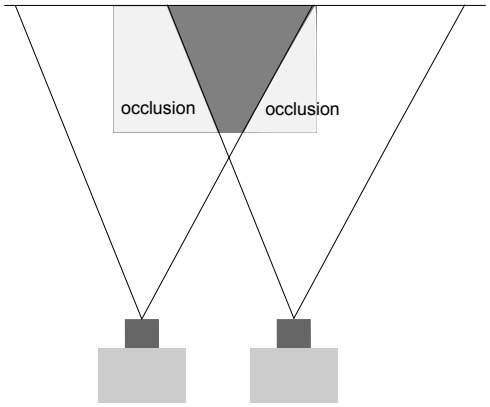


Figure 8: Occlusions that occur using a two-camera sensor

4.5 3-D-Reconstruction

After solving the correspondence problem, it is easy to reconstruct the three-dimensional scene. We already know, how a point $P = (\ X \quad Y \quad Z\ )^{\mathrm{T}}$ is projected onto the screen:

$$S \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = A(\ R \quad T\ ) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Assume that $f_x = f_y = f$ and that the stereo system is already rectified, hence $R = I$ and $T = (\ b \quad 0 \quad 0\ )^{\mathrm{T}}$, so the left and right projection matrices are simplified to

$$\begin{pmatrix} & & 0 \\ \mathrm{A} & & 0 \\ & & 0 \end{pmatrix} \text{ and } \begin{pmatrix} & & fb \\ \mathrm{A} & & 0 \\ & & 0 \end{pmatrix}.$$

Therefore the pixel coordinates of corresponding points in the left and right image are

$$\begin{pmatrix} \frac{fX}{Z} + u_0 \\ \frac{fY}{Z} + v_0 \end{pmatrix} \text{ and } \begin{pmatrix} \frac{fX}{Z} + u_0 + \frac{fb}{Z} \\ \frac{fY}{Z} + v_0 \end{pmatrix}.$$

Through triangulation it follows that $\frac{fb}{Z} = d$, as we have expected.

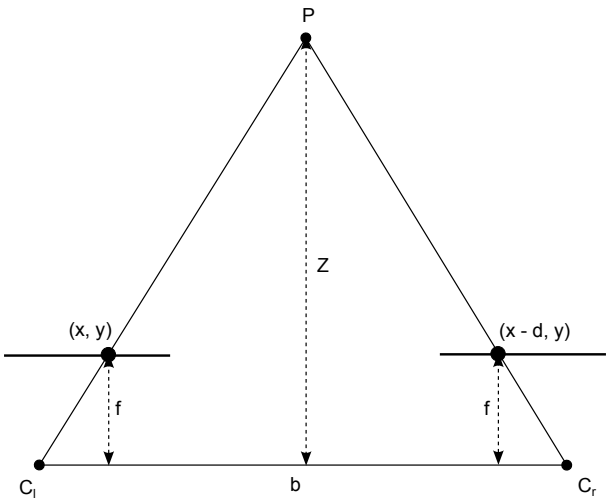


Figure 9: Triangulation

To reproject a point uniquely, corresponding points have to be found. The stereo matching procedure solves this problem. More precisely, it calculates the disparity of every pixel (x, y) in the first image, which means, that the corresponding point in the second image has the pixel coordinates $(x-d, y)$. The according depth Z of the 3D point can again be calculated using triangulation, $Z = \frac{fb}{d}$. Looking at the formulas for the pixel coordinate calculation, X and Y are easy to compute, $X = \frac{(x-u_0)Z}{f} = \frac{(x-u_0)b}{d}$ and $Y = \frac{(y-v_0)Z}{f} = \frac{(y-v_0)b}{d}$. This can be combined by using homogeneous coordinates:

$$\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{b} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix}$$

The final 3D coordinates are then $\begin{pmatrix} \frac{X}{W} & \frac{Y}{W} & \frac{Z}{W} \end{pmatrix}^{\mathrm{T}}$.

4.6 Summary

In this chapter it is presented how to reconstruct a three-dimensional scene using two cameras. First of all some details about a camera's geometry as well as its mathematical model are explained. Furthermore, two camera calibration methods are introduced, first the method for a single camera and second the method for a stereo camera system consisting of two cameras. The geometry which models such a stereo system is called epipolar geometry, which is also discussed in detail.

An ideal stereo camera setup is formed by two cameras mounted in parallel, therefore a process called rectification is performed when using systems where the cameras do not

form such a parallel setup. The main advantage is, that point correspondences now can be searched along image rows which is performed by a stereo matching method. When the so-called correspondence problem is solved, the disparity values for most of the pixels are available which enables the three-dimensional reconstruction through triangulation.

This chapter gives an overview of the fundamentals of stereo vision, further information can be found in [7] or [8].

5 Related Work

Stereo vision systems are basically divided into two categories, passive and active stereo vision. Passive approaches use two or more cameras to reconstruct the scene without actively controlling the scene's illumination. Textureless areas cause problems in the stereo matching process there. For this purpose active stereo vision systems, which are also known as *structured light systems*, are advised. Here, one camera is replaced by a projector, which illuminates the scene with a specially designed pattern. Therefore two camera images are no longer considered for the correspondence problem, but every point in the pattern is searched in the camera's image of this pattern. Depending on the used structured light technique the correspondence problem is solved differently. In some cases classical stereo matching methods, pixel-based or feature-based, are used. Here, only pixel-based approaches are considered.

5.1 Structured Light

Today numerous structured light techniques are available, which are all based on the same principle, projecting some sort of light, capturing it with a camera and searching correspondences. They are classified into multi shot or single shot categories. Multi shot techniques are limited to static scenes and occupy much time, but the results are often more accurate. Single shot techniques are preferred, when measuring moving targets. It would be impossible to present all state of the art techniques based on structured light in this section, instead some interesting methods are picked out, which give a good overview about projection opportunities and technical principles.

A very popular and inexpensive projector-camera system is Microsoft's Kinect, which is also based on structured light and will be depicted as well.

5.1.1 Multi Shot Technique

The simplest structured light method is a zero-dimensional approach, where a single light point is projected onto the scene. Usually a laser-light source is used due to its high focusing capability. The position of this dot in the camera image can be calculated very accurately, but lots of shots have to be taken to measure the whole scene, because scanning along both axes is required. Another possibility is to project a single slit, which is a one-dimensional method. For this purpose also laser emitters are used, although the numbers of shots are reduced, because the scanning procedure only has to be executed along one axis. A last solution uses plan-based patterns, hence it is a two-dimensional approach. In general, whole areas can be scanned and so this method is much faster than the previous ones, but is not as accurate as them. Another aspect is, that also a correspondence search is necessary to reconstruct the scene. Therefore most bi-dimensional pattern projectors are based on coded structured light, which means that a sequence of patterns is projected such that every illuminated point gets its own unique codeword and can be easily identified in the camera image. The three multi shot approaches are illustrated in Figure 10.

One coded structured light approach is proposed by Posdamer and Altschuler [9]. They used binary patterns to produce the unique codewords. These patterns consist of white and black stripes and during every shot one pixel gets either the value 0 or the value

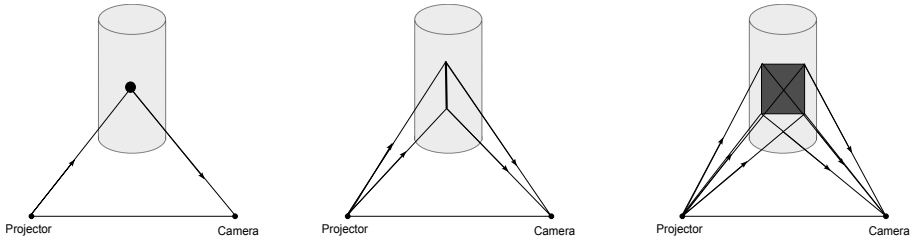


Figure 10: Zero-, one- and two-dimensional projection

1, depending if it is covered by a white or a black stripe. After n sequences every pixel possesses a codeword of n bits. Figure 11 shows a pattern sequence with 5 stages and the corresponding 5-bit codewords. This technique is very robust to surface structures, because only binary values are used, but it needs $log_2(n)$ different patterns to uniquely encode n positions.

To reduce the number of patterns, gray codes are provided as in [10]. The scanning procedure is exactly the same as for binary patterns with the only difference, that the patterns now contain more than two intensities. If m gray intensities are used and n positions have to be coded, only $log_m(n)$ patterns are required.

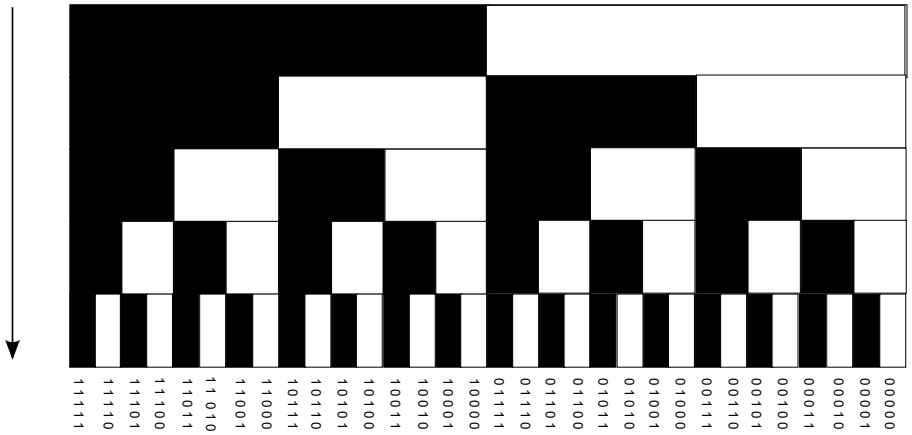


Figure 11: Binary pattern sequence and associated codewords

Another different and very popular multi shot technique is called phase shift. It is often used, because of its speed and accuracy. The scene is illuminated by some sinusoidal pattern with a constant phase-shift angle θ. Take for example a three step phase-shifting with $\theta = \frac{2\pi}{3}$, as it is proposed in [11]. The intensity values, I_1, I_2 and I_3 in every pattern, make it possible to compute the phase angle of every pixel. They can be calculated as

$$I_1(x,y) = I_0 + I_{mod}\cos(\phi(x,y) - \frac{2\pi}{3})$$

$$I_2(x,y) = I_0 + I_{mod}\cos(\phi(x,y))$$

$$I_3(x,y) = I_0 + I_{mod}\cos(\phi(x,y) + \frac{2\pi}{3})$$

where I_0 is the background's intensity, I_{mod} is the modulation of the intensity by the object and ϕ is the phase to be solved for. To extract ϕ, three trigonometric identities are used:

$$\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta)$$
$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$$
$$\tan(\frac{\alpha}{2}) = \frac{1 - \cos(\alpha)}{\sin(\alpha)}$$

Then we get $\frac{I_1 - I_3}{2I_2 - I_1 - I_3} = \frac{\tan(\phi)}{\tan(\frac{\pi}{3})} = \frac{\tan(\phi)}{\sqrt{3}}$ and finally

$$\phi = \arctan(\sqrt{3}\frac{I_1 - I_3}{2I_2 - I_1 - I_3})$$

The values of ϕ are within the range of 0 and 2π, with a discontinuity at 2π. In order to obtain a continuous phase function Φ, which identifies the exact position of one pixel, multiples of 2π have to be added, dependent on the period(stripe) where the pixel is located, $\Phi(x, y) = \phi(x, y) + 2\pi k(x, y)$. So if there are n projected periods, $k(x, y)$ is within the range of 0 and n. The process, which solves this problem, is called phase unwrapping. Usually phase unwrapping can only be performed precisely, if it is combined with other methods. One approach is to combine phase shifting with gray code patterns, which requires additional projections, as it is proposed by Bergmann [12]. Another possibility, which is introduced in [13], is to use a second camera to perform stereo matching between the two cameras in order to solve for $k(x, y)$.

Once the phase Φ for a pixel at location (x, y) is determined, the depth Z of the corresponding point in space can be calculated by means of a reference plane at known distance Z_{ref}, where its phase has the value $\Phi_0(x, y)$. This is shown in Figure 12.

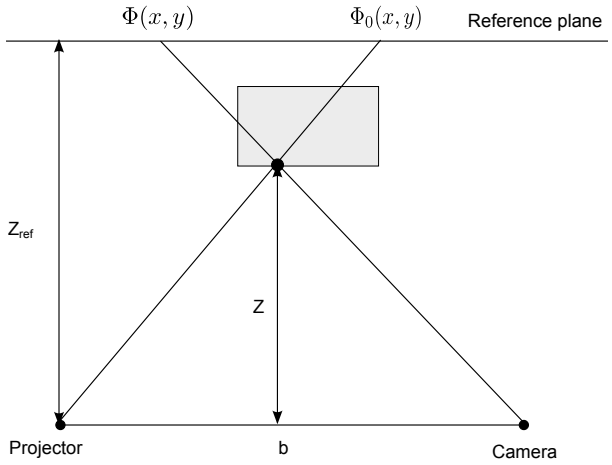


Figure 12: Triangulation using phase shift

$$\frac{Z_{ref} - Z}{\Phi(x, y) - \Phi_0(x, y)} = \frac{Z}{b} \Rightarrow Z \approx \frac{Z_{ref} b}{\Phi(x, y) - \Phi_0(x, y)}$$

5.1.2 Single Shot Technique

Single shot techniques have big advantages in dynamic scenes, because only one picture has to be taken for reconstruction. There are lots of patterns, that can be used, which are all based on different coding strategies. The first method that will be described here, uses multiple slits with random cuts, as it is introduced by Maruyama and Abe [1]. Due to the cuts, each line is separated into multiple slits. A sector of the pattern is shown in Figure 13. One slit is identified by its length and the six adjacent slits. The correspondence search starts with matching segments of equal length. Because this usually results in many matches, the six nearby slits of the observed one have to be considered. The problem of this technique is, that it only provides good results for measuring smooth and continuous surfaces because otherwise the length of the segments can vary due to strong deformations of the pattern.

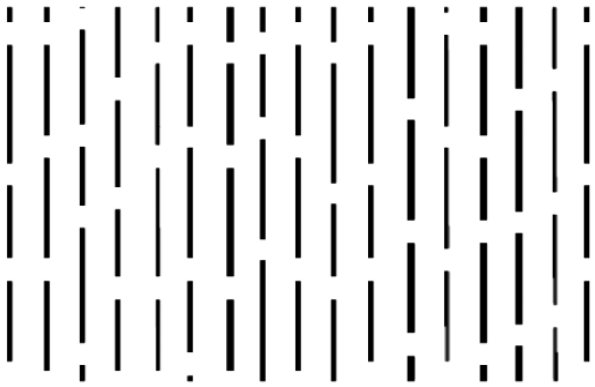


Figure 13: Pattern proposed by Maruyama and Abe [1]

Another approach, that uses coded strips, is proposed by Boyer and Kak [14]. The pattern is composed by some unique subpatterns, which are designed by vertical colored strips (red, green and blue), separated by black bands. To match each slit in the camera image with the projected slit, a four-step stripe indexing algorithm is performed to identify each stripe by its unique index. The four main components are called *correlation, crystal-growing, crystal-fitting* and *ordinal assignment*. During the first step each subpattern of the projected pattern is compared with the camera's image of it and possible matching candidates are listed. This list may contain more than one or no entries. The following steps have to decide, which match is the correct one. The crystal-growing process tries to find as many correspondences of slits within the subpatterns, which are referenced to be possible matches. Therefore a so-called crystal grows with every found correspondence. Afterwards some slits are indexed redundantly, some are indexed incorrectly and some are not indexed at all. The goal of the crystal-fitting stage is to eliminate false matches and keep the correct ones. For this purpose the decision criterion is the size of the crystals, if two crystals overlap, the larger one is accepted. The last component, ordinal assignment, simply takes the results from the previous stages to index all the

matched stripes.

Vuylsteke and Oosterlinck [2] presented a method, which uses a column coded binary pattern. The structure of the pattern is a checkerboard, where each grid point is marked with a single codebit. A part of the pattern is shown in Figure 14.

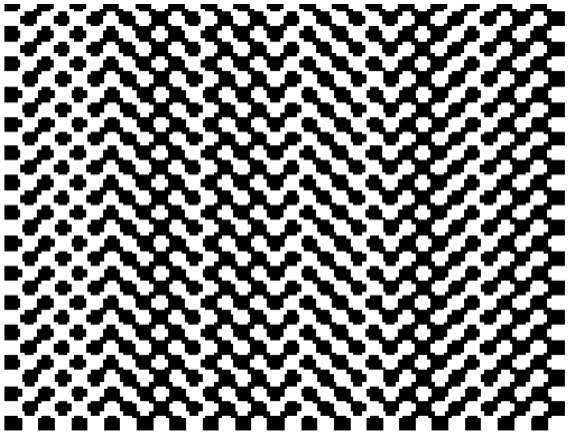


Figure 14: A detail of the pattern design by Vuylsteke and Oosterlinck [2]

The observed grid points are those points, where four squares of the chessboard meet and each of them is either marked with a bright or a dark spot, representing the binary values 1 and 0. There are two possible configurations, how the surrounding squares are painted, Vuylsteke and Oosterlinck denoted them as + or −, as Figure 15 shows. The whole pattern consists of 63 coded columns. The underlying code is based on two binary sequences, c_k and b_k, of 63 bits length. c_k represents the grid points that are marked as $0+$ and $1+$ and b_k those that are labeled $0-$ and $1-$. Since there are only two binary sequences, the assignment of every second row is repeated, but it can be proofed that c_k and b_k can be chosen, in a way that every 2×3 window carries its unique 6-bit codeword.

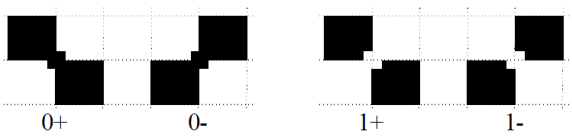


Figure 15: Grid point labeling [2]

Le Moigne and Waxman [15] proposed to use a grid pattern with several included dots. There are some parameters, which have to be chosen to design the accurate pattern, for example the thickness of the lines, the number of dots and the location of the dots. Especially the line thickness has to be chosen dependent on the smoothness of the scene, very thin lines show lots of discontinuities, which complicates the matching procedure

whereas too thick lines display not enough range variation at all. To identify each imaged point of the pattern in the projected one, several steps are performed. First the vertical lines are extracted, because Le Moigne and Waxman decided to use a vertical baseline, which implies, that vertical lines are approximately parallel and hardly deformed, which makes them easy to detect. Afterwards a so-called *albedo normalization* is performed along every vertical line. This means, that the gray level distribution is adjusted to reduce albedo and highlight effects. Then the intersections of the horizontal lines and dots with the vertical lines are searched by scanning the vertical lines. Once they are found, they can be labeled by combining and interpolating two initial labelings, wherefore the positions of the dots are used. Le Moigne and Waxman indicated, that their proposed method works very well in dynamic scenes, since it was developed for the navigation of a mobile robot.

A very interesting one shot technique was given by Morano et al. [3]. Their method is based on a pseudorandom coded pattern. The associated code is generated by a square pseudorandom matrix M, in which every 3×3 submatrix appears exactly once. The elements of M are letters of an alphabet A and the position of every element m_{ij} is determined by the letters of itself and its neighboring letters, which is stored in a nine-element vector and composes a 3×3 submatrix. Since all of them are unique within M, the Hamming distance, which is the number of positions at which the corresponding elements differ, of two such vectors is always larger than one. When generating the pseudorandom matrix there are three criteria, which have to be fixed, the size of M, the size of A and the minimal Hamming distance, each pair of vectors has to exceed. Different elements of A appear as different colors, different shape or different texture. The code generation starts with a 3×3 matrix, which is located at the top left corner of M. It is filled with random letters of A. The generation continues by considering three positions each time and also labeling them with random values. During every iteration the Hamming distances are checked and if some does not exceed the predefined minimum, other letters have to be chosen. This procedure keeps going, as long as the whole matrix M is filled. If at any iteration no valid letter is left, the generation is discarded and the process starts at the beginning. Figure 16 illustrates the first code generation steps.

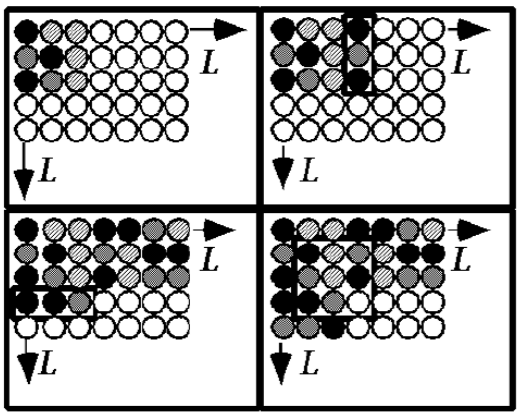


Figure 16: Code generation [3]

Once the pattern is designed, the correspondence search can be performed. For each dot within the pattern its eight neighboring dots are viewed, all of them labeled with a letter of A. A nine-letter codeword is generated, which indicates the position of the observed dot in the pattern. If this position is found within the projection, the neighborhood is compared and a correspondence value, depending on the number of found correspondences, is assigned to the dot. As each dot is contained in nine different windows, there are nine possible correspondence values and the codeword that provides the highest value labels the dot.

5.1.3 Microsoft Kinect

Microsoft's Kinect is a special structured light sensor. Originally it was developed as a gaming gadget, but due to its high accuracy in depth calculation and since it is an inexpensive 3D sensor compared to other ones, the Kinect has become a wide-spread device in computer vision and robotic areas. The sensor consists of three elements, an infrared projector, an infrared camera and a color camera, whereby only the infrared devices are used for 3D reconstruction. Since the Kinect projects infrared light, it is recommended to reduce ambient light, which implies, that the depth calculation only works well indoor. In [16] it is explained, that the pattern is not directly projected, but contained on a transparency, which is illuminated by optical rays. More precisely, two diffractive optical elements are used, as it is explained in [4]. The first one applies the pattern to the beam and the second splits the beam. The pattern stays constant during depth calculation, so it is based on the one-shot technique. What is very interesting about this structured light sensor, is the special design of the projected pattern. Lots of researchers tried to uncover it, which turned out to be not as simple as it seemed. For example [17] shows a picture of the speckled pattern, which can be seen in Figure 17. It is recognizable, that it is basically rectangular but deformed because of a pin-cushion distortion.

What researchers have been discovered so far, is that the pattern is constructed of a 3×3 repetition of a 211×165 subpattern and that it is $180°$- rotation invariant. In fact this is not important for depth calculation but allows to mount the transparency, which includes the pattern, upside down. Within every subpattern there are some bright and some dark spots and the number of them varies. A detailed discussion can be found in [18]. Another interesting fact is, that within all of the nine subpatterns a bright spot, nearly at the center of the field, can be detected. According to [19], this spot does not have an affect on the depth calculation, it is simple a side effect of the diffraction, the so-called zero-order beam. There are certain guesses how the Kinect actually calculates depth. In [18] it is assumed, that there are small regions that are unique and can therefore be used to identify the location within the pattern that is projected. Another approach is assumed in [20]. It is guessed that disparity is computed through area-based stereo matching.

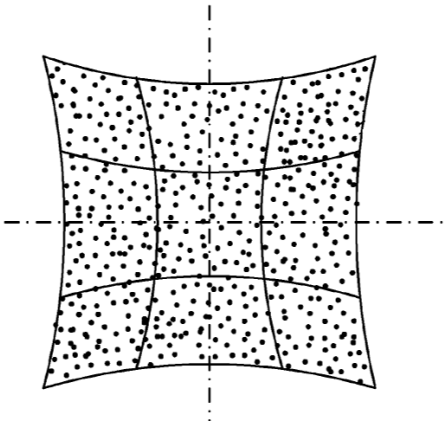


Figure 17: Kinect's pattern deformed by pin-cushion distortion [4]

5.2 Stereo Matching methods

Apart from multi shot structured light techniques or coded structured light methods some active stereo vision approaches use pixel-based methods to calculate correspondences. As already explained not single pixels are compared, but small blocks surrounding them.

The initially introduced disparity calculation methods have the following in common. A block in the first image I_1 is compared to a block in the second image I_2, according to some similarity measure. During every iteration the block in the second image is moved in x-direction in steps of one pixel according to a particular searching range. The shift, which minimizes the matching costs, is the desired disparity.

The first common used measure is the *Sum of Absolute Differences (SAD)*. The costs of a block B in the images I_1 and I_2, surrounding (x, y), are calculated as

$$\sum_{(i,j)\in B} |I_1(x+i, y+j) - I_2(x+i+d, y+j)|$$

As can be seen, the absolute difference between the intensities of the neighborhood of (x, y) are considered. The computational effort is low, because only additions and subtractions are used. The size of the block B is important for the accuracy of the disparity calculation, because a greater neighborhood is considered, which ensures, that the correct corresponding block is found. On the other hand larger blocks imply a higher computational effort as well as difficulties in measuring small structures and edges, so a suitable compromise has to be chosen.

Another similarity measure is the *Sum of Squared Differences (SSD)*, where the costs are computed according to

$$\sum_{(i,j)\in B} (I_1(x+i, y+j) - I_2(x+i+d, y+j))^2$$

Here, the effort is much higher, because of the additional multiplications. Furthermore, this measure is more sensitive to outliers, because large errors are higher weighted.

The matching costs can also be calculated according to the *Normalized Cross Correlation (NCC)*:

$$\frac{\sum_{(i,j)\in B} I_1(x+i,y+j)\cdot I_2(x+i+d,y+j)}{\sqrt{\sum_{(i,j)\in B} I_1^2(x+i,y+j)\cdot\sum_{(i,j)\in B} I_2^2(x+i+d,y+j)}}$$

This measure is independent of the intensities of the compared blocks and only calculates the relative difference between them.

All of these similarity measures do not take into account, that the two images I_1 and I_2 are recorded or produced by two different devices. Therefore the mean brightness can differ. To overcome this problem, the mean brightness in every image, $\overline{I_1(x+i,y+j)} = \frac{1}{|B|^2}\sum_{(i,j)\in B} I_1(x+i,y+j)$ and $\overline{I_2(x+i+d,y+j)} = \frac{1}{|B|^2}\sum_{(i,j)\in B} I_2(x+i+d,y+j)$, is subtracted. The resulting measures are called *Zero-mean Sum of Absolute Differences (ZSAD), Zero-mean Sum of Squared Differences (ZSSD)* and *Zero-mean Normalized Cross Correlation (ZNCC).*

$$ZSAD: \sum_{(i,j)\in B} |I_1(x+i,y+j) - \overline{I_1(x+i,y+j)} - I_2(x+i+d,y+j) + \overline{I_2(x+i+d,y+j)}|$$

$$ZSSD: \sum_{(i,j)\in B} (I_1(x+i,y+j) - \overline{I_1(x+i,y+j)} - I_2(x+i+d,y+j) + \overline{I_2(x+i+d,y+j)})^2$$

$$ZNCC: \frac{\sum_{(i,j)\in B}(I_1(x+i,y+j)-\overline{I_1(x+i,y+j)})\cdot(I_2(x+i+d,y+j)-\overline{I_2(x+i+d,y+j)})}{\sqrt{\sum_{(i,j)\in B}(I_1^2(x+i,y+j)-\overline{I_1^2(x+i,y+j)})\cdot\sum_{(i,j)\in B}(I_2^2(x+i+d,y+j)-\overline{I_2^2(x+i+d,y+j)})}}$$

The last matching cost calculation method, that will be mentioned here, is called *Sum of Hamming Distances (SHD).* Here, the cost formula for a block B is

$$\sum_{(i,j)\in B} h(I_1(x+i,y+j), I_2(x+i+d,y+j))$$

where h is the function which computes the Hamming distances. The Hamming distance of two bitstrings is the number of different bits between the two strings. Usually this measure is employed for census-transformed images, which will be introduced next.

The previous methods all compare the intensity values of the matching blocks. Another possibility is to compare the relative positions of the intensities within these blocks. Therefore both images are transformed, according to a special transformation instruction first, and then applied to one of the already introduced similarity measures. Two important transformation methods are called *census* and *rank* [21].

The census $C((x,y))$ assigns every pixel (x,y) a string of zeros and ones, a bit-string. Every pixel's intensity within a block surrounding a pixel (x,y), let us call it census block, is compared to the intensity of (x,y). If it is less, one is attached to the bit-string, otherwise zero is attached. The related formula for $C_k((x,y)), k \in \{1,2\}$, and a block B is

$$C_k((x,y)) = \bigotimes_{(i,j)\in B} \xi_k((x,y),(i,j))$$

where ξ_k is defined as

$$\xi_k((x,y),(i,j)) = \begin{cases} 1 & I_k(x,y) > I_k(i,j) \\ 0 & I_k(x,y) \leq I_k(i,j) \end{cases}$$

and $\bigotimes$ is the bit concatenation. An example of a census block and the resulting bit-string is shown in Figure 18. Once both images are census transformed, the matching costs are computed using SHD, whereby $I_k(x,y) = C_k((x,y))$.

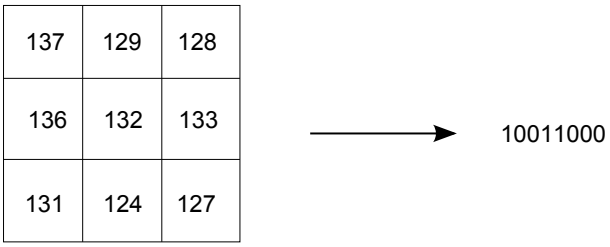


Figure 18: Matching block with intensities and resulting bit-string

The rank $R((x,y))$ works as follows. For every pixel (x,y) in one image and a $n \times m$ block surrounding it, the number of pixels within that block are counted, whose intensities are smaller than the intensity of (x,y). This will be a number between 0 and $nm-1$. Here, the related formula for $R_k((x,y)), k \in \{1,2\}$, is

$$R_k((x,y)) = |\{I_k((i,j)) : I_k(i,j) < I_k(x,y), (i,j) \in B\}|.$$

An example is illustrated in Figure 19. To calculate the disparity of the transformed images, one of the previous introduced similarity measures can be used.

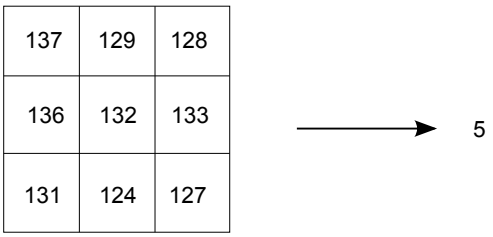


Figure 19: Matching block with intensities and resulting number

5.3 Summary

In this chapter various structured light methods are introduced. Basically they are divided into multi and single shot techniques. There are three multi shot approaches, zero-, one- and two-dimensional projection, whereby all of them provide accurate results but are not

recommended for dynamic scenes. Therefore single shot techniques are proven to provide better results in such scenes. In this chapter a few single shot approaches as well as a special structured light sensor, Microsoft's Kinect, are presented.

Furthermore, some stereo matching methods, more precise pixel-based stereo matching methods, are explained. First, some approaches are introduced which compare intensity values, for example SAD of SSD, and second some approaches which compare the relative positions of intensity values, for example census.

6 Projector-Camera System

A projector-camera system contains, as the name implies, of some sort of a light projector and a camera and forms a stereoscopic or structured light system. It is a different approach to passive stereo vision, where usually two cameras are viewing the scene. If classical stereo matching methods are used to solve the correspondence problem, it is necessary to calibrate the setup first in order to reconstruct the scene successfully and to simplify the stereo matching procedure. If the system is calibrated, the camera's image of the projected pattern and the pattern itself are rectified and correspondences can be searched along the same image rows.

Projector-camera systems have a big advantage in comparison to classical two-camera stereo vision sensors: Measurement is independent of the surface's structure. As already explained, conventional stereo vision systems have problems in measuring textureless or repetitive areas, because it is difficult to find the correct point correspondences. Since the projector illuminates the scene with a random pattern, it creates the necessary texture and irregularity and so projector-camera systems do not have any difficulties in measuring such areas. On the other hand these sensors are not able to calculate depth, if the projected pattern is not visible by the camera. This case may arise for example if the lighting conditions are bad. To overcome this, a combination of both sensors, a two-camera system and a projector-camera system, can be used. In this section it will be described, which calculations are necessary to combine them and furthermore, the advantages are discussed.

6.1 Projector-camera Calibration

Projector-camera calibration is an important task when working with structured light systems, which use area- or feature-based stereo matching procedures. It makes rectification possible and therefore simplifies the correspondence search. Theoretically a projector can be modeled as a pinhole camera, only the direction of projection differs. This means, that the projector is not able to capture the scene it is illuminating, which implies that it is impossible to compute the three-dimensional coordinates of one point in the projected pattern on its own. Hence an additional device, such as a camera, has to be used to calibrate the projector. What has to be calculated are the intrinsic parameters of the projector as well as the rotation and translation between the camera's and the projector's image plane. Therefore, it is some sort of a stereo calibration procedure, with the only difference that only one camera is capturing.

There exist different methods to solve this problem. Some of them require a pre-calibrated camera in order to calibrate the projector and their geometrical relationship, while others calculate this simultaneously. Here, the first method will be introduced. It is similar to the procedure proposed by G. Falcao et al. [22].

First, a pre-calibrated camera is required. It is very important, that the camera is calculated properly, because the accuracy of the projector calibration is dependent on the accuracy of the camera calibration. Furthermore, a planar projection surface, where the projector's calibration pattern is projected, is needed. This plane has to be identified by the camera. Therefore some markers are fixed on this surface, which can be uniquely detected in the camera's image.

Because the camera is already calibrated, 3D coordinates of every 2D point in the camera's image can be calculated, what is important for this calibration method, since it makes it possible to calculate the 3D coordinates of the markers on the calibration plane. To get rid of the camera's distortion, the camera's images are undistorted, before further calculations are performed. During the calibration procedure the plane is viewed from different angles, as it is done in camera calibration. The main task in calibrating the projector is to calculate the 2D-3D correspondences of the calibration points of the projected pattern in order to solve for the desired parameters. The whole work-flow of this calibration method is illustrated in Figure 20.

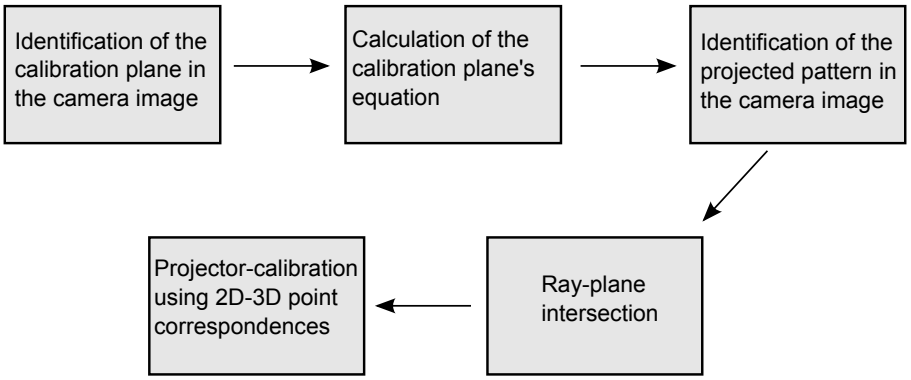


Figure 20: Projector-calibration work-flow

The first step is to calculate the equation of the projection surface in the camera's coordinate system. Therefore the fixed markers are used. These markers could be the outer corners of a chessboard or something else, where special points can be detected reliably. It is recommended that the outermost markers form a rectangle, because they determine the world coordinate system(the axes will be rectangular in a Cartesian coordinate system. The detailed calculation is described in the following.

The x- and y-axis of the world coordinate system are defined by the fixed pattern, hence the calibration plane is equal to the x-y-plane in the world coordinate system. Since the z-axis is orthogonal to both axes, it is equal to the normal vector of the calibration surface. So once the calibration pattern is identified in the camera' s image, it is possible to compute the rotation and translation between the camera coordinate system and the world coordinate system, because the camera is already calibrated. Here, the reference coordinate system is set at the camera's coordinate system. The corresponding extrinsic matrix is $\begin{pmatrix} r_1 & r_2 & r_3 & T \\ 0 & 0 & 0 & 1 \end{pmatrix}$, where r_1, r_2 and r_3 are the three columns of the rotation matrix and T is the translation vector. In order to obtain the equation of the calibration plane, a point P on the plane and its normal vector $\overrightarrow{n}$ have to be known. The equation is given by

$$\overrightarrow{n}\begin{pmatrix} x & y & z \end{pmatrix}^{\mathrm{T}} = \overrightarrow{n}P$$

Since the camera coordinate origin is translated to the world coordinate origin at the calibration plane according to the translation T, a point of the plane is already known,

T. The normal vector is equal to the z-axis of the world coordinate system, as already explained. Because the three vectors of the rotation matrix represent the orientations of the three axes, the normal vector is equal to r_3. Altogether we get the following equation for the calibration plane:

$$r_3(\ x \quad y \quad z\ )^{\mathrm{T}} = r_3 T \ \Rightarrow\ r_{31}x + r_{32}y + r_{33}z = r_{31}T_1 + r_{32}T_2 + r_{33}T_3$$

The next step is the detection of the pattern, that is projected by the projector and the calculation of the corresponding 3D coordinates in the camera coordinate system. Therefore the camera's image of the pattern is used as well as the previously calculated equation of the projection plane. First some points within the pattern have to be identified in the camera's image, let (x, y) be the coordinates of one such point. Because the camera is calibrated, the intrinsic matrix A and the rotation R and translation T are known, hence the coordinates $(\ X \quad Y \quad Z\ )^{\mathrm{T}}$ of the corresponding 3D point can be calculated up to a scale factor s:

$$s \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = (\ AR \quad AT\ )^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

This means, that the corresponding 3D point can be localized everywhere along the ray that goes through the origin of the camera coordinate system and the 2D point in the camera image. To calculate the coordinates of the 3D point uniquely, the equation of the projection plane is used. What has to be computed more exactly, is the intersection with the ray, that goes through the camera coordinate origin and the detected point in the camera image. This intersection is the projected point on the calibration surface.

$$s(r_{31}X + r_{32}Y + r_{33}Z) = r_{31}T_1 + r_{32}T_2 + r_{33}T_3$$

$$\Rightarrow s = \frac{r_{31}T_1 + r_{32}T_2 + r_{33}T_3}{r_{31}X + r_{32}Y + r_{33}Z}$$

In this way it is possible to obtain the 3D coordinates for all points within the projected pattern. The whole calibration setup is illustrated in Figure 21.

The last step in projector-camera calibration is to use the 2D-3D correspondences of some special points within the pattern to perform the main calibration procedure, as it is done in camera calibration. Therefore the 2D coordinates of the special calibration points have to be identified within the pattern first. These coordinates are part of the projector image plane. After the previous steps, the corresponding 3D coordinates in the camera coordinate system are available. For every different view of the plane different 3D coordinates are obtained. The input for the final calibration procedure are two matrices, the first contains all the calculated 3D points of the calibration points of all different views, while the second is filled with the corresponding 2D coordinates, whereby for every different view the same coordinates are used, because they always stay the same within the projected pattern. This procedure outputs the intrinsic and extrinsic parameters of the projector. Since the reference coordinate system is fixed at the camera and the

geometrical relationship between projector and camera always stays the same, only one rotation and one translation is calculated, which is in fact the rotation and translation between the camera coordinate system and the projector coordinate system.

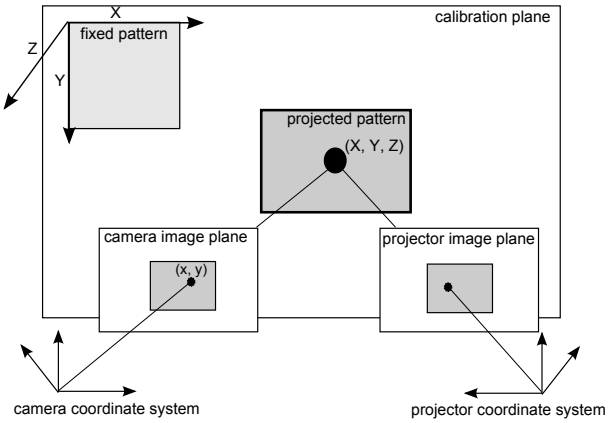


Figure 21: Projector calibration setup

Let us compare this step with the same procedure in camera calibration. In projector calibration the used 3D coordinates are calculated within the camera coordinate system. Therefore they differ when the viewing angle changes. The corresponding 2D coordinates are calculated within the projector image plane and always stay the same. What has to be calculated is the rotation and translation between the camera coordinate system and the projector coordinate system, which stays the same as long as the geometrical orientation between camera and projector stays the same. In camera calibration the coordinates of the used 3D points are computed belonging to the world coordinate system, which is identified by a calibration object. Since this object always stays the same, independent of the viewing angle, the 3D coordinates also stay the same for every different view. In this procedure the 2D points are part of the camera image plane, hence they differ, when the viewing angle changes. Here, the rotation and translation between the world coordinate system and the camera coordinate system has to be computed. This implies that the rotation and translation is different for every view.

The proposed projector-camera calibration procedure assumes that the camera, where the reference coordinate system is defined, is on the left side of the projector. Therefore the projection matrices have the form $A_c(\ I \quad 0\ )$ and $A_p(\ R \quad T\ )$

If the setup is reverted, hence the projector is on the left side of the camera and the reference coordinate system is fixed at the projector, the matrices have to be modified to $A_p(\ I \quad 0\ )$ and $A_c(\ R^{-1} \quad -R^{-1}T\ )$

6.2 Rectification and Undistortion

If the projector-camera system is calibrated, the intrinsic parameters of camera and projector are known as well as the rotation and translation between their image planes. Since the last step of this calibration method is equal to the camera calibration, the distortion parameters of the projector's lens are also available. Altogether the same information is

obtained, that is available after stereo calibration of two cameras, which is not surprising, because the projector-camera calibration is some sort of a stereo calibration procedure. Now it is possible to rectify the two image planes. Since the camera is viewing the projected pattern, it can be found anywhere on the camera's image plane. The projector's image plane is usually covered by the whole pattern. After the rectification process the images are always undistorted in order to get rid of the different distortions, that deform the images. When working with projector-camera systems, one of the two processing images is the camera's image, while the other one is the projected pattern. The rectified camera image is undistorted as usual. Since the pattern is not deformed anyway, one might think, it does not have to be undistorted. But what modifies the image, is the distortion of the projector, which deforms the pattern during the projection process. So this is another difference between the model of a camera and a projector. A camera captures an undistorted image and due to its lens distortion a distorted one is obtained. The distortion model of the projector influences in the other direction, an undistorted image is projected, but a distorted one is put out. So in order to consider this fact, the undistorted and rectified pattern has to be distorted, which works the same way as the process of distortion correction. Altogether an undistorted and rectified camera image of the pattern, as well as a distorted and rectified pattern are available now and can be used for further calculations. The rectification matrices are computed as it is explained in the second chapter. In Figure 22 the rectification and distortion/undistortion procedure are shown.

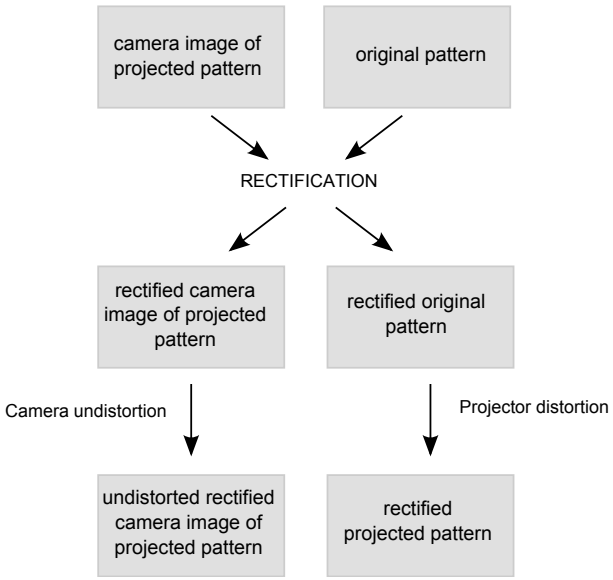


Figure 22: Distortion and undistortion procedure

6.3 Disparity Calculation

After the rectification procedure the rows of the pattern are aligned to the corresponding rows in the camera's image. Now the stereo matching process can be performed in order

to calculate the disparity image. Which method is used depends for example on the scenes that will be reconstructed or on the pattern that is projected. Here, census combined with SHD will be explained in more detail, especially some modifications and improvements will be introduced.

Stereo matching using census first performs a transformation of the input images, where every pixel obtains a bit-string, whose size depends on the size of the concerned neighborhood block. An increase in this size allows to gather more information, since a wider neighborhood is considered. This may improve the quality of the matching procedure, but results in higher computational effort and memory occupation. One possible compromise could be the so-called *sparse census*. It is called sparse, because not every neighboring pixel within the census block is used for the calculation of the bit-string.

This allows to increase the size of the census block, but keep the computational effort the same. Let us choose for example a census block size of 8. Using the standard census, 64 pixels have to be compared with the reference pixel, which results in a bit-string of size 64. The sparse census transformation, where every second row and column within the census block is considered, needs only 16 compares, as it is illustrated in Figure 23.

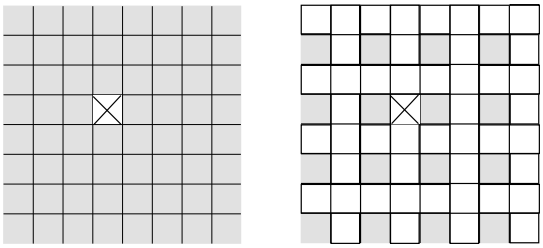


Figure 23: Whole census block and sparse census block

One way to improve the correctness of the disparity values for every pixel, is to calculate subpixel disparities. By now all the occurring values are integers, since the matching block moves along an epipolar line in steps of pixels. But cases may arise, where the real disparity of one pixel lies between two integer values. The calculation of the disparity is done after the images are census transformed, using SHD. More precisely, the minimum of the matching cost function is searched and exactly this can cause some errors, if only integers are considered. Therefore a parabola, $y = Ax^2 + Bx + c$, is calculated, that goes through the minimum integer disparity, d_{min} and its two neighboring disparity values, $d_{prev} = d_{min} - 1$ and $d_{next} = d_{min} + 1$. This parabola is an approximation of the matching cost function $SHD(d)$ in the interval $[d_{prev}, d_{next}]$. What we are looking for, is the minimum of this parabola, which is the root of its first derivate, $y' = 2Ax + B$.

$$\Rightarrow x_{min} = \frac{-B}{2A}$$

The coefficients, A and B, of the parabola are calculated as follows.

$$SHD(d_{prev}) = Ad_{prev}^2 + Bd_{prev} + C$$

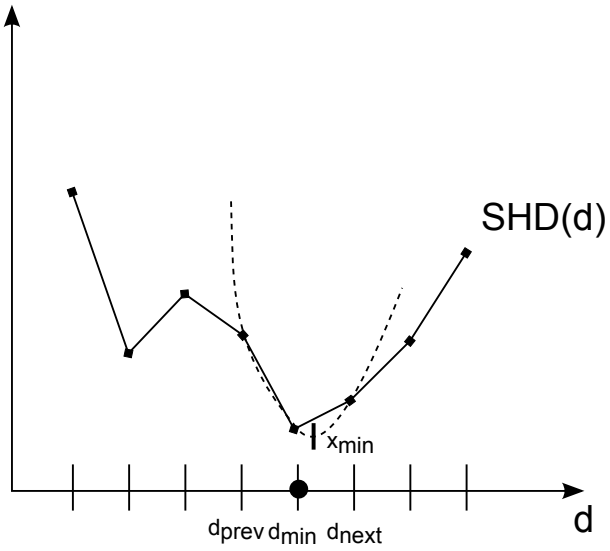


Figure 24: Subpixel disparity calculation

$$SHD(d_{min}) = Ad_{min}^2 + Bd_{min} + C$$

$$SHD(d_{next}) = Ad_{next}^2 + Bd_{next} + C$$

$$\Rightarrow SHD(d_{prev}) - SHD(d_{min}) = A(d_{prev}^2 - d_{min}^2) + B(d_{prev} - d_{min}) = A(d_{prev}^2 - d_{min}^2) - B$$

$$\Rightarrow SHD(d_{min}) - SHD(d_{next}) = A(d_{min}^2 - d_{next}^2) + B(d_{min} - d_{next}) = A(d_{min}^2 - d_{next}^2) - B$$

Solving this system of equations we get

$$A = \frac{SHD(d_{prev}) - 2SHD(d_{min}) + SHD(d_{next})}{2}$$

and

$$B = \frac{-SHD(d_{prev})(1 + 2d_{min}) + 4SHD(d_{min})d_{min} + SHD(d_{Next})(1 - 2d_{min})}{2}$$

and further

$$x_{min} = d_{min} + \frac{SHD(d_{prev}) - SHD(d_{next})}{2(SHD(d_{prev}) - 2SHD(d_{min}) + SHD(d_{next}))}$$

As can be seen, the subpixel disparity x_{min} does not have exactly the same value as the minimal integer disparity d_{min} and therefore calculating subpixel disparities increases the accuracy and results in much smoother disparity maps.

Another way to check and improve the correctness of the calculated disparity values, is called *Left-Right-Consistency Check*. Therefore two disparity images have to be calculated, first the left image is the reference image and corresponding pixels are searched in

the right image and second the right image is the reference image and correspondences are searched in the left image. If (x, y) is one pixel in the left image, the corresponding pixel in the right image is $(x - d_1, y)$, where d_1 is the calculated disparity, if the left image is the reference image. For the pixel $(x - d_1, y)$ in the right image an own disparity value d_2 is calculated, if the right image is the reference image. The corresponding pixel in the left image is $(x - d_1 + d_2, y)$ and hopefully this pixel is equal to (x, y), hence $d_2 - d_1 = 0$. The *Left-Right-Consistency Check* accepts disparity values, if this difference is less than a given threshold δ, otherwise the disparity value is invalid. Formally this can be written as:

$$d = \begin{cases} \frac{|d_1 + d_2|}{2} & \text{if } |d_1 - d_2| < \delta \\ 0 & \text{else} \end{cases}$$

6.4 A multiple baseline sensor

Depth calculation using two-camera stereo vision sensors often suffers from different factors. First of all occlusions, which are areas, that are not visible in both camera images, lead to imperfect disparity and depth images. Second, the structure of the scene is a major factor, that causes problems. Measuring textureless areas is difficult, because point correspondences are not obvious as in the case of repetitive areas.

Projector-camera systems would be a good alternative, since the projection of an irregular and random pattern produces the necessary structure and irregularity. But these sensors also have some measuring problems. Point correspondences between the pattern and the camera's image of it are impossible to find, if the pattern is not visible by the camera. Such a case may happen, if the lighting conditions are bad, for example if the ambient light is too bright to identify the projected pattern in the camera's image. Another example is a dark scene, where it is difficult to recognize the structure and shape of the pattern. Projector-camera systems are also dependent on the projector's depth of field. Hence, if the projector has a shallow depth of field, scenes with large depth ranges can be measured only partly.

One possible solution, that eliminates both disadvantages, is a combination of a stereo camera pair and a projector-camera system. The combined setup might look like it is shown in Figure 25, two cameras and in the middle a pattern projector. Both sensors have to be precalibrated accurately on their own, hence two independent stereo systems are available to measure the scene.

The problems that occur during reconstruction using two-camera systems with measuring textureless and repetitive areas are eliminated due to the pattern projection. Furthermore, this combined sensor is less sensitive to bad lighting conditions than a projector-camera system, in which under these circumstances the pattern would not be visible by the camera. Depth can be reconstructed anyway, due to the two-camera head, which does not have problems with these scenes.

Since there are two calibrated stereo sensors available, two depth images of the same scene can be calculated. Both of them are not completely dense and suffer from individual matching difficulties. A major problem is the reconstruction of areas, that are not visible by both cameras when using the two-camera sensor or areas that aren't illuminated by the projected pattern or visible by the associated camera when using projector-camera

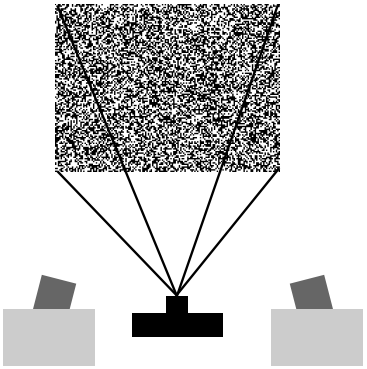


Figure 25: Stereo cameras combined with pattern projector

systems, the so-called occlusions. These areas are impossible to reconstruct. Since there are two different stereo systems, the occlusions are also different and occur on variable areas, hence the ones that appear in only one depth image can be reduced.

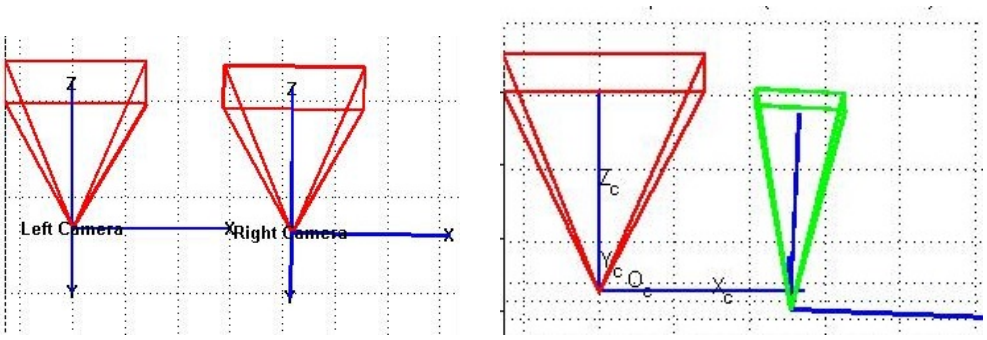
Another aspect, that should be considered, is the availability of two different baselines, a longer baseline, in the setup shown in Figure 25 this is the distance between the two cameras, and a shorter baseline, the distance between the left camera and the projector. A shorter baseline could result in imprecise depth reconstruction for large depth ranges, due to the narrow triangulation, while a long baseline requires a larger disparity search range, which leads to higher computational effort. So if two baselines of various length are available, the advantages of both can be deployed. For example the possible uncertain results of the disparity calculation using the shorter baseline can be used to constrain the disparity search range for the computation using the longer one.

But which steps are necessary to calculate this combined depth image of the two depth images, that are available from the two stereo systems?

Let's recap briefly how a depth image is calculated. If a stereo sensor is calibrated, the two images, that are used for reconstruction, are rectified, i.e., corresponding lines are aligned on the same row and parallel to the x-axis. The needed rotation matrices for the left and for the right image, R_1 and R_2 are composed by two rotation matrices, $R_1 = R_{new}$ and $R_2 = R_{new}R^{-1}$, where R represents the rotation matrix between the two image planes and R_{new} rotates both of them to be parallel to the x-axis. Afterwards disparities can be calculated and then the depth values are computed, using the formula $Z = \frac{fb}{d}$.

These depth values are calculated with respect to the left camera's coordinate system, so one might think, that the two depth images can be compared directly, but this is not true. The reason for this is, that the two stereo systems are oriented differently, since the rectification matrices turn the image planes parallel to the individual x-axis, or in other words parallel to the baseline. Therefore the left camera's coordinate system is also orientated differently and the depth values are computed with respect to two diverse coordinate systems. The two differently orientated stereo systems are illustrated in Figure 26.

To solve this problem, one of the coordinate systems has to be transformed to the other



(a) Two-camera (b) Projector-camera

Figure 26: Stereo setups: (a) Two-camera setup and (b) Projector-camera setup

coordinate system or the calculated 3D points have to be rotated. Here, the reference system is set to the two-camera sensor, hence the found 3D points of the projector-camera system have to be transformed. Before calculating depth, the left camera's coordinate systems have been rotated according to the rectification matrices R_1^c and R_1^p. Therefore the calculated points of the projector-camera system have to be multiplied with the inverse of R_1^p and further with R_1^c to get the coordinates with respect to the left camera's coordinate system of the two-camera system. This is shown in Figure 27.

Next the related depth image can be calculated by projecting the 3D coordinates on the screen. The appropriate projection matrix is $\begin{pmatrix} A & 0 \end{pmatrix}$, where A is the new intrinsic matrix of the left camera after rectification. The 3D point $\begin{pmatrix} X & Y & Z \end{pmatrix}^{\mathrm{T}}$ has the pixel coordinates $\begin{pmatrix} \frac{f \cdot X}{Z} + c_x & \frac{f \cdot Y}{Z} + c_y \end{pmatrix}^{\mathrm{T}}$, hence the value of the depth image at this position is Z.

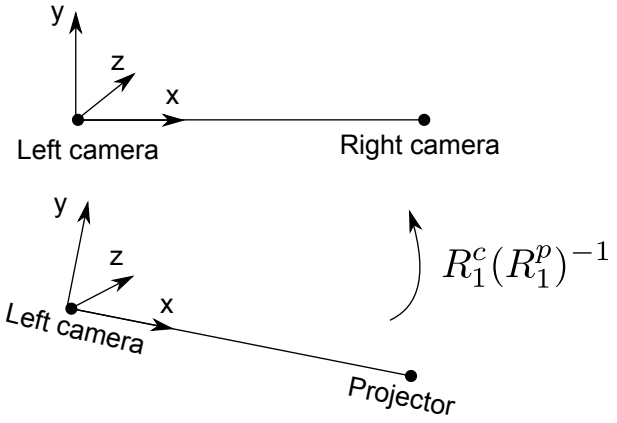


Figure 27: Coordinate system transformation

If all these modifications are performed, every pixel in the first depth image corresponds to exact the same pixel in the second modified depth image and now they can be combined. The whole procedure is illustrated in Figure 28.

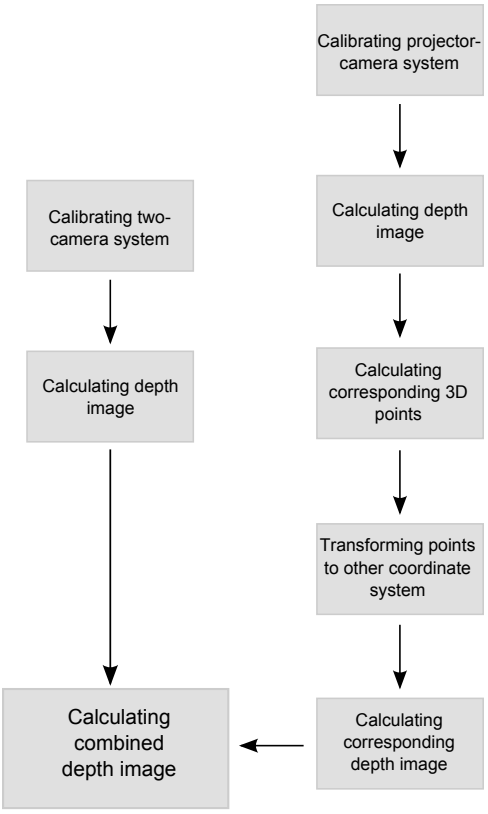


Figure 28: Depth map combination procedure

A possible approach for the combination is to check each pixel in the individual depth images. If the depth value at that position is not available in both images, the combined depth map also stays empty there. If one value is available, it is directly inserted at that position and if both depth images are filled, the location in the combined depth map is filled with the mean value.

6.5 Summary

In this chapter a projector-camera system is presented. It is a special stereo vision system, where one of the two cameras is replaced by a pattern projector. The main advantage is, that textureless areas, which are difficult to measure with two-camera systems, can also be reconstructed, since it is covered by the projected pattern, which produces the necessary structure. To measure depth, the projector-camera system have to be calibrated, hence a projector-camera calibration method, which is a special stereo calibration method, is introduced here. Correspondences are no longer searched between two camera images but the projected pattern is compared with the camera's image of it.

Furthermore, a particular stereo matching method, census, is discussed in detail and some improvements, for example sparse census or subpixel disparity calculation, are introduced. Last, a multi baseline sensor consisting of two cameras and a projector is presented. This sensor includes two stereo systems, the two-camera system and the projector-camera system, which enables the availability of two different depth maps.

These maps can then be combined in order to reduce measuring problems of both sensors.

7 Experimental Setup and Implementation

In this section the experimental setup consisting of the hardware and software we developed during this work, is described. Furthermore, all the existing software, for example the Camera Calibration Toolbox for MATLAB, the Camera Projector Calibration Toolbox or the Stereo Vision Engine S3E, is presented. Next the implementations and modifications, that are developed during this work are explained and described in detail. This includes the modifications of the Camera Projector Calibration Toolbox in MATLAB, the stereo matching algorithms in MATLAB and the modifications of the S3E software concerning the projector-camera integration and depth map combination. Last an automatic projector calibration is presented, that needs a precalibrated stereo camera system and is implemented in MATLAB with additional information, which is received from the S3E software.

7.1 Hardware

The cameras used during this work are all devices from iDS [23], in detail USB 2 uEye cameras, with a resolution of 752×480 or 640×480. All of them are monochrome imagers and are connected by USB 2 cables with the processing PC. The used lenses have a focal length of 3.6 mm.

The projector-camera calibration and further algorithms were tested with different projectors. The first one was a standard video and presentation projector from BenQ, BenQ MP624. This projector has a resolution of 1024×768, but since the resolution of the projected pattern and the combined cameras has to be the same, only a part of the projection field is used. Working with this projector provides accurate results, since it is not very sensitiv to bad lighting conditions, because of its high luminous intensity of 3000 lumen. But what is not ideal about this projector in combination with an iDS camera, is the vertical offset between the projector's lens and the camera's lens. First of all, this results in rectified images which are much more tilted than it is shown in Figure 29. The reason is the tilted baseline, since the rectification procedure rotates the image planes parallel to the baseline. The second effect of the vertical offset is that not the whole projected pattern is visible by the camera, which can also be seen in Figure 29.

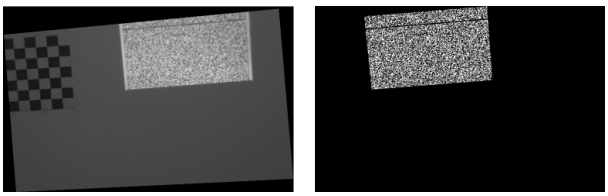


Figure 29: Tilted rectified images

Therefore another projector was also used, Philips Pico Pix, with a resolution of 854×480, where according to the combined camera only 752×480 or 640×480 are used. This beamer has much less luminance, 55 lumen, which makes it necessary to work with less background luminance. But the main advantage is the practical size and the

compatibility with the iDS cameras to a nearly perfect stereo sensor system, with barely any vertical offset. The resulting rectified images are shown in Figure 30. It can be seen that they are not tilted as much as the ones from the previous projector.

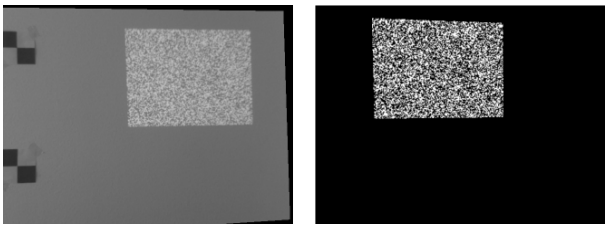


Figure 30: Less tilted rectified images

The used setup, consisting of an iDS camera and the Philips Pico Pix projector, is illustrated in Figure 31.

Figure 31: Projector-camera setup

7.2 Implementation

7.2.1 Camera Calibration in MATLAB

To calibrate a single camera accurately some calibration software is needed to calculate the intrinsic parameters, i.e. the focal length, the principle point and the distortion parameters. In this work, the Camera Calibration Toolbox for MATLAB from J. Y. Bouguet, [24], is used. This software includes a graphical user interface, which makes the whole procedure simple and clear. It is illustrated in Figure 32.

What is required to calibrate a camera using this toolbox? First of all a planar chessboard with known chessboard square size is needed. This is fixed somewhere at a planar surface and several pictures with different viewing angles are captured with the camera to be calibrated. Afterwards the chessboard corners have to be found in every image. Therefore the user has to identify the four outer corners, while the software calculates the other ones. This outer corners also define the origin and axes of the world coordinate system. If all the 2D coordinates of the corners in every image are available

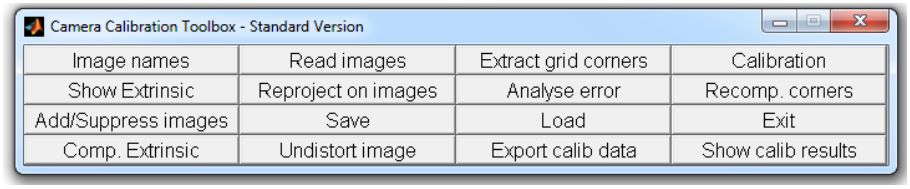


Figure 32: Camera Calibration Toolbox from Bouguet

the parameters are calculated as it is described in section 4.1. Since the viewing angle is different for every image, the extrinsic parameters, i.e., the rotation and translation between camera and world coordinate system, also vary.

Additionally there are some refinement functions included to post process the calibration results. For example it is possible to remove some images, so that they are not included in the calibration calculations. Furthermore, the chessboard corners for single images can be recomputed, if the results for some images are not satisfying.

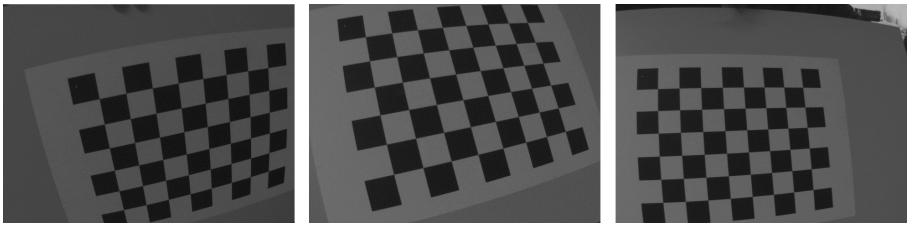


Figure 33: Calibration images

A possible output of the software might be:

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 718.86203   721.45018 ]
Principal point:       cc = [ 305.53215   251.41131 ]
Skew:             alpha_c = [ 0.00000 ]  => angle of pixel axes = 90.00000 degrees
Distortion:            kc = [ -0.43351   0.13072   0.00263   -0.00148  0.37500 ]
Pixel error:          err = [ 0.06274   0.05722 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

There exists also an extension of this toolbox to calibrate a stereo camera system, also with an included graphical user interface, which is shown in Figure 34.

For this calibration procedure both cameras have to capture the calibration chessboard simultaneously. These images are used to calibrate both cameras on their own. Since the world coordinate system is defined by the planar chessboard and the cameras are capturing at the same time, the extrinsic parameters for both cameras refer to the same

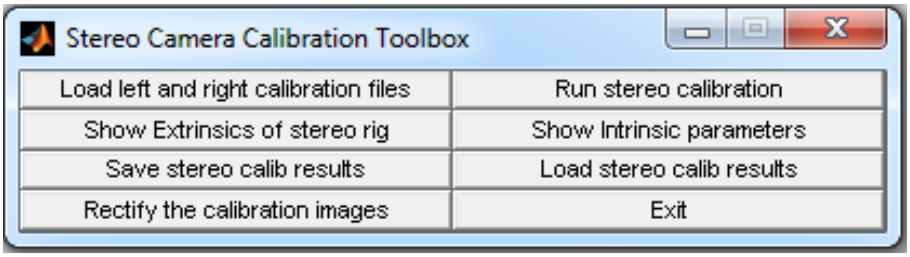


Figure 34: Stereo Calibration Toolbox

world coordinate system. Therefore the rotation R and the translation T between them can be calculated, as it is also explained in section 4.1. The software does not directly output the rotation matrix R, but the corresponding rotation vector om, which represents the rotation axis and its norm represents the rotation angle. The matrix R can be calculated as follows:

$$om = \begin{pmatrix} x & y & z \end{pmatrix} \text{ and } \alpha = ||om||$$

$$\Rightarrow R = \begin{pmatrix} (1-\cos(\alpha))\frac{x^2}{\alpha^2} + \cos(\alpha) & (1-\cos(\alpha))\frac{xy}{\alpha^2} - \sin(\alpha)\frac{z}{\alpha} & (1-\cos(\alpha))\frac{xz}{\alpha^2} + \sin(\alpha)\frac{y}{\alpha} \\ (1-\cos(\alpha))\frac{xy}{\alpha^2} + \sin(\alpha)\frac{z}{\alpha} & (1-\cos(\alpha))\frac{y^2}{\alpha^2} + \cos(\alpha) & (1-\cos(\alpha))\frac{yz}{\alpha^2} - \sin(\alpha)\frac{x}{\alpha} \\ (1-\cos(\alpha))\frac{xz}{\alpha^2} - \sin(\alpha)\frac{y}{\alpha} & (1-\cos(\alpha))\frac{yz}{\alpha^2} + \sin(\alpha)\frac{x}{\alpha} & (1-\cos(\alpha))\frac{z^2}{\alpha^2} + \cos(\alpha) \end{pmatrix}$$

Here, a possible output might look like:

```
Stereo calibration parameters:


Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 718.86203   721.45018 ]
Principal point:       cc_left = [ 305.53215   251.41131 ]
Skew:             alpha_c_left = [ 0.00000 ]  => angle of pixel axes = 90.00000
Distortion:            kc_left = [ -0.43351   0.13072   0.00263   -0.00148  0.37500 ]


Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 721.32566   723.84952 ]
Principal point:       cc_right = [ 312.85926   254.12161 ]
Skew:             alpha_c_right = [ 0.00000 ]   => angle of pixel axes = 90.00000
Distortion:            kc_right = [ -0.44225   0.25683   0.00210   0.00022  -0.10160 ]


Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:             om = [ -0.00323   -0.00553  -0.01247 ]
Translation vector:           T = [ -208.84569   1.84183  3.77565 ]
```

```
Note: The numerical errors are approximately three times the standard deviations (for reference).
```

7.2.2 Projector-camera Calibration in MATLAB

For the projector-camera calibration the basic functions as well as the graphical user interface of the toolbox from G. Falcao et al. [22], are used, while some of the functions are modified during this work. The GUI is shown in Figure 35.

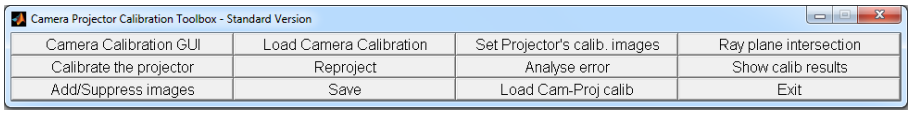


Figure 35: Camera Projector Toolbox

As it is described in section 6.1, a precalibrated camera is needed for this calibration method. It is recommended, that the camera calibration is done using Bouguet's toolbox, since the parameters are named and needed exactly as they are outputted from this toolbox. Also a planar calibration surface with some fixed markers is necessary. The original software requires a fixed chessboard as well as a projected chessboard in order to calibrate the projector. The camera should capture some pictures with different viewing angles, whereon the fixed chessboard as well as the projected pattern should be visible.

First of all the parameters of the camera calibration have to be loaded. By pressing the button *Load Camera Calibration*, the MATLAB function `loading_calib` is called, which loads all the necessary parameters into the workspace. Next the button *Set Projector's calib images* has to be pressed, which executes the function `define_Projector_Images`. This function prompts the user for the names of the projector calibration images, that should show the fixed chessboard on the calibration plane as well as the projected one. An example of such a calibration image is shown in Figure 36.

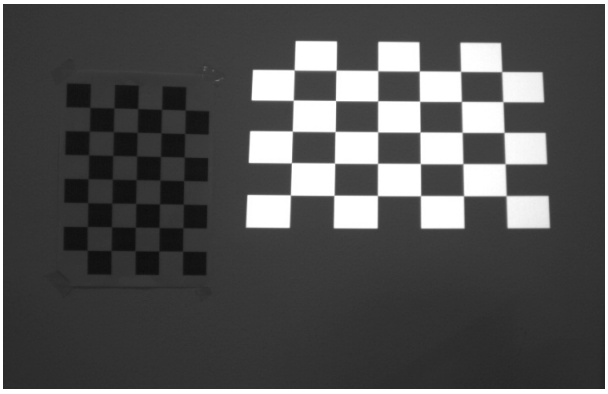


Figure 36: Fixed and projected calibration chessboard

The user has to identify the outer corners of the fixed chessboard in the camera's image, just as in camera calibration, while the software finds the other corners. Then the

equation of the calibration plane can be calculated, as it is described in chapter four. By pressing the button *Ray plane intersection* the function `cam_proj_3d_points` is called. Again the user is asked for identifying the outer chessboard corners in the camera's image, but now the ones of the projected pattern, while the others are also found automatically. The software calculates the 3D coordinates of all these points by calling the function `obtain3Dpoints_from_knownPlane`, which uses the equation of the calibration plane for the calculations. Last the outer corners of the original pattern has to be indicated in order to obtain the corresponding 2D coordinates. Then the button *Calibrate the projector* has to be pressed to call the function `calibrate_projector`, which calculates the intrinsic parameters of the projector as well as the rotation and translation between the camera's and the projector's image plane and outputs the final calibration results, for example:

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 1634.04251   1641.13557 ]
Principal point:       cc = [ 312.93471   442.69443 ]
Skew:             alpha_c = [ 0.00000 ]   => angle of pixel axes = 90.00000
Distortion:            kc = [ -0.03112   0.22608   -0.00007   -0.00381  0.00000 ]
Pixel error:          err = [ 1.04880   1.00317 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

The modifications, that were implemented concern the functions `define_Projector_Images` and `extract_grid`. They accelerate the whole calibration procedure and enable the calibration of the projector without the projection of a calibration chessboard, i.e., calibration is possible with every pattern, especially with random patterns, that are used here. The main advantage is, that every projector can be calibrated, even if its projected pattern is not changeable. The calibration method implemented here, is adopted to the projection of rectangular patterns, but is easily expandable to other forms. It is important, however, that somewhere within the pattern are some clearly identifiable points. Further it is not necessary to fix a whole chessboard to identify the calibration plane. Only four markers, that are arranged in a rectangle with known distances between them, are required. Another fact that has changed for the calibration procedure is, that the calibration images have to be undistorted first, before they are used for the projector calibration. This results in more precise projector parameters, since the distortion of the camera is eliminated. An example of such a calibration image is shown in Figure 37.

The function `define_Projector_Images` is modified in such a way, that every calibration image is shown only once. The user has to identify the four markers as well as the four corners of the projected pattern. For the identification of the corners of the pattern, the modified function `extract_grid` is called. Since the pattern does not include any chessboard corners any more, the other calibration points are just interpolated between them, depending on the number of points in x and y direction. This only works for projectors with hardly any distortion, because the distortion would influence the position of the interpolated points within the projected pattern. Otherwise some additional points have to be identified. Figure 38 shows the interpolated points within the pattern for an undistorted projector.

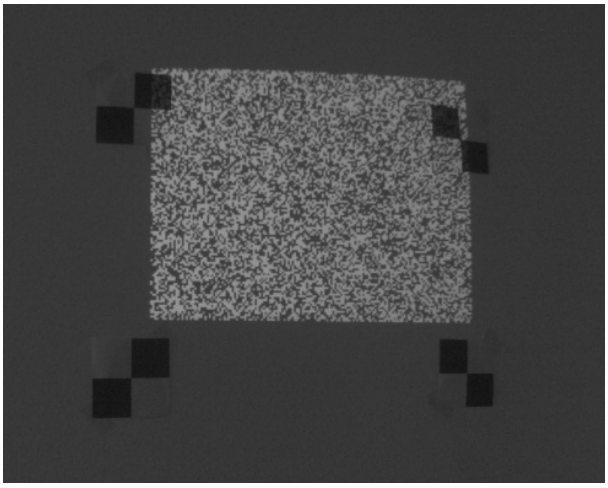


Figure 37: Fixed markers on calibration plane

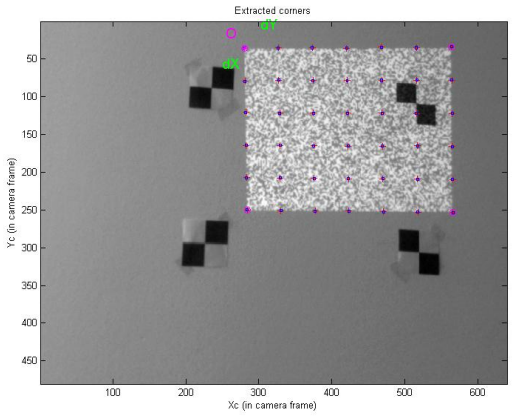


Figure 38: Interpolated calibration points

The 2D coordinates of the corresponding points within the original pattern does not have to be identified, because they could be calculated, since the four outer points are $(0,0),(w,0),(w,h)$ and $(h,0)$, where w is the width of the pattern and h the height. The inner points are interpolated equidistant between them. The remaining calibration processes and functions stay the same.

7.2.3 Rectification and Stereo Matching in MATLAB

Once the projector-camera system is calibrated using the software described in the previous section, the rectification procedure can be performed. Therefore the MATLAB function `rectify_stereo_pair` from the Camera Calibration Toolbox from Bouguet was used. The required input parameters comprise the intrinsic parameters of camera and projector, the rotation and translation vector between their image planes and some special informations about the images to be rectified.

First of all, the rotation matrices $R_1 = R_{new}$ and $R_2 = R_{new}R^{-1}$, that turn the image

planes to be parallel, are calculated. R is the rotation matrix between the two image planes before rectification, hence it is part of the input parameters. To calculate R_{new}, the rotation matrix, that aligns the image rows parallel, the unit normal vector n and the rotation angle α are needed. As it is described in section 4.3, $n = \begin{pmatrix} 0 & \frac{-T_z}{\sqrt{T_y^2+T_z^2}} & \frac{T_y}{\sqrt{T_y^2+T_z^2}} \end{pmatrix}^{\mathrm{T}}$ and $\alpha = \arccos\left(\frac{-T_x}{||T||}\right)$. The corresponding rotation matrix is given by

$$R_{new} = \frac{1}{||T||} \begin{pmatrix} -T_x & -T_y & -T_z \\ T_y & \frac{||T||T_z^2 - T_x T_y^2}{(T_y^2+T_z^2)} & -\frac{(||T||+T_x)T_y T_z}{(T_y^2+T_z^2)} \\ T_z & -\frac{(||T||+T_x)T_y T_z}{(T_y^2+T_z^2)} & \frac{||T||T_y^2 - T_x T_z^2}{(T_y^2+T_z^2)} \end{pmatrix}$$

Next the new projection matrices A_1 and A_2 are calculated. Therefore the new focal lengths and the new principle points have to be computed. Since the two image planes are parallel after rectification, the focal lengths and the principle points of both devices should be the same. For simplicity it is assumed, that pixels are square now, which implies $f_x = f_y = f$. The original intrinsic matrices are

$$A_l = \begin{pmatrix} f_x^l & 0 & u_0^l \\ 0 & f_y^l & v_0^l \\ 0 & 0 & 1 \end{pmatrix} \text{ and } A_r = \begin{pmatrix} f_x^r & 0 & u_0^r \\ 0 & f_y^r & v_0^r \\ 0 & 0 & 1 \end{pmatrix}.$$

The initial guess for f is f_y^l. Then it is distinguished, if the first of the three radial distortion parameters, k_1, is less or greater than 0. In the first case, f is multiplied with $1 + k_1^l \cdot \frac{W^2+H^2}{4 \cdot f^2}$, where W is the width of the images, H the height and k_1^l the first radial distortion parameter of the left camera. In the second case f remains the same.

The second guess for f is f_y^r and again the two cases are considered. The final value for f is the minimum of the two calculated focal lengths.

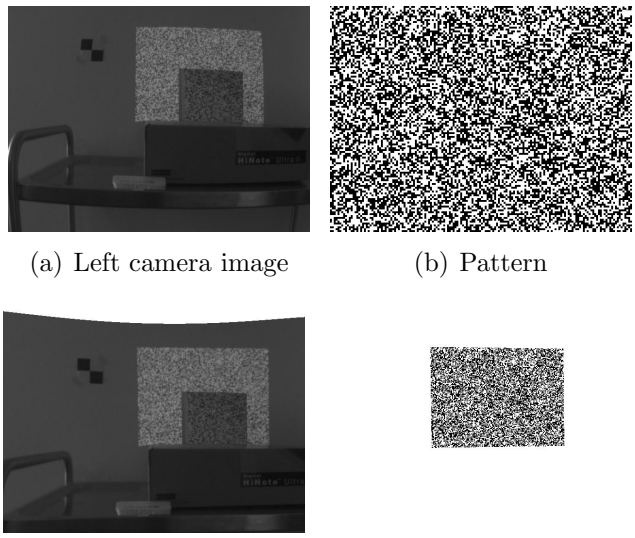
Next the principle point (c_x, c_y) is computed as follows. The four corners of the screen in pixel coordinates, $(0,0), (H,0), (H,W)$ and $(0,W)$, are transformed on the left image plane as well as on the right. Further they are rotated according to the previous calculated rotation matrices R_1 and R_2. Then the mean value of this calculated coordinates is subtracted from the center of the screen, $\frac{H+W}{2}$. The results are two principle points, one for the left image plane c_{new}^l, and one for the right, c_{new}^r, which maximize the visible areas in the rectified images. Since we want equal principle points for both new intrinsic matrices, the final principle point is the mean value $\frac{c_{new}^l + c_{new}^r}{2}$. Now all the rectification parameters are available to warp the left input images as well as the pattern, which is projected.

In Figure 39 the original camera image and pattern are shown as well as the results from the rectification process. The old intrinsic matrices are

$$A_l = \begin{pmatrix} 718.862 & 0 & 305.5321 \\ 0 & 721.4502 & 251.4113 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } A_r = \begin{pmatrix} 1634.492 & 0 & 304.6246 \\ 0 & 1635.0373 & 454.9134 \\ 0 & 0 & 1 \end{pmatrix},$$

and the new ones are

$$A_1 = A_2 = \begin{pmatrix} 711.8361 & 0 & 322.0776 \\ 0 & 711.8361 & 301.4767 \\ 0 & 0 & 1 \end{pmatrix}$$



(a) Left camera image (b) Pattern

(c) Rectified left camera image (d) Rectified pattern

Figure 39: Stereo images before and after rectification, (a) and (b) Original images, (c) and (d) Rectified images

Next the stereo matching procedure can be performed on the rectified images. Here, the census transformation in combination with SHD is used. It is modified as explained in section 6.3, the sparse transformation is implemented and subpixel disparities are calculated. The MATLAB functions `censusSparseL2R` and `censusSparseR2L` take as input parameters the two images to be matched, the census block size, the size of the matching window and the minimum and maximum disparity, which define the disparity search range $[d_{min}, d_{max}]$. The difference of the two functions is the reference image. In `censusSparseL2R` it is the left image and correspondences are searched in the right image, while in `censusSparseR2L` it is reversed.

First of all, the function `censusTransformSparse` is called for both images. It performs the sparse census transformation and outputs an image the same size as the input image, but filled with the census bitstrings, whose size is dependent on the census block size.

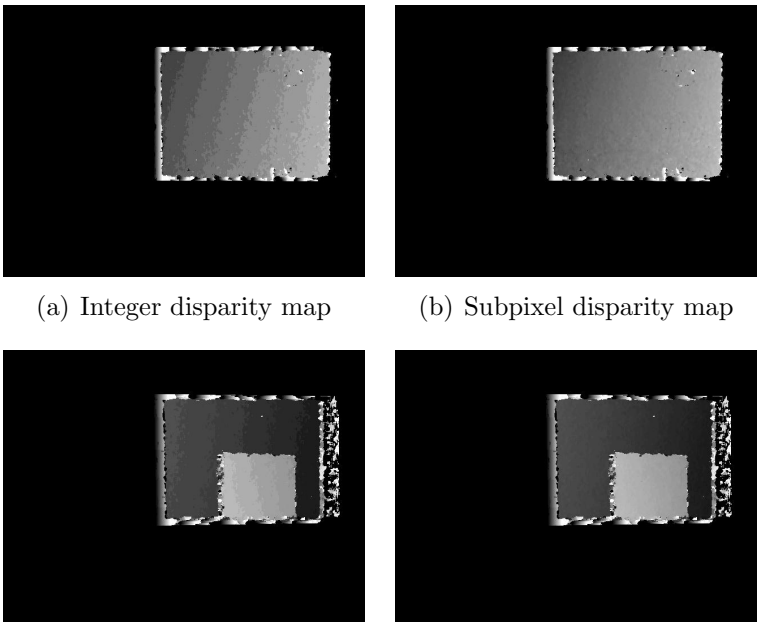
Next, the matching cost function $SHD(d)$ for every pixel and $d \in [d_{min}, d_{max}]$ is calculated. Therefore a pixel block surrounding a pixel (x, y) in the left image, whose size is defined by the matching window size, is compared with a pixel block the same size surrounding the pixel $(x - d, y)$ in the other image, if the function `censusSparseL2R` is called. The value of d, that minimizes SHD is the integer disparity. If the function `censusSparseR2L` is executed, the pixel block surrounding (x, y) in the right image is

compared with the one surrounding the pixel $(x+d, y)$ in the left image. This calculated integer disparity d is then added with

$$\frac{SHD(d-1) - SHD(d+1)}{2(SHD(d-1) - 2SHD(d) + SHD(d+1))}$$

to get the subpixel disparity.

In Figure 40 the integer and subpixel disparity maps can be compared. As it can be seen, the subpixel calculations result in much smoother images.



(a) Integer disparity map (b) Subpixel disparity map

(c) Integer disparity map (d) Subpixel disparity map

Figure 40: Comparison of integer and subpixel disparities, (a) and (c) Integer disparity maps, (b) and (d) Subpixel disparity maps

If both functions, `censusSparseL2R` and `censusSparseR2L`, are called, two disparity maps are available, hence the *Left-Right-Consistency Check* can be performed. Here, a threshold of 0.5 is chosen and the results are shown in Figure 41. The pixels colored black indicate a mismatch, which means that the subpixel disparity values differ more than 0.5. As can be seen, edges and occlusions cause most of the problems.

7.2.4 S3E Software

The S3E Stereo Vision Engine is a software module, developed for calculating depth using stereo vision algorithms in real time. It is devided into three stages, a two or three camera system, which captures the scene and is connected to the processing PC, the stereo matching software, which is based on census in combination with the Hamming

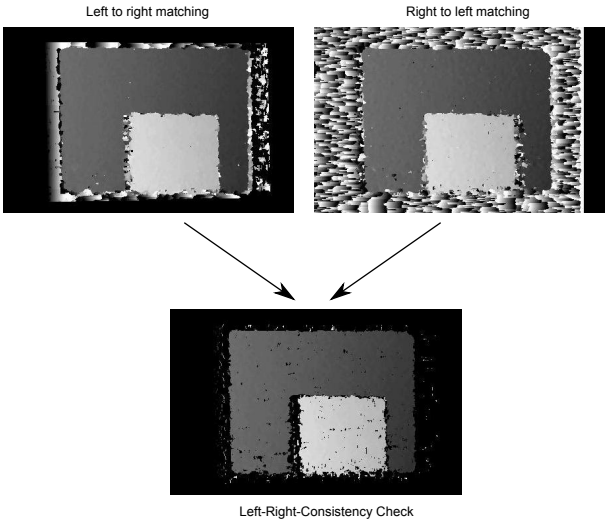


Figure 41: Left-Right-Consistency Check

distance and a data visualization tool called PfeDxHost, which shows the results of the stereo matching algorithms and communicates via Ethernet. For every image, that should by shown, an own DX-channel has to be created, for example the rectified images are sent via the channels named `left_rect` and `right_rect`.

In general, the whole stereo matching calculation process is the same, as it is implemented in MATLAB. Furthermore, lots of quality improvements, post-processing steps and speed enhancements are performed, which results in dense depth images. In detail it is explained in [25] and [26]. The information, that is necessary to perform the stereo algorithms, for example the internal and external camera parameters as well as the rectification matrices and the new camera matrices, are stored in an INI-file, which is loaded at the beginning of the program call. This file also contains other configuration settings, which are explained in [27]. An example of such an INI-file is listed below:

```
[S3E]
ptInputRoiOrigin.x = -50
ptInputRoiOrigin.y = -20
 szInputRoiSize.u32Width = 800
 szInputRoiSize.u32Height = 500
 ptInputRoiOrigin.x = 20
 ptInputRoiOrigin.y = 50
szInputRoiSize.f32Width = 640
szInputRoiSize.f32Height = 512
 u32Vergence = 150
szOutputImg.u32Width = 320
szOutputImg.u32Height = 256

CamLeft = left
CamRight = right
u32CensusType = 0x0b000000
szAggregation.u32Width = 5
szAggregation.u32Height = 5
```

```
u32MinDisp = 20
u32MinValidDisp = 1
u32MaxDisp = 100

pfe3DFileType = esvXYZ

u8ConfidThrshld = 10
u8TextureThrshld = 0
u16CCThreshold = 12
u16CCMinNrOfPix = 150
u32NumThreads = 6

u32DispScaleExp = 0

[left]
CM= 718.862031 0.0 305.532149 0.0 721.450182 251.411314 0.0 0.0 1.0
DC= -0.433507 0.130720 0.374995 0.002626 -0.001480
RM= 0.999714 0.00371767 -0.0236071 -0.003717679 0.9999930 0.00004 0.023607 0.000043 0.99972
PM= 711.8361 0.0 322.0776 0.0 711.8361  259.1078 0.0 0.0 1.0
BL= 0.2088879
CamSerNo = 4002841299

[right]
CM= 721.325664 0.0 312.859263 0.0 723.849515 254.121606 0.0 0.0 1.0
DC= -0.442249 0.256826 -0.101597 0.002104 0.000220
RM= 0.999797 -0.008817 -0.01807499 0.00876 0.999956 -0.0031732 0.018102 0.003014 0.9998314
PM= 711.8361 0.0 322.0776 0.0 711.8361  259.1078 0.0 0.0 1.0
BL= 0.2088879
CamSerNo = 4002841300
```

The parameters in the section `[left]` and `[right]` concern the camera parameters of left and right camera. `CM` defines the camera matrix, `DC` the distortion coefficients, `RM` the rectification matrix, `PM` the camera matrix after rectification, `BL` the baseline between the two cameras in meters and `CamSerNo` the serial number of the used camera.

If an iDS camera is connected, the software also needs a second INI-file, which includes the camera specific information, for example the camera driver, the actual used image width and height or the frame rate. This file is necessary for the software to communicate with the used camera.

An extract of this file is shown below:

```
[Versions]
ueye_api.dll=3.90.0011
ueye_usb.sys=3.90.0009
ueye_boot.sys=3.90.0009


[Sensor]
Sensor=UI124xSE-M


[Image size]
Start X=320
```

```
Start Y=256
Start X absolute=0
Start Y absolute=0
Width=640
Height=480
Binning=0
Subsampling=0


[Scaler]
Mode=0
Factor=1.000000


[Multi AOI]
Enabled=0
Mode=1
x1=0
x2=0
x3=0
x4=0
y1=0
y2=0
y3=0
y4=0


[Shutter]
Mode=2
Linescan number=0


[Timing]
Pixelclock=20
Framerate=17.644480
Exposure=1.787930
Long exposure=0
```

During this work the S3E software was extended to work with projector-camera systems. Therefore some functions have to be modified as well as some sections in the INI-file have to be added. The existing software is capable to work with two or three camera systems, which requires at least two cameras connected to the processing PC. For projector-camera systems just one camera is needed but nevertheless also two or three cameras could be connected. So first of all the function *IdsUsb_Init*, which loads the connected iDS cameras and their individual specifications, was modified in the way, that it also accepts one connected camera.

The second important process step of the software, which was changed, was the input of the images, that are used for the stereo calculations. The existing software uses the camera's images therefore. They are captured and stored by the function *CaptureImages* for futher calculations. When working with projector-camera systems one image of the stereo pair comes from the camera and the second one is the pattern, which is projected.

Therefore the pattern has to be provided to the software. For this purpose the visualization tool PfeDxHost was used, it is, namely, not just able to receive images, but also to send images. An own DX-channel has to be created, where the pattern is put in and sent. The software receives this image and performs the stereo calculations on the related camera's image and the pattern. But to perform rectification and stereo matching the necessary parameters have to be available. They are also stored in the INI-file. This file was modified as well as the function *EsvUpdateFromConfFile*, which loads the INI-file and stores the including informations. Therefore the INI-file got an own section called `[Projector]` and for every projector-camera pair two additional sections, that include the parameters necessary for rectification and stereo matching. The modified file is listed below:

```
[S3E]
ptInputRoiOrigin.x = -50
ptInputRoiOrigin.y = -20
 szInputRoiSize.u32Width = 800
 szInputRoiSize.u32Height = 500
 ptInputRoiOrigin.x = 20
 ptInputRoiOrigin.y = 50
szInputRoiSize.f32Width = 640
szInputRoiSize.f32Height = 512
 u32Vergence = 150
szOutputImg.u32Width = 320
szOutputImg.u32Height = 256

CamLeft = left
CamRight = right
u32CensusType = 0x0b000000
szAggregation.u32Width = 5
szAggregation.u32Height = 5


u32MinDisp = 20
u32MinValidDisp = 1
u32MaxDisp = 100

pfe3DFileType = esvXYZ

u8ConfidThrshld = 10
u8TextureThrshld = 0
u16CCThreshold = 12
u16CCMinNrOfPix = 150
u32NumThreads = 6

u32DispScaleExp = 0

[Projector]
bUseProjector = 1
ProjectorCalibl = projleft
CameraCalibl = camprojleft
ProjectorCalibr = projright
CameraCalibr = camprojright
PatternSource = proj_input
bSwapLR = 1
```

```
[left]
CM= 718.862031 0.0 305.532149 0.0 721.450182 251.411314 0.0 0.0 1.0
DC= -0.433507 0.130720 0.374995 0.002626 -0.001480
RM= 0.999714 0.00371767 -0.0236071 -0.003717679 0.9999930 0.00004 0.023607 0.000043 0.99972
PM= 711.8361 0.0 322.0776 0.0 711.8361  259.1078 0.0 0.0 1.0
BL= 0.2088879
CamSerNo = 4002841299

[right]
CM= 721.325664 0.0 312.859263 0.0 723.849515 254.121606 0.0 0.0 1.0
DC= -0.442249 0.256826 -0.101597 0.002104 0.000220
RM= 0.999797 -0.008817 -0.01807499 0.00876 0.999956 -0.0031732 0.018102 0.003014 0.9998314
PM= 711.8361 0.0 322.0776 0.0 711.8361  259.1078 0.0 0.0 1.0
BL= 0.2088879
CamSerNo = 4002841300

[camprojright]
CM= 721.325664 0.0 312.859263 0.0 723.849515 254.121606 0.0 0.0 1.0
DC= -0.442249 0.256826 -0.101597 0.002104 0.000220
RM=  0.987977 -0.0195901 0.153352 0.0172335 0.999712 0.0166817 -0.1536347 -0.0138384 0.9880307
PM= 743.0000 0.0 169.2886 0.0 743.0000 291.9651 0.0 0.0 1.0
BL= 0.080904
CamSerNo = 4002841300

[projright]
CM= 1628.0 0.0 348.7 0.0 1637.4 451.7 0.0 0.0 1.0
DC= -0.0426 0.2741 0.0 0.0018 0.0030
RM=  0.9783302 -0.012187 0.206691 0.0121871 0.9999249 0.0012732 -0.2066914 0.001273 0.9784053
PM= 743.0 0.0 169.2886 0.0 743.0 291.9651 0.0 0.0 1.0
BL= 0.080904

[camprojleft]
CM= 718.862031 0.0 305.532149 0.0 721.450182 251.411314 0.0 0.0 1.0
DC= -0.433507 0.130720 0.374995 0.002626 -0.001480
RM= 0.9924 0.0225 -0.1214 -0.0225 0.9997 0.0014 0.1214 0.0014 0.9926
PM= 711.8361 0.0 322.0776 0.0 711.8361 259.1078 0.0 0.0 1.0
BL= 0.1270019
CamSerNo = 4002841299

[projleft]
CM= 1634.0 0.0 312.9 0.0 1641.1 442.7 0.0 0.0 1.0
DC= -0.0311 0.2261 0 -0.0001 -0.0038
RM= 0.9956 0.0115 -0.0934 -0.0140 0.9996 -0.0259 0.0931 0.0271 0.9953
PM= 711.8361 0.0 322.0776 0.0 711.8361 259.1078 0.0 0.0 1.0
BL= 0.1270019
```

In the section `[Projector]` some definitions and settings are stored. `bUseProjector` is a boolean, which decides, if the projector is consulted for the further calculations. Only if this value is set, the pattern is loaded from the PfeDxHost and the stereo algorithms are performed on the projector-camera pair. `ProjectorCalibl`, `CameraCalibl`, `ProjectorCalibr` and `CameraCalibr` define the sections, which include the projector and camera parameters for the stereo system consisting of left camera and projector as well

as right camera and projector. If only one camera is connected just one section pair has to be defined. With the boolean `bSwapLR` the projector camera setup can be defined. The default setting is `bSwapLR = 0`, which means that the camera is on the left side and the projector on the right. If it is set one, the software performs the calculations the other way round. This boolean only has an influence if one camera is connected. Otherwise it is assumed that the projector is placed in the middle of the two cameras, hence for the pair left camera and projector `bSwapLR` is zero and for the pair projector and right camera it is one. The default setup is illustrated in Figure 42. `PatternSource` defines the name of the DX-channel, which sends the pattern from the PfeDxHost. The further calculations, which include rectification, stereo matching and all the post processing steps are performed the same as it is done by the existing S3E software.

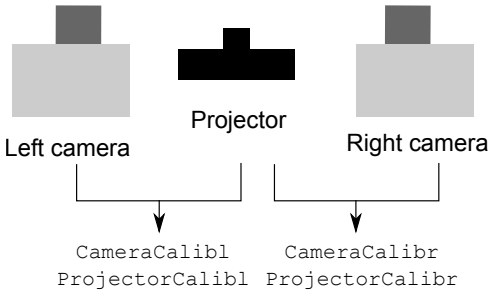


Figure 42: Default setup

An image of the visualization tool is shown in Figure 43. In the top row the left camera's image and the blurred pattern is shown. The second row illustrates the results from the stereo matching procedure: the left camera's rectified image, the rectified pattern, the resulting disparity and depth map.

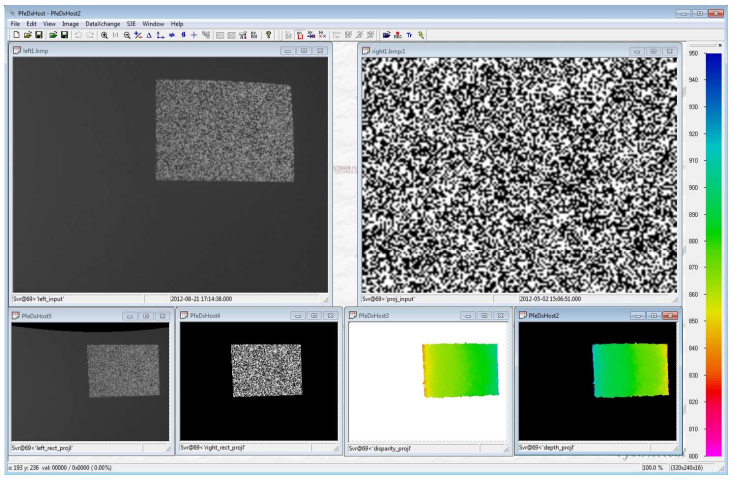


Figure 43: PfeDxHost platform

A further function, which has been integrated in the S3E software during this work, concerns the depth map combination. Therefore it is necessary, that two cameras are connected and that the projected pattern is sent from the PfeDxHost, in order to obtain two depth images, one from the two-camera sensor and one from the projector-camera sensor. All the processes, which are performed by the software to result in a depth image, have to be executed twice. To illustrate both outcomes, different DX-channel have to be created for the rectified, disparity or depth images, here they are named, *left_rect*, *right_rect*, *disparity* and *depth* for the two-camera results and *left_rect_projl*, *right_rect_projl*, *disparity_projl* and *depth_projl* for the corresponding images from the projector-camera sensor using the left of the two cameras. The function *CombineDepth* performs the combination of the two depth images. It takes as input parameters the image structure, which will be filled with the combined depth values, the depth image of the two-camera sensor, the 3D point set of the projector-camera sensor, the individual calibration parameters for the left camera of the two-camera system and for the left camera of the projector-camera setup and the predefined value for the pixels, whose depth is not available. The function transforms the 3D points to the coordinate system according to the left camera of the two-camera sensor, by multiplying them with $R^c(R^p)^{-1}$, where R^c is the rectification matrix of the first calibration input parameters and R^p the one of the second. Furthermore, the final combined depth map is filled, as it is explained in section 6.4. For the resulting depth image, an own DX-channel, *depth_final* is created. The output within the PfeDxHost is illustrated in Figure 44. The first row shows the left and right camera's image as well as the projected pattern, the second row illustrates the two-camera and projector-camera disparity images, while the third row shows the corresponding depth images as well as the combined depth image. As can be seen, the combined depth image is the densest one since it contains depth values from both sensors. More evaluation will be presented in section 8.

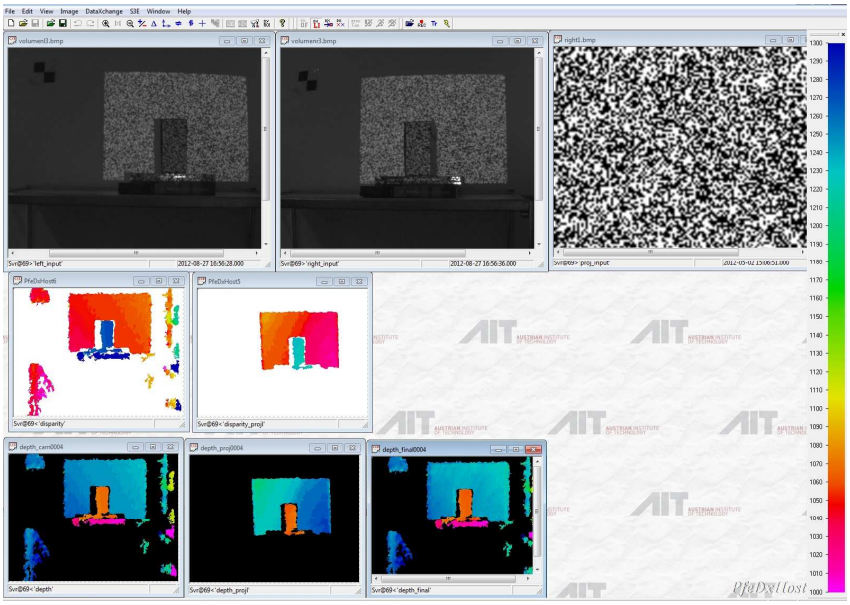


Figure 44: PfeDxHost output

7.3 Automatic projector-camera calibration

The existing projector-camera calibration, as it is implemented in MATLAB and explained in section 7.2.2, needs input from the user, the calibration plane has to be identified in the camera's images as well as the projected pattern. Therefore a window is opened for every image, where the user has to click on the necessary points. This is not very practical, since it is time consuming and dependent on the accuracy of the user's input. To make this calibration method more practical, some automation would be desirable. For this purpose, it is important to know the structure of the used pattern, since the implemented software has to find some calibration points within this pattern. Therefore such an automation cannot be generalized for every projector. The methods which are implemented here, can deal with pointed patterns, an example is shown in Figure 45.

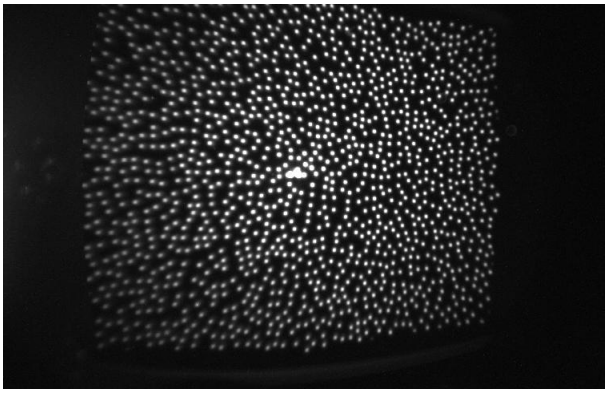


Figure 45: Pointed pattern

The first step of every calibration procedure is the capturing of the calibration images. The existing projector-camera calibration expects pictures, where the identification markers of the plane as well as the projected pattern are visible. The markers are necessary to calculate the equation of the planar surface. For the automatic calibration procedure implemented here, another device is used to achieve that, namely a second camera. The two cameras build an own stereo system, which also needs to be calibrated first, but with the help of it, the equation of the plane can be calculated. Therefore the calibration images have to be captured by both cameras simultaneously and with different viewing angles. It should be taken care that most of the points of the pattern are visible in both images. As it is done by the existing projector calibration, the pattern is projected onto a planar surface and since the two cameras are calibrated, they build a stereo system, hence it is possible to calculate the 3D coordinates of the matched point correspondences. Here, the existing S3E software is used to achieve that. A binary file is written by the software, which includes these coordinates for all found correspondences, where the reference coordinate system is set at the left camera's one. The MATLAB function `ReadCameraBinary` reads this file and the 3D points P are further taken as an input to the MATLAB function `PlaneFittingLTS(P)`, which calculates the best fitting plane, expressed by its normal vector and distance from the coordinate system's origin for this set of points. In this manner every image pair is processed and the plane's equations

with respect to the left camera's coordinate system are calculated. What has to be taken into account is, that the left camera's image plane has already been rotated, according to the related rectification matrix R before the 3D coordinates of the point correspondences are calculated. Therefore the left camera's coordinate system is not orientated, as it was originally. To reverse this rotation, the normal vector of every calibration plane has to be multiplied by the inverse of R, which transforms the vector back to the left camera's original coordinate system.

The next step during the automatic projector-camera calibration is the identification of some points within the projected pattern. The original pattern is illustrated in Figure 46.

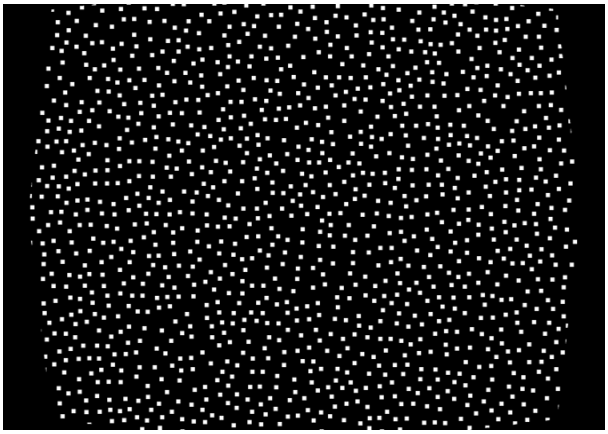
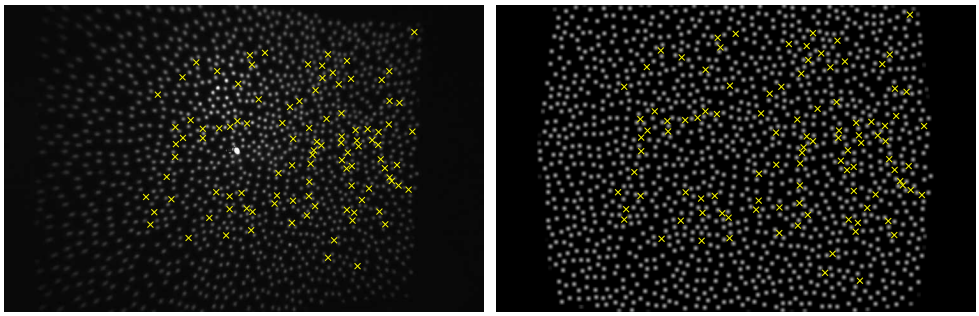


Figure 46: Original pointed pattern

The point identification is based on the SIFT algorithm, which is an algorithm to detect image features and which describes their local neighborhood with vectors called descriptors [28]. This feature detection method is chosen because of its scale and rotational invariance. Especially the rotational invariance is important here, since the calibration images are captured from different angles. The software used for this work is a MATLAB implementation from A. Vedaldi and B. Fulkerson [29]. First of all, every calibration image of the left camera is processed and the found SIFT features and the appropriate descriptors are saved. To get rid of the camera's distortion, the images are undistorted before the feature extraction is executed. Next, the original pattern is processed. For this purpose it is recommended to blur the binary pattern. Especially for the pointed pattern used here, it improves the quality of the feature extraction procedure because the pattern consists of many black areas, which do not provide any information and features are difficult to find. The software from Vevaldi and Fulkerson also provides a function which matches the descriptors of two images, here, the descriptors of each camera image are matched with the ones of the pattern. According to the different viewing angles, different points and as a result, different matches are found within each image. Therefore, the matches which are found within every image have to be filtered. An example of the remaining matches is shown in Figure 47.

Basically the projector calibration could be performed using these points, but since the found matches do not coincide with the dots of the pattern, the correspondences are inaccurate. To overcome this, a rectangular neighborhood is scanned for the white dots.



(a) Camera image (b) Pattern

Figure 47: Found matches within (a) camera image and within (b) pattern

For every feature point exactly one neighboring point is chosen. These points are defined within the pattern first. For every found match, (x, y) a rectangular neighborhood, $[x-20, x+20] \times [y-20, y+20]$ is scanned using the MATLAB function `regionprops`, which finds the centers of every white dot within the neighborhood. This function works best for binary images, hence for the local neighborhood of each feature point the binary version of the pattern is chosen. The dot, which is closest to the SIFT feature point, is defined as its neighboring pattern point. This is illustrated in Figure 48. The feature point is colored yellow, the found neighboring dots red, and the neighboring pattern point is colored green.

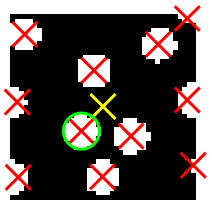


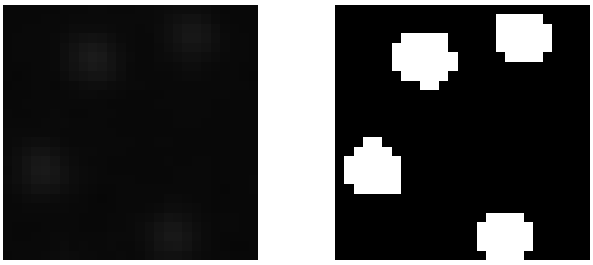
Figure 48: Neighborhood of pattern feature point

Next, exactly the same neighboring dots have to be found within all camera images. Therefore, also a rectangle neighborhood surrounding each feature point is scanned. Here, a binary image should be taken as well to extract the neighboring points. The binarisation of the neighborhood blocks is based on a predefined threshold δ, hence the values of the binary image I_{bin} are calculated as

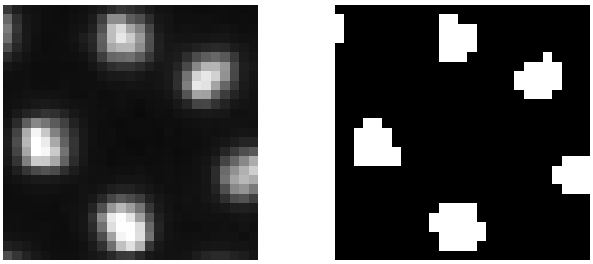
$$I_{bin}(x, y) = \begin{cases} 1 & I(x, y) > \delta \\ 0 & I(x, y) \leq \delta \end{cases}$$

Since the image intensities within the camera's images are varying, a global threshold for all neighborhood blocks cannot be defined. If it is too high, the dots within less intense regions cannot be identified and if it is too low, additional points could be arise within more intense areas due to the image noise, hence for every local neighborhood an own threshold has to be defined according to its intensity. Here, the MATLAB function

`graythresh` provides this value. In Figure 49, a less intensive neighborhood and a more intensive neighborhood together with their binary images are shown. The threshold of the first one is 0.0510, while the threshold of the second image is 0.3804.



(a) Less intensive (b) Corresponding binary



(c) More intensive (d) Corresponding binary

Figure 49: Different neighborhood blocks, (a) and (b): Less intensive neighborhood and corresponding binary image ($\delta = 0.0510$), (c) and (d): More intensive neighborhood and corresponding binary image ($\delta = 0.3804$)

For the selection of the neighboring image point it is not advisable to chose the dot, which is closest to the feature point, because the camera's images of the projected pattern are all oriented differently and therefore different neighboring image points could be chosen. A better approach is to chose these ones which are closest to the previously found neighboring pattern points. This is shown in Figure 50.

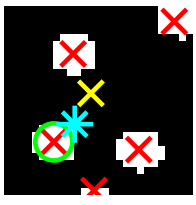
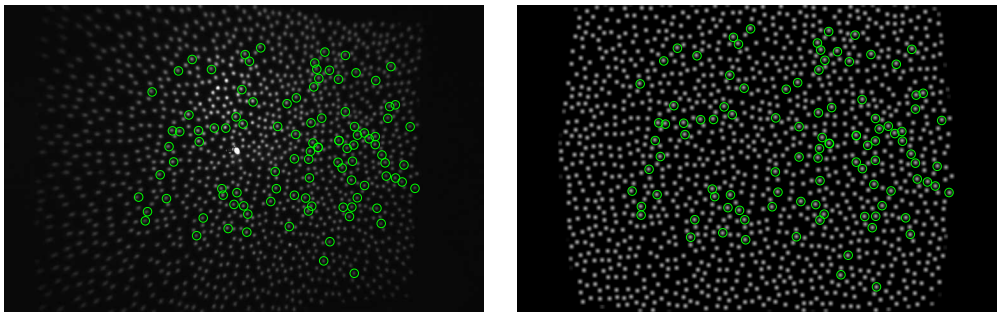


Figure 50: Neighborhood of image feature point

Here, the feature point is also colored yellow, the found neighboring dots red, and the neighboring image point is colored green. The cyan star marks the location of the

previously defined neighboring pattern point. In this example it is recognizable that another dot is closer to the feature point, hence a wrong neighboring image point would have been chosen.

In Figure 51 the final neighboring image and pattern points which are used for the further calculations can be seen.



(a) Neighboring image points (b) Neighboring pattern points

Figure 51: Chosen dots within (a) camera image and within (b) pattern

Now all the information which is necessary for the projector-camera calibration is available, hence the remaining calibration procedure can be performed the same way as it is done in the implemented projector-camera calibration in MATLAB.

The calibration results are listed below:

```
Calibration results (with uncertainties):

Focal Length:          fc = [ 594.66509   594.63885 ]
Principal point:       cc = [ 356.90350   234.93515 ]
Skew:             alpha_c = [ 0.00000 ]    => angle of pixel axes = 90.00000 degrees
Distortion:            kc = [ -0.50345   0.35287   0.00386   0.00007  -0.20264 ]
Pixel error:          err = [ 0.67358   1.27117 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

If you look at the calibration results in section 7.2.2, which is the result of the previous calibration procedure, it can be seen that the error values are much smaller now. This is due to the higher accuracy in identifying the calibration points, since they are found by the algorithm and not clicked by the user. An important fact concerning this automatic calibration method is, that the two-camera stereo system has to be calibrated accurately since its 3D reconstruction results are used during the projector calibration for the calculation of the equation of the illuminated plane. So the better this system is calibrated, the more accurate are the results of the projector-camera calibration.

7.4 Summary

In this chapter the used hardware as well as the used software for this work is presented. This concerns the camera calibration and the projector-camera calibration in MATLAB as

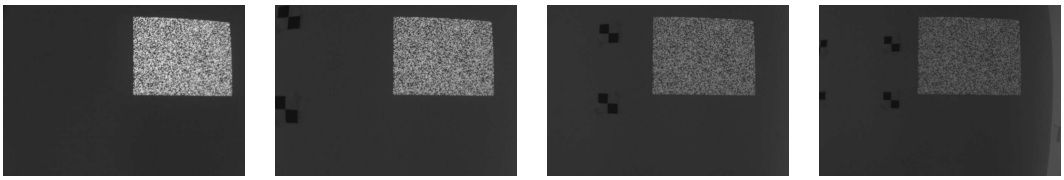
well as the stereo vision engine S3E. Furthermore, the implemented software is described in detail. This concerns modifications of the projector-camera calibration, the rectification and stereo matching in MATLAB as well as modifications of the S3E software to integrate projector-camera systems. Last an automatic projector-camera calibration method which is implemented in MATLAB is presented. It can deal with pointed patterns and is based on the feature detection algorithm SIFT. First, the calibration points have to be detected in all calibration images and in the pattern. Then some refinement procedures are processed in order to select the same points within all images. Last, the already existing calibration method can be performed on this set of point correspondences. The calibration results of this method are proven to result in more accurate parameters, since it is independent of the user's input like the other projector-camera calibration method.

8 Evaluation

In this section the algorithms, used and developed in this work, are evaluated. First of all, it is proven that the projector-camera calibration implemented in MATLAB provides correct results. This is achieved by measuring a planar surface from different known distances, documenting the percentage of calculated points, which do not fulfill the plane's equation and comparing the calculated 3D coordinates of the plane with its real ones. Furthermore, the matching quality is analysed according to the density of the resulting depth map. Next, the accuracy of the proposed calibration method is determined by evaluating the discrepancies in measuring the dimensions of a selected test object and last the reconstruction results of the projector-camera stereo system are compared with the ones of a classical two-camera stereo sensor. It is shown in which cases the first one provides better results and in which the second is more accurate.

8.1 Verification and measuring accuracy

To prove that the projector-camera calibration method, which is implemented in MATLAB, works and its outputted parameters provide correct results for 3D reconstruction, a plane is measured. Therefore, the projector-camera system is placed in front of a planar surface, the projector illuminates it, and the camera captures an image of it. This is done from different distances, the left camera's images are illustrated in Figure 52 and ordered from near to far.



(a) Nearest (b) Second nearest (c) Second farthest (d) Farthest

Figure 52: Plane test images from near to far, (a) - (d)

These images together with the blurred pattern are processed by the S3E software and the calculated 3D coordinates are stored in a binary file. Next, the 3D coordinates are read in MATLAB with the function `ReadCameraBinary`. For the first verification test, the equations of the planes according to the computed 3D points at each distance are calculated. The MATLAB function `PlaneFittingLTS` calculates the normal vector and the distance from the origin as well as a vector, which encodes the points which do not fulfill the plane's equation. Those are the points whose absolute distances to the calculated planes are higher than a predefined bound. Since only planar surfaces are measured, all of the calculated coordinates should be situated on these planes, hence, the number of the outliers indicates how exact the 3D coordinates are computed. The results are listed in Table 1.

Since only at most 1.65 % of the calculated 3D points do not fulfill the plane's equations, it can be said that the calibration procedure and the subsequent stereo matching

	calculated points	outliers	%
Figure 52, (a)	15018	247	1.6447
Figure 52, (b)	15254	224	1.4685
Figure 52, (c)	15477	172	1.1113
Figure 52, (d)	15409	155	1.0059

Table 1: Plane measuring results

algorithms provide accurate results.

Next, the measuring accuracy is analysed. Therefore, the 3D coordinates of the four corners of the pattern are considered. The 3D coordinates stored in a binary file are calculated with respect to the rotated coordinate system of the left camera after rectification. Therefore, they have to be multiplied with the inverse rectification matrix to get the coordinates in the original left camera's coordinate system. Then, the calculated coordinates of the four corners of the pattern have to be found and their Euclidean distances have to be computed. To verify these calculated points, the distances of the four corners of the pattern to the left camera are also measured with a measuring tape. These distances can then be compared with the computed distances. The whole verification workflow is illustrated in Figure 53 and the results are shown in Table 2.

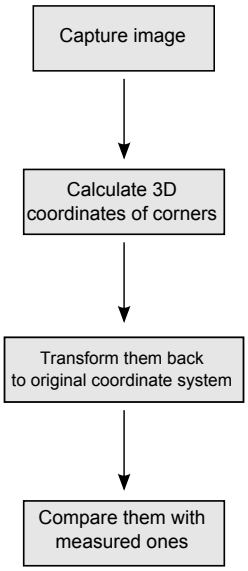


Figure 53: Verification workflow

The four values for every image in the table are the Euclidean distances between the camera and the four corners of the pattern, in the order top left, top right, bottom right, and bottom left. All of them are given in the unit centimeters. The theoretical error is calculated as follows.

Depth is calculated by $Z(d) = \frac{b \cdot f}{d}$, where b is the length of the baseline in centimeters, f the focal length in pixel, and d the value of the disparity also in the unit of pixel. The

	measured	calculated	disparity	error	theoretical error
Figure 52, (a)	62.2	61.47	310	0.73	0.7552
	67.5	66.82	292	0.68	0.8512
	66.1	64.79	290	1.31	0.8630
	60.9	59.07	309	1.83	0.7601
Figure 52, (b)	78.8	77.91	246	0.89	1.1993
	84.6	83.72	231	0.88	1.3601
	82.8	81.2	229	1.6	1.384
	76.9	75.27	244	1.63	1.219
Figure 52, (c)	109.5	108.53	178	0.97	2.2906
	114.9	112.26	171	2.64	2.482
	112	109.18	169	2.82	2.5411
	106.7	104.57	177	2.13	2.3166
Figure 52, (d)	138.5	137.7	141	0.8	3.6506
	144.7	141.68	135	3.02	3.9823
	140.8	138.15	133	2.65	4.1029
	134.7	133.85	139	0.85	3.7564

Table 2: Measuring accuracy

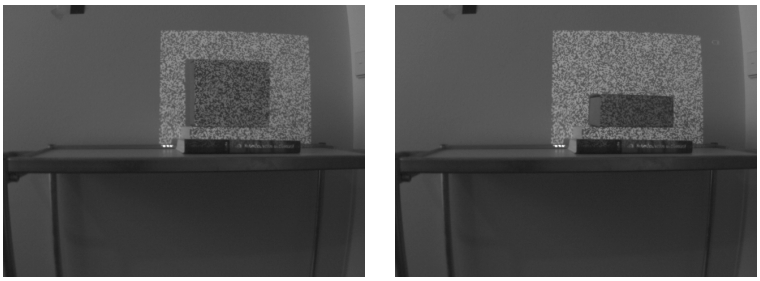
modulus of the first derivation, $\frac{b \cdot f}{d^2}$, describes the change of depth near every disparity value. The camera used for this test has a focal length of 711.09 pixel and the baseline between camera and projector amounts to 12.758 centimeters. The resolution of the camera images is 640×480 and since the output images, the rectified, disparity and depth images, have a width of 320 pixel, the value of the derivation has to be multiplied by $\frac{320}{640} = 0.5$. Another fact which has to be taken into account is, that the disparity values are calculated with subpixel accuracy, the setting in the INI-file used here is 2^2, which means that every outputted disparity value has to be divided by 2^2 to determine the floating-point value. So in this test case the formula for the theoretical error is

$$\frac{12.758 \cdot 711.09 \cdot 0.5}{(\frac{d}{4})^2}$$

As can be seen, within the table, the error values are mostly smaller than the theoretical error values. Only for a few entries the calculated distances are not within the theoretical range, which can also be the result of inaccuracies during the measurement with the measuring tape.

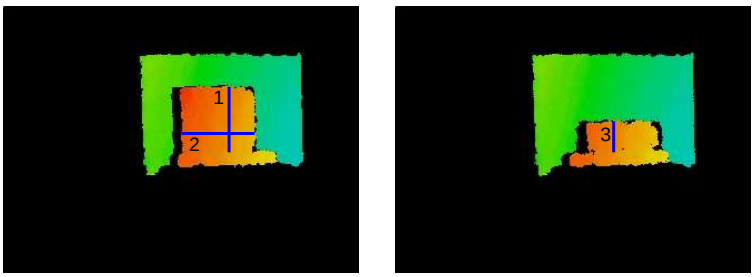
Another accuracy test is performed by measuring the dimensions of a rectangular box. Two images are captured, one from the front and one from the side, in order to obtain the height, the width, and the length. They are shown in Figure 54. Again, the 3D coordinates are stored in a binary file and read in MATLAB by the function `ReadCameraBinary`. Next, the necessary coordinates are extracted. Chosen are those two points for each distance which are in exactly the same line or row of the depth image and where a valid depth value is available. The 3D coordinates of these points are extracted and the length of the vector is calculated. In Figure 55, the chosen vectors for distance calculations are shown and in Table 3 the calculated results can be compared with the

measured results. As can be seen, the error values are not higher than 3 millimeters, which shows that the projector-camera system is calibrated correctly and provides accurate results.



(a) Test image 1 (b) Test image 2

Figure 54: Dimension measuring test images



(a) Distances 1 and 2 (b) Distance 3

Figure 55: Calculated distances: (a) Distances 1 and 2, (b) Distance 3

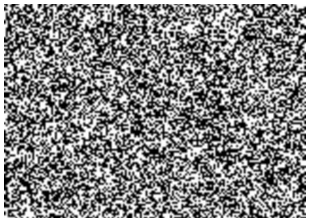
	measured	calculated	error
Distance 1	15.5	15.35	0.15
Distance 2	18	17.78	0.22
Distance 3	7.3	7.29	0.01

Table 3: Dimension measuring results

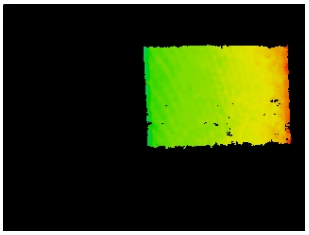
8.2 Matching quality

An important detail concerning the matching quality has to be taken into account, when measuring with projector-camera systems. Since the camera's images of the projected pattern are not as sharp as the original pattern, which is usually a binary image, the density of the resulting depth map is not as high as it could be. Therefore the pattern,

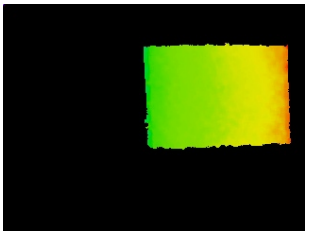
which is sent from the PfeDxHost, has to be blurred first. Here, this is achieved by using the MATLAB functions `fspecial`, which creates an image filter, and `imfilter`, which applies this filter to the pattern. The blurred pattern and a comparison of the calculated depth maps, using the binary pattern and the blurred are illustrated in Figure 56. As can be seen, the reconstruction using the blurred pattern results in a denser depth map. In Table 4 the numbers of calculated points are listed.



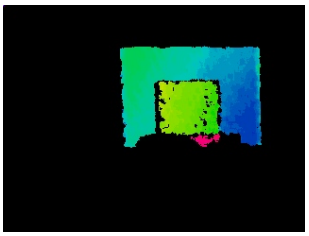
(a) Blurred pattern



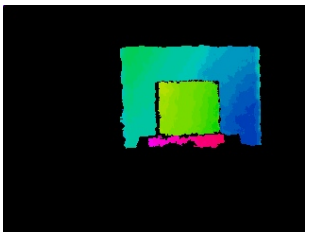
(b) Depth map using binary



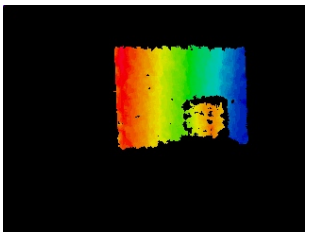
(c) Depth map using blurred



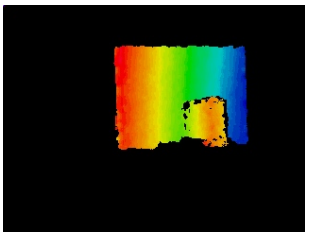
(d) Depth map using binary



(e) Depth map using blurred



(f) Depth map using binary



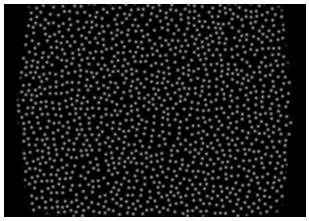
(g) Depth map using blurred

Figure 56: Calculation results with binary and blurred pattern, (a) Blurred pattern, (b), (d) and (f) Depth maps using the binary pattern, (c), (e) and (g) Depth maps using the blurred pattern

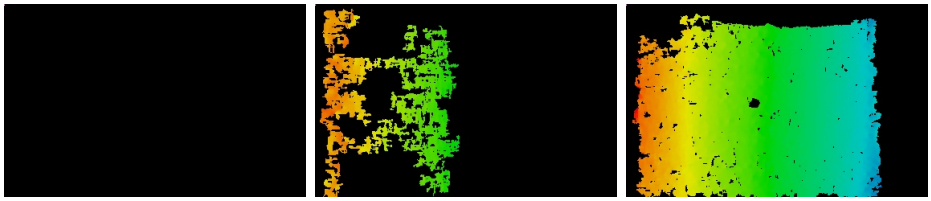
		binary	blurred	difference
Figure 56	(b) and (c)	15845	16173	328
	(d) and (e)	13296	14442	1146
	(f) and (g)	12810	13891	1081

Table 4: Number of matching points

The pointed pattern, which is used for the automatic calibration procedure, explained in section 7.3, is also blurred before the stereo matching procedure is performed. If the binary pattern would have been used, hardly any matches could be found due to the huge black fields that cover most of the pattern. Therefore a greater matching block size would be necessary which would decelerate the whole procedure. The blurred pattern and a comparison of the matching results is illustrated in Figure 57 and in Table 5 the number of calculated points is listed.

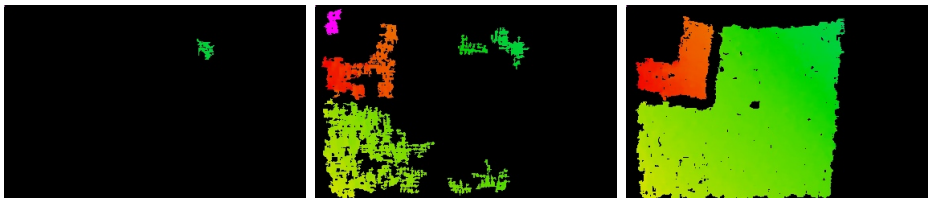


(a) Blurred pointed pattern



(b) Depth map using 7×7 binary (c) Depth map using 11×11 binary (d) Depth map using 7×7 blurred

(e) Depth map using 7×7 binary (f) Depth map using 11×11 binary (g) Depth map using 7×7 blurred



(h) Depth map using 7×7 binary (i) Depth map using 11×11 binary (j) Depth map using 7×7 blurred

Figure 57: (a) Blurred pointed pattern, (b), (e) and (h) Depth maps using 7×7 binary pointed pattern, (c), (f) and (i) Depth maps using 11×11 binary pointed pattern, (d),(g) and (j) Depth maps using 7×7 blurred pointed pattern

		bin.(7×7)	bin.(11×11)	bl.(7×7)	diff.(7×7)	diff.(11×11)
Figure 57	(b) - (d)	0	14117	60362	60362	46245
	(e) - (g)	789	10233	39661	38872	29428
	(h) - (j)	258	13236	47034	46776	33798

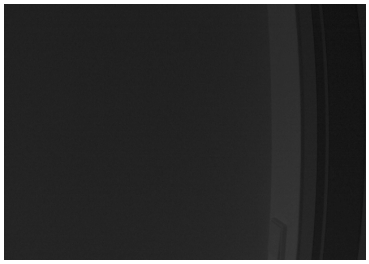
Table 5: Number of matching points

8.3 Depth map comparison

As already explained in this work, there are scenes and setups, which are difficult to measure with classical two-camera stereo systems, for example textureless areas. In these cases projector-camera systems provide better results. On the other hand such systems have difficulties, if the projected pattern is not visible sufficiently. Since two different baselines are used for measuring, different occlusions occur, hence there are scenes, where the resulting depth maps are denser when using the projector-camera setup and others where the two-camera sensor creates better results.

For the first depth map comparison a near textureless area, a white wall, is measured. The two-camera stereo system is not able to find any point correspondences, as it is illustrated in Figure 58. The same scene does not make any problems to the projector-camera system, since the projected pattern structures the white wall, it is also possible for the two cameras to measure the illuminated part of it, as it can be seen in Figure 59.

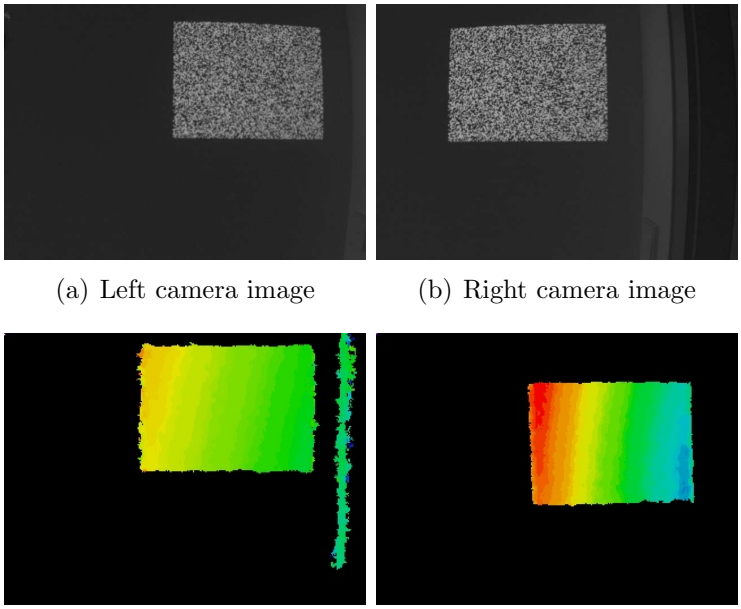
(a) Left camera image



(b) Right camera image

(c) Two-camera depth map

Figure 58: Textureless area, (a) and (b) Left and right camera image, (c) Resulting two-camera depth map

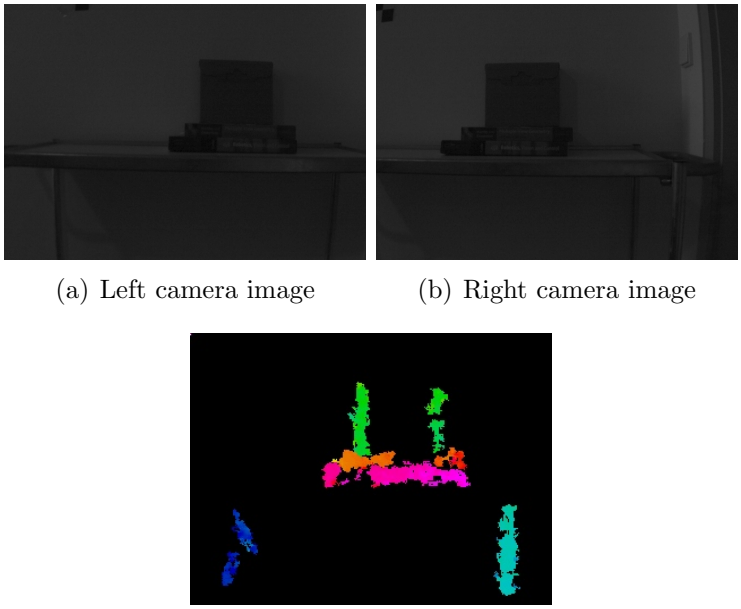


(a) Left camera image (b) Right camera image

(c) Two-camera depth map (d) Projector-camera depth map

Figure 59: Textureless area with projected pattern, (a) and (b) Left and right camera image, (c) Resulting two-camera depth map, (d) Resulting projector-camera depth map

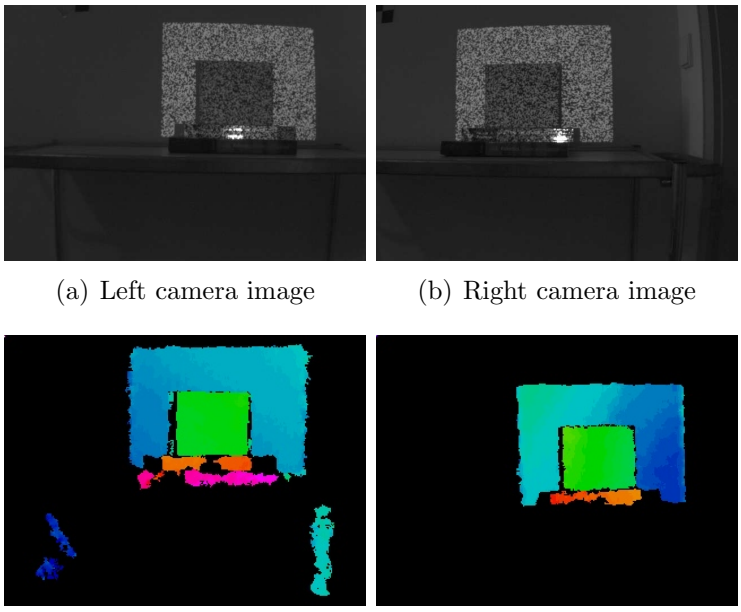
Next, a paper box is placed in front of the white wall. Here, the classical stereo sensor finds some correspondences, but in comparison to the illuminated scene, the depth image is barely filled. The test images and results are shown in Figure 60 and 61.



(a) Left camera image (b) Right camera image

(c) Two-camera depth map

Figure 60: Paper box, (a) and (b) Left and right camera image, (c) Resulting two-camera depth map



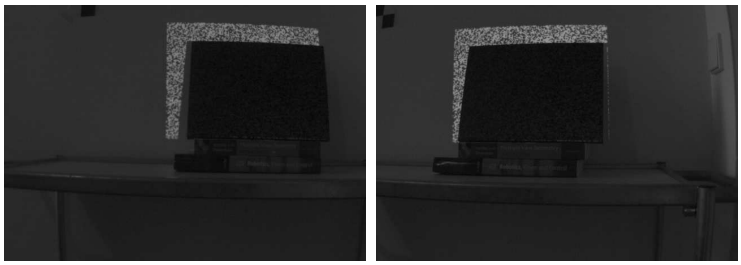
(a) Left camera image (b) Right camera image

(c) Two-camera depth map (d) Projector-camera depth map

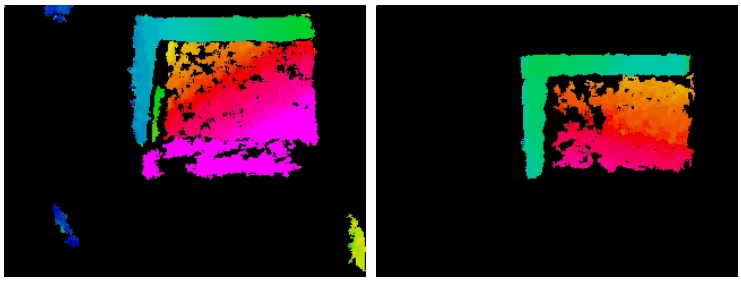
Figure 61: Paper box with projected pattern, (a) and (b) Left and right camera image, (c) Resulting two-camera depth map, (d) Resulting projector-camera depth map

The third setup includes a black book in front of the white wall. The book is a nearly textureless area, hence the two-camera sensor has difficulties in measuring it. The

projected pattern is hardly identifiable in the camera's image, hence the projector-camera system has also problems to calculate a dense depth map. In Figure 62 these results are shown.



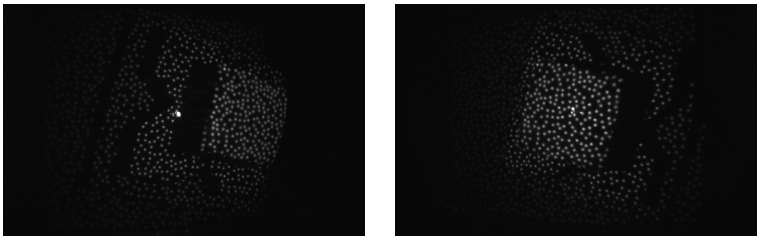
(a) Left camera image (b) Right camera image



(c) Two-camera depth map (d) Projector-camera depth map

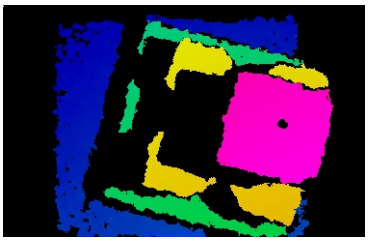
Figure 62: Black book with projected pattern, (a) and (b) Left and right camera image, (c) Resulting two-camera depth map, (d) Resulting projector-camera depth map

Next, some scenes are analysed, where the pointed pattern is used. Figure 63 and Figure 65 show the left and right camera images. As can be seen, there are areas which are visible in the left camera image but not in the right camera image, due to the different viewing angles of the two cameras. These areas are the so-called occlusions, where it is impossible for the two-camera sensor to calculate depth, but the projector-camera system does not have any difficulties, since the original pattern is always completely available. The resulting depth maps are illustrated in Figure 64 and 66.

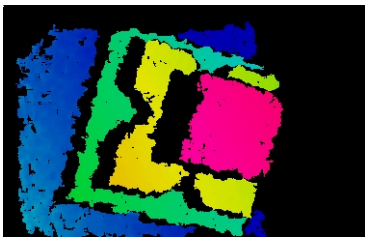


(a) Left camera image (b) Right camera image

Figure 63: Images of test object, (a) and (b) Left and right camera image

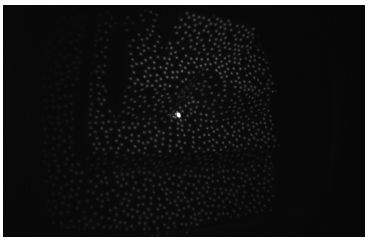


(a) Two-camera depth map

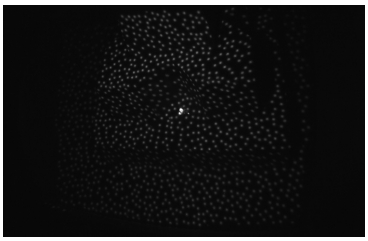


(b) Projector-camera depth map

Figure 64: Calculation results, (a) and (b) Two-camera and projector-camera depth maps

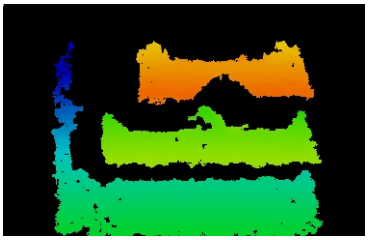


(a) Left camera image

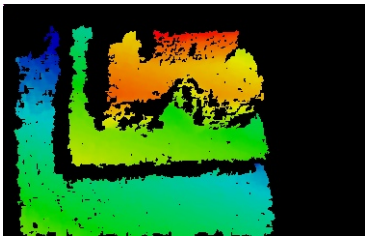


(b) Right camera image

Figure 65: Images of test object, (a) and (b) Left and right camera image



(a) Two-camera depth map

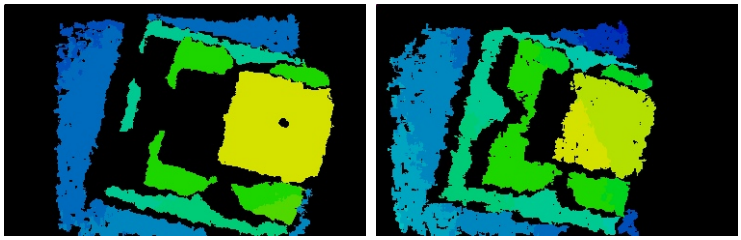


(b) Projector-camera depth map

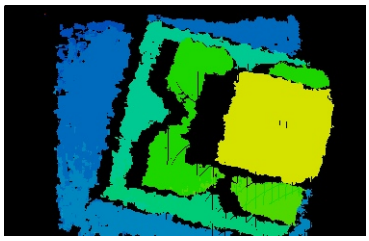
Figure 66: Calculation results, (a) and (b) Two-camera and projector-camera depth maps of test object

Furthermore, some results of the depth combination of the images from the two-camera sensor and the projector-camera system are shown. As can be seen, the final combined depth image is always denser than the first and the second one. The occlusions are eliminated due to the availability of depth values in the projector-camera result

and areas, where the projected pattern is not sufficiently visible in the left camera's image can also be measured, because the two-camera sensor provides the necessary depth information.

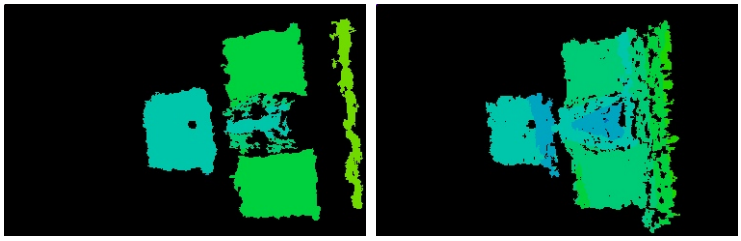


(a) Two-camera depth map (b) Projector-camera depth map

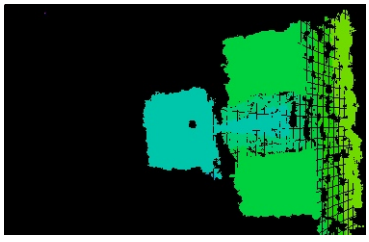


(c) Combined depth map

Figure 67: Calculation results, (a), (b) and (c) Two-camera, projector-camera and combined depth maps of test object

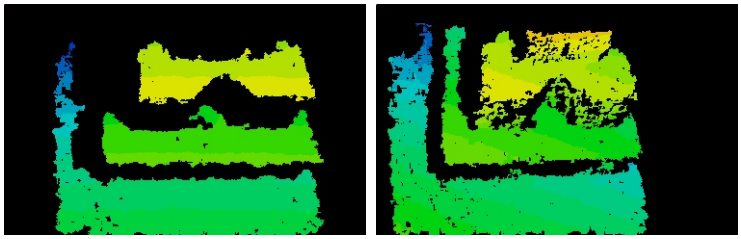


(a) Two-camera depth map (b) Projector-camera depth map

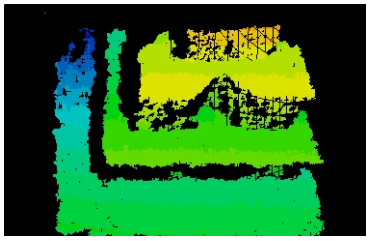


(c) Combined depth map

Figure 68: Calculation results, (a), (b) and (c) Two-camera, projector-camera and combined depth maps of test object

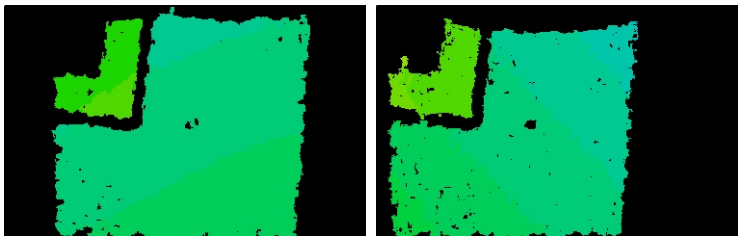


(a) Two-camera depth map (b) Projector-camera depth map



(c) Combined depth map

Figure 69: Calculation results, (a), (b) and (c) Two-camera, projector-camera and combined depth maps of test object



(a) Two-camera depth map (b) Projector-camera depth map

(c) Combined depth map

Figure 70: Calculation results, (a), (b) and (c) Two-camera, projector-camera and combined depth maps of test object

In Table 6, the number of calculated points can be compared. It is recognizable, that the combined depth image is always the densest one, hence the effort of additional

calculation costs is worth it.

	two-camera	projector	combination	difference(cameras)	difference(projector)
Figure 67	37141	39692	46626	9485	6934
Figure 68	20563	22383	27842	7279	5459
Figure 69	33584	40001	42762	9178	2761
Figure 70	51898	47160	52552	654	5392

Table 6: Number of matching points

8.4 Summary

In this chapter the implemented algorithms are tested. First of all, the projector-camera calibration is verified and proven to output accurate calibration parameters. First, this is done by measuring a plane, fitting a plane through all the calculated 3D points and documenting the percentage of outliers. Next, the 3D coordinates of the corners of the projected pattern are calculated and compared with the results of measuring with a measuring tape. Last, the dimensions of a rectangular box are measured and compared with the real dimensions.

Furthermore, the matching quality of the projector-camera system is analysed. It is shown that a blurred pattern results in denser depth maps. Last, the depth maps of a two-camera system are compared with the depth maps of a projector-camera system and also with the combined depth maps. Some setups are shown where the two-camera system is not able to measure depth but in combination with the projection of the random pattern it has no measuring difficulties. Furthermore some results of the depth map combination are documented which show that missing depth values of the two-camera system as well as of the projector-camera system are reduced.

9 Conclusion and Future work

In this work a special stereo vision system is presented, a projector-camera system. This sensor is similar to classical stereo vision systems where two cameras are used. The only difference is, that one camera is replaced by a pattern projector. Usually two-camera systems are calibrated. This means, that both cameras are capturing a calibration object, while the calibration software computes the camera parameters, that are needed for further stereo calculations. Therefore also projector-camera sensors have to be calibrated, whereby the challenge here is, that the projector is not able to capture the scene, hence classical camera calibration methods cannot be used. In this work a projector calibration method is introduced, which is implemented in MATLAB and provides the information, that is necessary for three-dimensional reconstruction of a scene. This procedure works best, when rectangular patterns are projected. It is shown, that the proposed calibration method works correctly and provides accurate results for the reconstruction procedure. A disadvantage of this calibration approach is, that it needs much input from the user, hence the accuracy of the calculation results is depended on the user's input accuracy. Therefore a different method is introduced here, which performs the projector calibration processes automatically and is adopted to pointed patterns. Also this procedure is proven to result in dense and accurate depth images.

Furthermore, the census, which is used for the disparity calculation in combination with SHD, is discussed. This method is chosen here, because it compares relative positions of local intensity values between the two images used for stereo matching, rather than the intensities themselves. Since one of the images is the projected pattern and the other one is the camera's image of it, the global intensities are different, hence matches would be difficult to find. If the relative intensity positions are compared, it does not matter, which values the pixels have. In connection with this method some improvements are also discussed and implemented.

It is shown, that projector-camera sensors improve the reconstruction quality in areas or scenes, where classical stereo vision systems have measuring problems. It is also discussed, when projector-camera sensors have difficulties. In this context a multiple baseline sensor is introduced. It is a combination of a two-camera stereo system with a projector-camera setup. The two baselines are the distances between left and right camera and between left camera and projector. Both of these two calibrated stereo systems provide a depth image and each of them has different measuring problems, hence a combination of the depth images eliminates both difficulties. It is shown, that the combined sensor improves the reconstruction density significantly, hence the additional calibration and calculation effort is worth it.

The future work could include an improvement of the projector-camera calibration method. Up to now, a precalibrated camera is necessary to perform the calibration. The accuracy of the projector calibration is therefore depended on the accuracy of the previously executed camera calibration. A desirable solution could be a method, where the parameters of both, camera and projector, are calculated simultaneously, without the need of different calibration images. Another disadvantage is, that the implemented projector calibration methods are adapted to particular patterns. It would be desirable, to develop an universally usable procedure, which can deal with all kinds of patterns and

performs all processes automatically.

Furthermore, the calculations concerning the multiple baseline sensor could be studied more closely. Especially the combination of two available depth values could be improved. Up to now, the mean value is inserted, but due to the different lengths of the two baselines, the depth value of one system may be more accurate than the other one. Therefore more precise values should be weighted higher.

10 Bibliography

References

[1] M. Maruyama and S. Abe. Range sensing by projecting multiple slits with random cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:647–651, 1993.

[2] P. Vuylsteke and A. Oosterlinck. Range image acquisition with a single binary-encoded light pattern. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:148–163, 1990.

[3] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissanov. Structured light using pseudorandom codes. *Pattern Analysis and Machine Intelligence*, 20:322–327, 1998.

[4] A. Shpunt and B. Pesach. Optical pattern projection. US Patent Application 20100284082, 2010.

[5] D.C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37:855–866, 1971.

[6] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *Seventh International Conference on Computer Vision (ICCV 99)*, 1:666, 1999.

[7] O. Schreer. *Stereoanalyse und Bildsynthese*. Springer, 2005.

[8] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly Media, Inc., 2008.

[9] J. L. Posdamer and M. D. Altschuler. Surface measurement by space-encoded projected beam systems. *Comput. Graph. Image Processing*, 18:1–17, 1982.

[10] S. Inokuchi, K. Sato, and F. Matsuda. Range imaging system for 3D object recognition. *International Conference on Pattern Recognition*, pages 806–808, 1984.

[11] Peisen, S. Huang, and Song Zhang. Fast three-step phase-shifting algorithm. *Applied Optics*, 45:5086–5091, 2006.

[12] D. Bergmann. New approach for automatic surface reconstruction with coded light. *Proc. SPIE*, 2572:2–9, 1995.

[13] T. Weise, B. Leibe, and L. Van Gool. Fast 3D scanning with automatic motion compensation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, pages 2–4, 2007.

[14] K. L. Boyer and A. C. Kak. Color-encoded structured light for rapid active ranging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:14–28, 1987.

[15] J. Le Moigne and A. M. Waxman. Structured light patterns for robot mobility. *IEEE International Journal of Robotics and Automation*, 4:541–548, 1988.

[16] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns. US Patent Application 20100118123, 2010.

[17] http://www.ros.org/news/2010/12/technical-information-on-kinect-calibration.html, accessed 05.03.2012.

[18] http://azttm.wordpress.com/2011/04/03/, accessed 05.03.2012.

[19] A. Shpunt. Optical design for zero order reduction. US Patent Application 20110069389, 2011.

[20] http://www.ros.org/wiki/kinect_calibration/technical, 05.03.2012.

[21] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. *Proc. Euro. Conf. Comput. Vis.*, pages 151–158, 1994.

[22] G. Falcao, N. Hurtos, J. Massich, and D. Fofi. Projector-camera calibration toolbox. http://code.google.com/p/procamcalib/, 2009.

[23] http://www.ids-imaging.de, accessed 05.03.2012.

[24] J. Y. Bouguet. Camera calibration toolbox for MATLAB. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2008.

[25] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze. A fast stereo matching algorithm suitable for embedded real-time systems. *Computer Vision and Image Understanding*, 114:1180–1202, 2010.

[26] C. Zinner, M. Humenberger, K. Ambrosch, and W. Kubinger. An optimized software-based implementation of a census-based stereo matching algorithm. *Lecture Notes in Computer Science*, 5358:216–227, 2008.

[27] C. Zinner and M. Humenberger. *S3E Evaluation Pack HOWTO*, 2009.

[28] D. Lowe. Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, 2:1150–1157, 1999.

[29] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org, 2008.