NTNU
Norwegian University of
Science and Technology

# Passive depth estimation using stereo vision, an experimental study

**Jørgen Jøsok Høklie**

Master of Science in Cybernetics and Robotics
Submission date: June 2017
Supervisor: Annette Stahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

# Passive Depth Estimation Using Stereo Vision, an Experimental Study

**Jørgen Jøsok Høklie**

TTK4900 - Engineering Cybernetics, Master's Thessis
June 11, 2017
Supervisor: Annette Stahl, NTNU
Co-supervisor: Kristin Paulsen, Kongsberg Defence & Aerospace

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

## Student project

Stereo vision is a passive technique that gives a sense of depth based on two (or more) digital images taken from different positions. The technique can be expanded upon to retrieve full 3D scene information. Stereo vision techniques are highly applicable in a vast number of fields and products; from robot navigation and simultaneous localization and mapping to production, security, defense, exploration and entertainment applications.

### Part 1 (Prosjektoppgave: 15 studiepoeng)

**Title: Passive depth estimation using stereo vision, a theoretical study**

Compare the performance of different stereo vision techniques for retrieving depth information in different range regimes, e.g. meters, tens of meters, hundreds of meters. Stereo refers to multiple images being taken simultaneously from two cameras. Only passive techniques for depth estimation using stereo cameras should be explored in this study.

The text is largely based on 3D Imaging, Analysis and Applications by Pears et al.

The basic idea behind retrieving depth information using stereo vision with two cameras is to automatically detect the same feature in both images and use triangulation to find the distance to the feature. This means that a large portion of the problem is related to ensuring that the features recognized in the images are actually the same feature. Another challenge is that a feature visible in one image may be occluded or partially occluded in the other image, and features based on false corners[1] may mislead the algorithm.

In contrast to stereo vision, Structure from Motion (SfM) refers to a single moving camera scenario, where the sequence of images is used to gather depth and 3D information about the scene. In this project, SfM will not be used with a single camera, but is should be explored how the motion of a stereo vision platform can be used to yield additional confidence in the depth estimation.

The comparison study should be done using a MATLAB framework. The test images should be taken in ambient light scenarios. The comparison study should be done for each range regime separately. For the tests conducted in this study a suitable benchmark stereo image set should be used. The test stereo images should be of a static scene. Image series, each series of images should include a set of stereo images of the same scene taken at different positions, are required to test Structure from Motion (SfM) algorithms.

The MATLAB program should include:

- **Functionality for calibrating single camera**
  The calibration should include calibration of the cameras intrinsic parameters and a functionality to compensate for radial lens distortion effects.

[1] A false corner may result from the transition point where one object overlaps with an other object at a different position.

The appropriate functions in the camera calibration toolbox in MATLAB may be used.

- **Functionality for calibrating a stereo camera rig**
  The calibration should include calibration of each of the two cameras intrinsic parameters and a functionality to compensate for radial lens distortion effects. Additionally, it should include one set of extrinsic parameters that describes the relative rotation and translation between the two cameras.

  The appropriate functions in the camera calibration toolbox in MATLAB may be used.

- **Solve the correspondence problem**
  For a position **x** in the left image, which is the corresponding point **x'** in the right image, where **x** and **x'** are images of the same physical scene point?

  In this step the different techniques for finding the corresponding points in the two images should be tested.

- **Solve the reconstruction problem**
  Given the two corresponding points **x** and **x'**, how do we compute the 3D coordinates?

- **Analysis**
  The program should include an analysis tool that can compare the different stereo vision techniques.

**Milestones**

1. Completing the framework from comparison tests
2. Test stereo vision techniques for depth estimation
   a. Correlation based methods
      i. SSD (Sum of Squared Differences)
      ii. SAD (Sum of Absolute Differences)
      iii. NCC (Normalized Cross Correlation)
   b. Feature based methods
      i. Harris Corner Detector
      ii. SIFT (Scale Invariant Feature Transform)
      iii. SURF (Speeded Up Robust Feature)
3. Test SfM techniques in combination with stereo vision techniques. The same algorithms that are mentioned under point 2 should be tested with images taken in a time series.

## Part 2 (Masteroppgave: 30 studiepoeng)

**Title: Passive depth estimation using stereo vision, an experimental study**

The aim of this part of the study is to test and compare the implemented algorithms from part 1 in a practical application. The application is a case where a stereo camera is used to find the position of a platform relative to the stereo camera. The platform will consist of periodic structures.

The algorithms should be compared, and a subject of interest is to estimate the robustness of each algorithm under different conditions. The system should be tested under different weather conditions and under varying degrees of illumination.

Furthermore, the algorithms should be tested for different distances between the cameras, i.e. baseline lengths. A consideration of an optimal baseline length for the application should be made.

### Milestones

1. Create a test setup that includes:
    a. A stereo camera
    b. An alternative sensor system that can take accurate measurements of the distance between the test-rig and the target.
2. Calibrate the stereo camera, by calculating the intrinsic and extrinsic parameters
3. Create a database with test images and matching relative distance measurements
   The test set should include images taken under different weather condition and with varying degrees of illumination. Furthermore, different baseline lengths should be tested.
4. Compare the results from the different algorithms
5. Conclude on the best algorithm
6. Find the best baseline length for this application
7. Test the system in real time

## Alternative/bonus tasks

### Exploit the geometry of the environment

Use the geometry of the platform to set up a range interval that the scene points should be found in.

### Implement the algorithms using GPU

Graphic Processing Units (GPUs) are immensely powerful processors outperforming Central Processing Units (CPUs) in a variety of applications on supercomputers and PCs. The architecture of the GPU is highly parallel and tailored to efficiently construct images of 3D models. The GPUs are therefore highly suitable for implementations of stereo vision algorithms.

## Preface

The work presented in the following is the result of the course TTK4900 - Engineering Cybernetics, Master's Thesis. Carried out at the Norwegian University of Science and Technology, NTNU, Department of Engineering Cybernetics. The initiative for the thesis came from Kongsberg Defence & Aerospace.

The work has been time-demanding, but has rewarded me with new knowledge and experience. Working with a practical problem has been an extra motivation.

I would like to thank my supervisor Annette Stahl and co-supervisor Kristin Paulsen for their support and valuable advice throughout the work with the thesis.

Jørgen Jøsok Høklie
*Trondheim, June 11, 2017*

## Abstract

This master's thesis is an experimental study on passive stereo techniques for retrieving 3D scene information. The stereo camera system is tested for finding the position of a platform relative to the stereo camera where the measurements are used for navigating an autonomous docking system installed on offshore vessels.

The accuracy of passive stereo techniques depends on having precise knowledge of the cameras position, orientation and internal parameters. Calibrating these parameters is a key part of the success of the stereo camera. For a point in the scene imaged from two different known view points, the displacement of the point reprojection in the images are inversely proportional to depth and can be used to compute the 3D coordinate. The problem of establishing correspondences in the image pair is a difficult task and there exist several algorithms for solving the correspondence problem.

In this thesis, we have evaluated the accuracy of the stereo camera system for computing 3D information of the scene and proposed an optimal baseline length to improve accuracy. Six different of algorithms have been tested for solving the correspondence problem for this application. The algorithms considered are the correlation based methods: Sum of absolute differences, square sum of differences and normalized cross correlation. Together with the feature based methods: Harris corner detector, scale invariant feature transform and speeded up robust features. The performance of each algorithm is compared through a field experiment to conclude on the best one.

Based on the results of the experiment, the correlation based methods proves to be best suited for this application where the sum of squared differences is considered as the superior algorithm for solving the correspondence problem.

**Keywords**: Camera Calibration, Stereo Camera, Rectification, Triangulation, Correspondence Problem, Epipolar Geometry, Correlation Based Methods, Feature Based Methods.

# Contents

# 1 Introduction

## 1.1 Motivation

Kongsberg Gruppen is developing new an autonomous bridge system. This system is planned to be installed on ships or other offshore vessels with the purpose of getting personnel safely on and off a windmill, oil rig or other offshore installations. To be able to steer the bridge to the desired location the control system needs a measurement of where in space the docking slot is relative to the bridge. Stereo cameras have been widely used for autonomous robots to provide accurate depth information of the surroundings. Kongsberg wants to see the potential of a passive stereo camera system being used for this application and have formulated the problem description for this thesis which partly builds on the student project.

The problem of getting depth information from the scene with a stereo camera can be divided into two main subjects, namely, photogrammetry and the correspondence problem. Photogrammetry involves topics as projective geometry and optics with the purpose of creating a camera model that can convert corresponding image points into three-dimensional world coordinates. To be able to find correspondences between the two images taken by the stereo camera we need to solve the correspondence problem. By use of correlation and feature based computer vision algorithms we can be able to find matching points between two images. Solving the correspondence problem is often difficult and successful matching may depend on light conditions, texture of objects in the image etc. We often get ambiguous matches and image points within regions of homogeneous intensity is hard to match. The main focus in this thesis will be to create a precise camera model and compare different computer vision algorithms for solving the correspondence problem and look at the accuracy of the estimated three dimensional world coordinate.

## 1.2 Objective and Scope

This master thesis will investigate the performance of a stereo camera rig with the objective of supplying the control system of the bridge with accurate measurements of the position of the docking slot. The following sub tasks are defined for the thesis:

- Present necessary theory for creating a camera model and different computer vision algorithms for solving the correspondence problem and measure depth.

- Create a test setup and a database containing test images with accurate measurements of distance between camera and target.
- Compare the performance of the different algorithms and conclude on the best one.
- Find an optimal length for the baseline between the cameras to improve accuracy of the world coordinate.

Some assumptions are made to simplify the problem:

- The system is not supposed to detect the docking slot. It is assumed that a point of interest in one of the images is provided by a tracker. Only the problem of finding the corresponding point given by the tracker in the other image is investigated.
- The docking slot is always visible in both images.
- Noiseless measurements, i.e. no blur in the images.
- We have an estimate, not precise, of the depth to the docking slot relative to the camera.

## 1.3 Contributions

The contributions of this thesis is the following:

- A review of topics in the field of photogrammetry such as the pinhole camera model, planar camera calibration, epipolar geometry, rectification and triangulation.
- Analysis of the different computer vision algorithms used for solving the correspondence problem. The algorithms considered are the correlation based methods: Sum of absolute differences(SAD), sum of squared differences(SSD) and normalized cross correlation(NCC). Further the feature based methods: Harris corner detector(HCD), speeded up robust features(SURF) and scale invariant feature transform(SIFT).
- Comparison of the different algorithms performance, through a field experiment.

## 1.4 Outline

Chapter 2 presents the mathematical background and theory for different subjects in photogrammetry. Chapter 3 presents the approach for solving the correspondence problem and the theory behind each algorithm. Chapter 4 includes a description of the field experiment and the presentation of the experimental results followed by a discussion. Chapter 5 conclusion and proposed further work.

## 1.5 Notation and Definitions

- The set of all real numbers is denoted $\mathbb{R}$
- All vectors in this report are column vectors. $\mathbb{R}^n$ denotes the set of $n \times 1$ real vectors. The set of real matrices with $n$ rows and $m$ columns are denoted $\mathbb{R}^{n \times m}$. All vectors and matrices are given in bold font. $x \in \mathbb{R}$ is a scalar and $\boldsymbol{x} \in \mathbb{R}^n$ is a vector.
- The transpose of $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ is denoted $\boldsymbol{A}^T \in \mathbb{R}^{m \times n}$.
- The inverse of a invertible matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is denoted $\boldsymbol{A}^{-1} \in \mathbb{R}^{n \times n}$
- The notation $\text{diag}(x_1, x_2, \ldots, x_n)$ is used to denote the diagonal matrix:

$$\begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{1.1}$$

- The determinant of a square matrix $\boldsymbol{A}$ is denoted $\text{Det}(A)$.
- The notation $\text{Tr}(\boldsymbol{A})$ denotes the sum of the diagonal elements of an square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$:

$$\text{Tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii} \tag{1.2}$$

- The Euclidean norm of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is denoted $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$
- The Forbenius norm of an square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is denoted $\|\boldsymbol{A}\|_F = \sqrt{\text{Tr}(\boldsymbol{A^T A})}$

- The cross product operator $\boldsymbol{S}(\cdot)$ is defined such that

$$\boldsymbol{t} \times \boldsymbol{x} = \boldsymbol{S_t}\boldsymbol{x}, \quad \boldsymbol{t}, \boldsymbol{x} \in \mathbb{R}^3 \tag{1.3}$$

  where $\boldsymbol{S_t}$ is the skew-symmetric matrix

$$\boldsymbol{S_t} = -\boldsymbol{S_t}^T = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

- The instrinsic matrix is denoted $\boldsymbol{K} \in \mathbb{R}^{3\times3}$
- The origin of the camera center in world coordinates is denoted $\boldsymbol{O} \in \mathbb{R}^{3\times3}$
- A $n \times n$ identity matrix is denoted $\boldsymbol{I}_n$ and $\boldsymbol{1}_n$ is a $n \times 1$ column vector of ones.
- The convolution operation is denoted $*$. The convolution of two $\mathbb{R}^{2\times2}$ matrices is given by:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = (d * 1) + (c * 2) + (b * 3) + (a * 4) \tag{1.4}$$

# 2 Photogrammetry

This chapter covers different topics in the field of photogrammetry. First we will define the camera model used in this thesis. The camera model describes the mathematical relation between a three dimensional point and its projection onto the image sensor. The unknown parameters of the camera model is estimated by performing a camera calibration. A complete description of the planar-based calibration method is included in this chapter. Finally we will present necessary theory around stereo camera geometry, to be able to calculate depth and simplify the correspondence problem.

## 2.1 Camera Modeling

The model used in this thesis is the pinhole camera model. Where $\boldsymbol{O}_c$ is the position of the camera model in world coordinates, also called the center of projection. A vector from $\boldsymbol{O}_c$ to $\boldsymbol{X}$ intersects with the image plane, where $\boldsymbol{X}$ is a world point. For some point on the image plane, $\boldsymbol{x}_c$, the corresponding scene point, $\boldsymbol{X}$, must lie somewhere on the infinite ray connecting $\boldsymbol{O}_c$ and $\boldsymbol{x}_c$ as shown in figure 2.1. The theory in this chapter is based on work done by Stephen Se & Nick Pears [15].

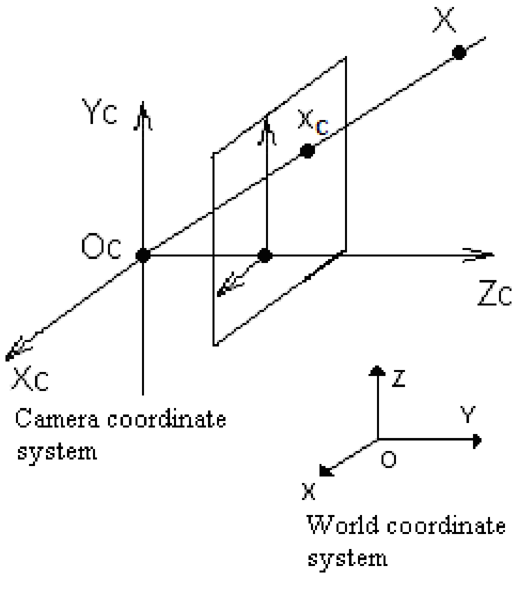


Figure 2.1: The pinhole camera model [12].

### 2.1.1 Coordinate Transformation

A transformation from the world coordinate system to the camera coordinate system can be described by a rigid transformation. The world coordinate of the center of projection is denoted $\boldsymbol{O}_c$. The rotation of the camera frame relative to the world frame is described by $\boldsymbol{R}$. The coordinate transformation from a point $\boldsymbol{X}$ in world coordinates to the camera coordinate system in inhomogeneous coordinates can be written as:

$$\boldsymbol{X}_c = \boldsymbol{R}(\boldsymbol{X} - \boldsymbol{O}_c) \tag{2.1}$$

$$\boldsymbol{X}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where $\boldsymbol{X}_c \in \mathbb{R}^3$ is the representation of $\boldsymbol{X}$ in the camera coordinate system. We can transform equation 2.1 to homogeneous coordinates follows:

$$\begin{bmatrix} \boldsymbol{X}_c \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_3 & -\boldsymbol{O}_c \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R}\boldsymbol{O}_c \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} \tag{2.2}$$

where the rigid translation is given by:

$$\boldsymbol{t} = -\boldsymbol{R}\boldsymbol{O}_c \tag{2.3}$$

Finally we denote $P_r$ as the matrix representing the rigid coordinate transformation:

$$\boldsymbol{P}_r = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{2.4}$$

### 2.1.2 Perspective Projection

The perspective projection is the mapping of a point in camera coordinates, $\boldsymbol{X}_c$, to a point in the image plane, $\boldsymbol{x}_c$. We define the image plane axes to be parallel to the camera coordinate system. This mapping is called an ideal perspective projection since we assume the image plane to be flat i.e. we assume that the lens is free from distortion. We define the image plane to lie $f$ metric units from $\boldsymbol{O}_c$ along the $Z_c$-axis. Where $f$ is usually set to be the focal length of the camera. The origin of the image plane is called the principal point. We define the principal point as:

$$\boldsymbol{x}_p = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \tag{2.5}$$

The set of all points that lies on the image plane is given by:

$$\{\boldsymbol{x}_c \in \mathbb{R}^3 \,|\, \boldsymbol{n}^T(\boldsymbol{x}_c - \boldsymbol{x}_p) = 0\} \tag{2.6}$$

where $\boldsymbol{n} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ defined in the camera coordinate system is perpendicular to the image plane. From figure 2.2 the image point $\boldsymbol{x}_c$ and camera coordinate $\boldsymbol{X}_c$ is projected onto the $X_cZ_c$ and $Y_cZ_c$ plane.

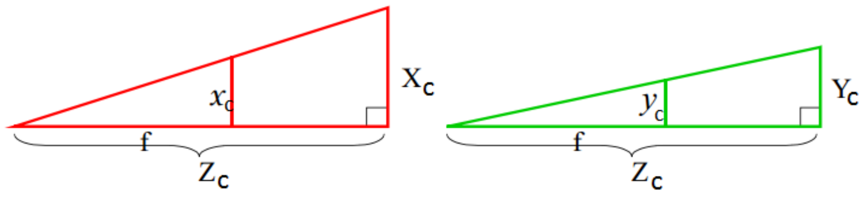


Figure 2.2: Similar triangles in the $X_cZ_c$ and $Y_cZ_c$ plane [4].

By the rule of similar triangles we get the perspective projection equations:

$$\frac{x_c}{f} = \frac{X_c}{Z_c}, \quad \frac{y_c}{f} = \frac{Y_c}{Z_c}. \tag{2.7}$$

We can write the similarities in linear form:

$$Z_c \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \tag{2.8}$$

The perspective projection matrix $\boldsymbol{P}_p$ is defined as:

$$\boldsymbol{P}_p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times4}. \tag{2.9}$$

### 2.1.3 Image Sampling

The image plane is sampled by the image sensor of the camera, shown in figure 2.3. This is a mapping from $\begin{bmatrix} x_c & y_c \end{bmatrix}^T$ to the corresponding pixel position $\begin{bmatrix} x & y \end{bmatrix}^T$ in the affine coordinate system of the cameras image sensor. The image

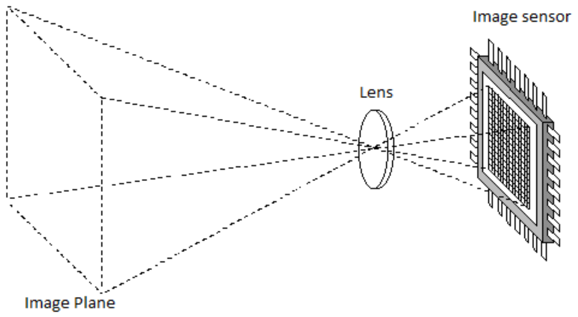


Figure 2.3: The mapping from the image plane to the affine image sensor [1].

sensor is usually not square so the number of pixels per unit distance may differ in the $x_c$ and $y_c$ directions. We call these ratios $m_x$ and $m_y$. Typically for the pixel coordinate system, the origin is placed at the top left corner of the camera sensor. To align the coordinate systems we model the position of the principal point in the image sensor with the pixel coordinate $\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$. To compensate for any skew in the image sensor we include $s$ as the skew coefficient in the camera model and we end up with the linear mapping:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_x & s & x_0 \\ 0 & m_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{2.10}$$

The homography $\boldsymbol{P}_c$ of the ideal image point on the image plane to the affine coordinate system of the camera sensor is defined as:

$$\boldsymbol{P}_c = \begin{bmatrix} m_x & s & x_0 \\ 0 & m_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{2.11}$$

### 2.1.4 Projection Matrix

The projection matrix describes the transformation from a world point $\boldsymbol{X}$ to a pixel position in the sensor frame. We can derive the projection matrix from the transformation matrices defined in the previous sections.

$$\boldsymbol{x} = \boldsymbol{P}_c \boldsymbol{x}_c, \quad Z_c \boldsymbol{x}_c = \boldsymbol{P}_p \boldsymbol{X}_c, \quad \boldsymbol{X}_c = \boldsymbol{P}_r \boldsymbol{X} \tag{2.12}$$

where

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{x}_c = \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix}, \quad \boldsymbol{X}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

By multiplying the transformation matrices in the the right order we get the transformation from $\boldsymbol{X}$ to $\boldsymbol{x}$, resulting in the projection matrix $\boldsymbol{P}$.

$$\lambda \boldsymbol{x} = \boldsymbol{P}_c \boldsymbol{P}_p \boldsymbol{P}_r \boldsymbol{X} = \boldsymbol{P}\boldsymbol{X}, \quad 0 < \lambda \in \mathbb{R} \tag{2.13}$$

All world points on an infinite ray from the camera coordinates systems origin maps to the the same pixel position, as mentioned in the introduction of this chapter. We say that the projection matrix is defined up to scale since any scaling of the projection matrix preforms the same projection.

$$\boldsymbol{x} \propto \boldsymbol{P}\boldsymbol{X} \tag{2.14}$$

The scalar $\lambda$ is equal to the imaged point depth in the camera coordinate system, $\lambda = Z_c$. This is a result of the perspective projection, $\boldsymbol{P}_p$. The projection matrix $\boldsymbol{P}$ can be factored into two matrices, namely the intrinsic and the extrinsic matrix in the following way:

$$\boldsymbol{P} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \in \mathbb{R}^{3\times 4}. \tag{2.15}$$

The parameters within $\boldsymbol{K}$ is called the cameras intrinsic parameters. It represents the fundamental camera characteristics and depends only on the camera. Where $\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$ describe the position and orientation of the camera in the world frame and is termed the cameras extrinsic parameters. The intrinsic matrix $\boldsymbol{K}$

is defined as:

$$\boldsymbol{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

where

$$\alpha_x = fm_x, \quad \alpha_y = fm_y.$$

The extrinsic matrix is given by:

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}. \tag{2.17}$$

By substituting 2.15 into 2.13 we get the relation between a 3D point $\boldsymbol{X}$ and its image projection $\boldsymbol{x}$ given by the intrinsic and extrinsic matrix:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.18}$$

### 2.1.5 Radial Distortion

Until now we have only discussed the direct linear transformation from world coordinates to the image sensor. There are several reasons for nonlinear errors in the camera model, one of them is caused by misalignment of the optics in the camera. The degree of distortion will vary throughout the image plane and will cause an individual shift for each pixel. The distortion may in general be irregular, but the most commonly encountered distortions are radially symmetrical. We define $\Delta$ as the shift function of each pixel in the $x$ and $y$ direction. The shift function depends on the position of the projected point $\boldsymbol{X}$ onto the image plane and image sensor, and how the the nonlinear distortion is modeled. We will define the shift function later in this section, for now we have:

$$u = x + \Delta x(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \tag{2.19}$$

$$v = y + \Delta y(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \tag{2.20}$$

where $\boldsymbol{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ is the linear projection of the worldpoint $\boldsymbol{X}$ given by equation 2.18. The detected point of $\boldsymbol{X}$ on the image sensor is the pixel coordinate $\boldsymbol{u} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$. The mapping between the detected point $\boldsymbol{u}$ and the projected point $\boldsymbol{x}$ can be written as:

$$\boldsymbol{u} = \boldsymbol{A}(\tilde{\boldsymbol{x}}, \boldsymbol{k})\boldsymbol{x} \tag{2.21}$$

where

$$\boldsymbol{A}(\tilde{\boldsymbol{x}}, \boldsymbol{k}) = \begin{bmatrix} 1 & 0 & \Delta x(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \\ 0 & 1 & \Delta y(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \\ 0 & 0 & 1 \end{bmatrix}$$

The general calibration matrix can be obtained by combining $\boldsymbol{K}$ with the mapping 2.21.

$$\boldsymbol{K}(\tilde{\boldsymbol{x}}, \boldsymbol{k}) = \boldsymbol{A}(\tilde{\boldsymbol{x}}, \boldsymbol{k})\boldsymbol{K} = \begin{bmatrix} \alpha_x & s & x_0 + \Delta x(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \\ 0 & \alpha_y & y_0 + \Delta y(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \\ 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

There are several ways to model the nonlinear errors, this report will use the distortion model presented in [19]. It is likely that the distortion function is dominated by radial components as mentioned earlier. The model only consider the first two terms of radial distortion since the distortion function is dominated by these first terms. The distortion model is given by:

$$u = x + (x - x_0)(k_1 r^2 + k_2 r^4) = x + \Delta x(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \tag{2.23}$$

$$v = y + (y - y_0)(k_1 r^2 + k_2 r^4) = y + \Delta y(\tilde{\boldsymbol{x}}, \boldsymbol{k}) \tag{2.24}$$

where $k_1$ and $k_2$ are the radial distortion coefficients. The radial term $r$ is the distance from the principal point, $\boldsymbol{x}_p = \begin{bmatrix} 0 & 0 & f \end{bmatrix}^T$ to the projection of $\boldsymbol{X}$ in the image plane, $\boldsymbol{x}_c = \begin{bmatrix} x_c & y_c & f \end{bmatrix}^T$.

$$r = \|\boldsymbol{x}_c - \boldsymbol{x}_p\| = \sqrt{x_c^2 + y_c^2} \tag{2.25}$$

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} x_c \\ y_c \\ x \\ y \\ x_0 \\ y_0 \end{bmatrix}, \quad \boldsymbol{k} = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

## 2.2 Camera Calibration

This chapter covers a general method for calibrating a single camera. The goal is to estimate the unknown parameters in the pin hole camera model presented in the previous chapter. First we will go through how we can estimate the linear part of the model containing the intrinsic and extrinsic matrix. Then we will estimate the radial distortion coefficients in the nonlinear part of the model. Finally we will show the results of a single camera calibration. The theory in this chapter in based on work done by Z. Zhang and Stephen Se & Nick Pears [19, 15].

### 2.2.1 Plane Based Calibration

With this method the idea is to present the camera for a set of pictures of a planar surface. Most common is to use a checkerboard as the planar surface. It is important that either the camera or the checkerboard is fixed in space. The camera characteristics are the same for all pictures, only the orientation and position, $\boldsymbol{R}$ and $\boldsymbol{t}$, of the camera are changing. By knowing the size and structure of the 2D pattern, in this case the the checkerboard, we can define a coordinate system on the checkerboard that describes where each corner is in the real world. We define origin of the world coordinate system as the upper left corner of the plane, the directions of $X$ and $Y$ moves right and down respectively and $Z$ is orthogonal to the plane. To make correspondences between the known world points and the image points it is common to use a corner detector to identify the corners of each square in the image and then pair each corner to the corresponding world point. Since the origin of the world coordinates are set to be at the top left corner of the checkerboard, the plane are fixed at $Z = 0$. This means that any point on the plane have the coordinates:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{2.26}$$

By looking at the columns of $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}$ we can simplify 2.26

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.27}$$

Where $\boldsymbol{H} \in \mathbb{R}^{3\times3}$ is the planar homography that maps points on the checkerboard to the corresponding pixel points in the image.

### 2.2.2 Homography Estimation

The camera calibration technique relies on a homography estimation. The homography matrix is denoted $\boldsymbol{H}$. From equation 2.27 and by writing $\boldsymbol{H}$ by its elements, we get:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{2.28}$$

By dividing by $\lambda$ we map homogeneous coordinates to inhomogeneous coordinates and we end up with

$$x = \frac{h_{11}X + h_{12}Y + h_{13}}{h_{31}X + h_{32}Y + h_{33}} \tag{2.29}$$

$$y = \frac{h_{21}X + h_{22}Y + h_{23}}{h_{31}X + h_{32}Y + h_{33}} \tag{2.30}$$

where

$$\lambda = h_{31}X + h_{32}Y + h_{33}$$

We want to solve for $\boldsymbol{H}$. By rearranging equations 2.29 and 2.30 we get:

$$\boldsymbol{a}_x^T \boldsymbol{h} = 0 \tag{2.31}$$

$$\boldsymbol{a}_y^T \boldsymbol{h} = 0 \tag{2.32}$$

where

$$\boldsymbol{h} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{bmatrix}^T$$

$$\boldsymbol{a}_x = \begin{bmatrix} -X & -Y & -1 & 0 & 0 & 0 & xX & xY & x \end{bmatrix}^T$$

$$\boldsymbol{a}_y = \begin{bmatrix} 0 & 0 & 0 & -X & -Y & -1 & yX & yY & y \end{bmatrix}^T$$

Given a set of corresponding points we can form the following linear system of equations:

$$\boldsymbol{A}\boldsymbol{h} = 0, \quad \boldsymbol{h} \neq 0 \tag{2.33}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_{x1}^T \\ \boldsymbol{a}_{y1}^T \\ \vdots \\ \boldsymbol{a}_{xm}^T \\ \boldsymbol{a}_{ym}^T \end{bmatrix}, \quad 4 \leq m$$

$m$ is the number of correspondences. The homography matrix has 8 degrees of freedom because it is defined up to scale. Therefore we need at least 4 correspondences that results in 8 linear equations. We can solve this using linear least squares method and properties of the singular value decomposition. A description of the singular value decomposition can be found in the appendix 6.2. The sum of the squared error can be written as:

$$f(\boldsymbol{h}) = \frac{1}{2}\boldsymbol{h}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{h} \tag{2.34}$$

$$\frac{df}{d\boldsymbol{h}} = 0 = \frac{1}{2}(\boldsymbol{A}^T\boldsymbol{A} + (\boldsymbol{A}^T\boldsymbol{A})^T)\boldsymbol{h} \tag{2.35}$$

$$0 = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{h}. \tag{2.36}$$

The singular value decomposition of $\boldsymbol{A}$ is given by:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T. \tag{2.37}$$

The solution for $\boldsymbol{h}$ is given by the right singular vector of $\boldsymbol{A}$ that corresponds to the smallest singular value.

### 2.2.3 Intrinsic Matrix

Now that we know the homography matrix we can continue to calculate the intrinsic matrix $\boldsymbol{K}$. For every camera position we have a projective transformation.

$$\boldsymbol{H} = \frac{\boldsymbol{K}}{\lambda_H}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix} \tag{2.38}$$

By writing 2.38 in column form

$$\lambda_H \boldsymbol{K}^{-1} \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \boldsymbol{h}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \tag{2.39}$$

we get 3 equations

$$\lambda_H \boldsymbol{K}^{-1} \boldsymbol{h}_1 = \boldsymbol{r}_1 \tag{2.40}$$

$$\lambda_H \boldsymbol{K}^{-1} \boldsymbol{h}_2 = \boldsymbol{r}_2 \tag{2.41}$$

$$\lambda_H \boldsymbol{K}^{-1} \boldsymbol{h}_3 = \boldsymbol{t} \tag{2.42}$$

Since $\boldsymbol{R}$ is a rotation matrix we know that the columns are unit vectors and orthogonal to each other.

$$\|\boldsymbol{r}_1\| = 1, \quad (\boldsymbol{r}_1)^T \boldsymbol{r}_2 = 0 \tag{2.43}$$

By using these properties of the rotation matrix we get two constraints to the problem.

$$\boldsymbol{h}_1^T (\boldsymbol{K}\boldsymbol{K}^T)^{-1} \boldsymbol{h}_2 = 0 \tag{2.44}$$

$$\boldsymbol{h}_1^T (\boldsymbol{K}\boldsymbol{K}^T)^{-1} \boldsymbol{h}_1 = \boldsymbol{h}_2^T (\boldsymbol{K}\boldsymbol{K}^T)^{-1} \boldsymbol{h}_2 \tag{2.45}$$

We define the matrix $\boldsymbol{B}$ as

$$\boldsymbol{B} = (\boldsymbol{K}\boldsymbol{K}^T)^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{2.46}$$

Note that $\boldsymbol{B}$ is symmetric, we define a 6D vector containing all elements of $\boldsymbol{B}$.

$$\boldsymbol{b} = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}^T \tag{2.47}$$

Let the $i^{th}$ column of the homography matrix, $\boldsymbol{H}$, be

$$\boldsymbol{h}_i = \begin{bmatrix} h_{1i} & h_{2i} & h_{3i} \end{bmatrix}^T \tag{2.48}$$

Equation 2.44 and 2.45 can be rewritten to

$$\boldsymbol{h}_i^T \boldsymbol{B} \boldsymbol{h}_j = \boldsymbol{w}_{ij}^T \boldsymbol{b} \tag{2.49}$$

$$\boldsymbol{h}_i^T \boldsymbol{B} \boldsymbol{h}_i = \boldsymbol{h}_j^T \boldsymbol{B} \boldsymbol{h}_j = \boldsymbol{w}_{ii}^T \boldsymbol{b} = \boldsymbol{w}_{jj}^T \boldsymbol{b} \tag{2.50}$$

where

$$\boldsymbol{w}_{ij} = \begin{bmatrix} h_{1i}h_{1j} & h_{1i}h_{2j} + h_{2i}h_{1j} & h_{2i}h_{2j} & h_{3i}h_{1j} + h_{1i}h_{3j} & h_{3i}h_{2j} + h_{2i}h_{3j} & h_{3i}h_{3j} \end{bmatrix}^T$$

The two constraints can now be written as two homogeneous equations

$$\begin{bmatrix} \boldsymbol{w}_{12}^T \\ \boldsymbol{w}_{11}^T - \boldsymbol{w}_{22}^T \end{bmatrix} \boldsymbol{b} = 0 \tag{2.51}$$

Note that we have six unknowns, each homography matrix only produce two equations so we need at least three test photos of the checkerboard. If $n$ photos from different positions have been taken, we can stack equations as 2.51 on top of each other forming matrix $\boldsymbol{W}$.

$$\boldsymbol{W}\boldsymbol{b} = 0, \quad \boldsymbol{W} \in \mathbb{R}^{2n \times 6} \tag{2.52}$$

If $3 \leq n$ we will in general have a unique solution for $\boldsymbol{b}$. We solve this the same way as for the homography matrix. It is recommended to have multiple views $3 < n$ and take the least square solution of

$$\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{b} = 0 \tag{2.53}$$

Where the solution to $\boldsymbol{b}$ is the right singular vector of $\boldsymbol{W}$ that corresponds to the smallest singular value. When $\boldsymbol{b}$ is known, we can solve for $\boldsymbol{K}$. Note that $\boldsymbol{B}$ is estimated up to an unknown scale factor $\lambda_B$.

$$\boldsymbol{B} = \lambda_B (\boldsymbol{K}\boldsymbol{K}^T)^{-1} \tag{2.54}$$

$$= \lambda_B \begin{bmatrix} \frac{1}{\alpha_x^2} & -\frac{s}{\alpha_x^2 m_y} & \frac{y_0 s - x_0 \alpha_y}{\alpha_x^2 \alpha_y} \\ -\frac{s}{\alpha_x^2 \alpha_y} & \frac{s^2}{\alpha_x^2 \alpha_y^2} + \frac{1}{\alpha_y^2} & -\frac{s(y_0 s - x_0 \alpha_y)}{\alpha_x^2 \alpha_y^2} - \frac{y_0}{\alpha_y^2} \\ \frac{y_0 s - x_0 \alpha_y}{\alpha_x^2 \alpha_y} & -\frac{s(y_0 s - x_0 \alpha_y)}{\alpha_x^2 \alpha_y^2} - \frac{y_0}{\alpha_y^2} & \frac{(y_0 s - x_0 \alpha_y)^2}{\alpha_x^2 \alpha_y^2} + \frac{y_0^2}{\alpha_y^2} + 1 \end{bmatrix} \tag{2.55}$$

It is possible to uniquely extract the intrinsic parameters from matrix $\boldsymbol{B}$. Solution taken from Z. Zhang [19].

$$y_0 = \frac{B_{12}B_{13} - B_{11}B_{13}}{B_{11}B_{22} - B_{12}^2} \tag{2.56}$$

$$\lambda_B = B_{33} - \frac{B_{12}^2 + y_0(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \tag{2.57}$$

$$\alpha_x = \sqrt{\frac{\lambda_B}{B_{11}}} \tag{2.58}$$

$$\alpha_y = \sqrt{\frac{\lambda_B B_{11}}{B_{11}B_{22} - B_{12}^2}} \tag{2.59}$$

$$s = \frac{B_{12}\alpha_x^2\alpha_y}{\lambda_B} \tag{2.60}$$

$$x_0 = \frac{sy_0}{\alpha_y} - \frac{B_{13}\alpha_x^2}{\lambda_B} \tag{2.61}$$

By substituting the intrinsic parameters from equations 2.56-36 into the definition of the intrinsic matrix, 2.16, we get the numerical values for the elements of $\boldsymbol{K}$.

### 2.2.4 Extrinsic Matrix

When $\boldsymbol{K}$ is known we can compute the extrinsic parameters for each homography $\boldsymbol{H}$. From 3.15-17 we get

$$\begin{aligned}
\boldsymbol{r}_1 &= \lambda_H \boldsymbol{K}^{-1}\boldsymbol{h}_1 \\
\boldsymbol{r}_2 &= \lambda_H \boldsymbol{K}^{-1}\boldsymbol{h}_2 \\
\boldsymbol{r}_3 &= \boldsymbol{r}_1 \times \boldsymbol{r_2} \\
\boldsymbol{t} &= \lambda_H \boldsymbol{K}^{-1}\boldsymbol{h}_3
\end{aligned}$$

where

$$\lambda_H = \frac{1}{\|\boldsymbol{K}^{-1}\boldsymbol{h_1}\|} = \frac{1}{\|\boldsymbol{K}^{-1}\boldsymbol{h_2}\|}$$

The extrinsic matrix is given by:

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t} \end{bmatrix} \tag{2.62}$$

### 2.2.5 Rotation Matrix Approximation

The estimated $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ when calculating the extrinsic parameters are not necessarily orthogonal because of noise in the data. Therefore we need to estimate $\boldsymbol{R}$ such that it satisfy the properties of a rotation matrix. This estimation method is taken from Z. Zhang [19]. Let

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix} \tag{2.63}$$

We want to find the best $\boldsymbol{R}$ to approximate $\boldsymbol{Q}$. This can be done by minimizing the Frobenius norm of the difference $\boldsymbol{R} - \boldsymbol{Q}$.

$$\min_{\boldsymbol{R}} \|\boldsymbol{R} - \boldsymbol{Q}\|_F^2 \quad \text{subjected to} \quad \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I} \tag{2.64}$$

$$\begin{aligned} \|\boldsymbol{R} - \boldsymbol{Q}\|_F^2 &= \operatorname{Tr}((\boldsymbol{R} - \boldsymbol{Q})^T(\boldsymbol{R} - \boldsymbol{Q})) \\ &= \operatorname{Tr}(\boldsymbol{R}^T\boldsymbol{R}) + \operatorname{Tr}(\boldsymbol{Q}^T\boldsymbol{Q}) - 2\operatorname{Tr}(\boldsymbol{R}^T\boldsymbol{Q}) \\ &= 3 + \operatorname{Tr}(\boldsymbol{Q}^T\boldsymbol{Q}) - 2\operatorname{Tr}(\boldsymbol{R}^T\boldsymbol{Q}) \end{aligned}$$

The minimization problem 2.64 is equivalent to maximizing $\operatorname{Tr}(\boldsymbol{R}^T\boldsymbol{Q})$. By taking the singular value decomposition of $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, where $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)$.

$$\operatorname{Tr}(\boldsymbol{R}^T\boldsymbol{Q}) = \operatorname{Tr}(\boldsymbol{R}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = \operatorname{Tr}(\boldsymbol{V}^T\boldsymbol{R}^T\boldsymbol{U}\boldsymbol{\Sigma}) \tag{2.65}$$

Note that the manipulation done to 2.65 is allowed since the trace is invariant under cyclic permutations. We now define the orthogonal matrix $\boldsymbol{Z} = \boldsymbol{V}^T\boldsymbol{R}^T\boldsymbol{U}$ and substitute into the 2.65.

$$\operatorname{Tr}(\boldsymbol{V}^T\boldsymbol{R}^T\boldsymbol{U}\boldsymbol{\Sigma}) = \operatorname{Tr}(\boldsymbol{Z}\boldsymbol{\Sigma}) = \sum_{i=1}^{3} z_{ii}\sigma_i \leq \sum_{i=1}^{3} \sigma_i \tag{2.66}$$

It is easy to see that the optimal solution is $\boldsymbol{Z} = \boldsymbol{I}$, then $\boldsymbol{R}$ is given by:

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T \tag{2.67}$$

which is the solution of 2.64.

### 2.2.6 Radial Distortion Coefficients Estimation

The last step of the calibration is to estimate the radial distortion coefficients. We start of with writing equation 2.23 and 2.24 on linear form

$$\begin{bmatrix} u - x \\ v - y \end{bmatrix} = \begin{bmatrix} (x - x_0)r^2 & (x - x_0)r^4 \\ (y - y_0)r^2 & (y - y_0)r^4 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{2.68}$$

For $n$ test photos and $m$ points that have a known correspondence we get $2nm$ equations which we can stack together forming matrix $D$ and vector $d$ such that

$$\boldsymbol{D}\boldsymbol{k} = \boldsymbol{d}, \quad \boldsymbol{D} \in \mathbb{R}^{2nm \times 2}, \quad \boldsymbol{d} \in \mathbb{R}^{2mn} \tag{2.69}$$

The linear least square solution for $\boldsymbol{k}$ is given by:

$$\boldsymbol{k} = (\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T\boldsymbol{d}. \tag{2.70}$$

### 2.2.7 Error Minimization

All camera parameters in the pinhole model can be estimated based on the methods presented in the previous sub chapters. When it was solved, we used $n$ test photos and $m$ correspondences in each image. We want to refine the parameters in the model by minimizing the reprojection error. The reprojection error is the difference between the pixel point of the detected world point $\boldsymbol{X}$ in the image sensor and the calculated pixel point of $\boldsymbol{X}$ using the pinhole camera model. We refine the parameters in the model by minimizing the non-linear least squares problem:

$$\underset{(\boldsymbol{K},k_1,k_2,\boldsymbol{R}_i,\boldsymbol{t}_i)}{\operatorname{argmin}} \sum_{i=1}^{n}\sum_{j=1}^{m} \|\boldsymbol{x}_{ij} - \hat{\boldsymbol{x}}_{ij}(\boldsymbol{K}, k_1, k_2, \boldsymbol{R}_i, \boldsymbol{t}_i, \boldsymbol{X}_j)\|^2 \tag{2.71}$$

where $\boldsymbol{x}_{ij} = \begin{bmatrix} u_{ij} & v_{ij} & 1 \end{bmatrix}^T$ is the detected point of $\boldsymbol{X}_j$ in the image sensor from test photo $i$ and $\hat{\boldsymbol{x}}_{ij}$ is the calculated image point of world point $\boldsymbol{X}_j$ using equation 2.27 combined with 2.21.

$$\hat{\boldsymbol{x}}_{ij} = \boldsymbol{A}(\tilde{\boldsymbol{x}}, \boldsymbol{k})\boldsymbol{K}\begin{bmatrix} \boldsymbol{R}_i & \boldsymbol{t}_i \end{bmatrix}\boldsymbol{X}_j \tag{2.72}$$

The non-linear least squares problem can be solved by using Levenberg-Marquardt algorithm. This is an iterative procedure and it need to have an initial guess for the parameters to begin with. The algorithm will converge to global minimum if the initial guess is sufficiently close to the final minimum. This is also a way to estimate radial distortion if not done previously, simply by setting the initial guess for $k_1$ and $k_2$ equal to zero. The reader is referred to [6] for more details about the Levenberg-Marquardt algorithm.

## 2.3 Stereo Camera Geometry

In this section we will look into the calibration, and some geometry of a stereo camera rig. First we want to calibrate the stereo rig to be able to calculate the 3D world coordinate of a point that corresponds to corresponding image points from the two cameras. By exploiting the geometry of the stereo rig we can simplify the correspondence problem. It is assumed that the cameras in the stereo camera rig are individually calibrated and that distortions are corrected for. This means that linear projections between world coordinates and pixel points can be used.

### 2.3.1 Calibration of a Stereo Rig

We have two cameras mounted on the same rig, both fixed in space relative to each other as shown in figure 2.4. It is important to mention that both cameras must take pictures of the scene simultaneously. The whole checkerboard must be visible for both cameras when taking pictures for the calibration. The sets of all corresponding points for the left and right camera are defined as:

$$\boldsymbol{x}'_{ij} = \boldsymbol{K}' \begin{bmatrix} \boldsymbol{R}'_i & \boldsymbol{t}'_i \end{bmatrix} \boldsymbol{X}_j, \quad \boldsymbol{x}''_{ij} = \boldsymbol{K}'' \begin{bmatrix} \boldsymbol{R}''_i & \boldsymbol{t}''_i \end{bmatrix} \boldsymbol{X}_j \quad i,j \in n,m \tag{2.73}$$

where $m$ is the number of test photos and $n$ is the number of correspondences. $'$ and $''$ denotes all the parameters and image points for the left and right camera respectively. From figure 2.4 you can see the geometry of the stereo camera rig.

### 2.3.2 Estimation of Rigid Transformation

We want to find the rigid transformation, $\boldsymbol{R}$ and $\boldsymbol{t}$, of the right camera relative to the left camera. This information is necessary later when rectifying the images and estimating depth of corresponding points in the images.

$$T(\boldsymbol{O}'_c) = \boldsymbol{R}\boldsymbol{O}'_c + \boldsymbol{t} = \boldsymbol{O}''_c \tag{2.74}$$

The rigid transformation matrices from the world coordinate system to each camera coordinate system can be written as:

$$\boldsymbol{T}'_i = \begin{bmatrix} \boldsymbol{R}'_i & \boldsymbol{t}'_i \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \quad \boldsymbol{T}''_i = \begin{bmatrix} \boldsymbol{R}''_i & \boldsymbol{t}''_i \\ \boldsymbol{0}^T & 1 \end{bmatrix}. \tag{2.75}$$

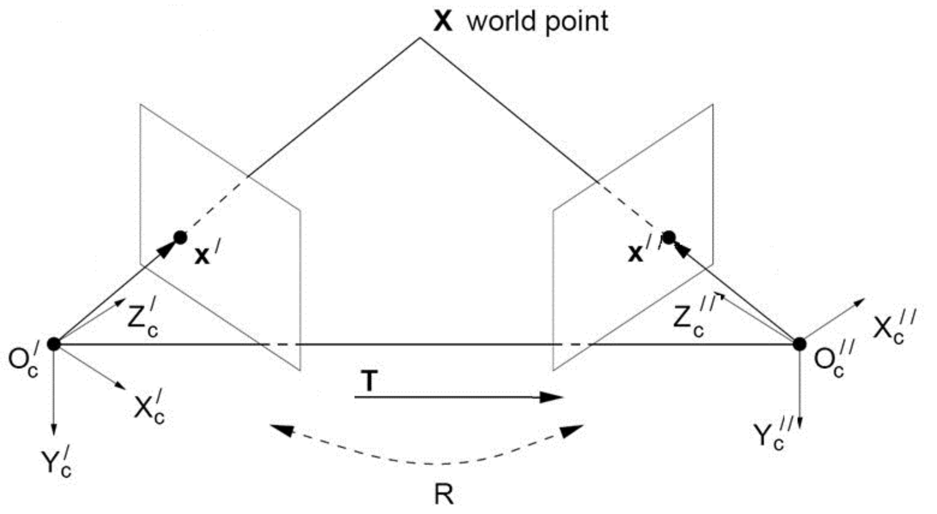


Figure 2.4: Stereo Camera Model [7].

The rigid transformation from left to right camera is then:

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \boldsymbol{T}_i'^{-1}\boldsymbol{T}_i'' \quad \forall i \in n. \tag{2.76}$$

Equation 2.76 holds for all images taken since the cameras are fixed in space relative to each other. The estimated rigid transformation between the cameras may differ slightly because of noise in the data, therefore we take the average over all camera positions to get a better estimate of the relative position of the cameras.

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{T}_i'^{-1}\boldsymbol{T}_i'' \tag{2.77}$$

If $\boldsymbol{R}$ doesn't have the properties of a rotation matrix we can approximate it as in section 2.2.5.

### 2.3.3 The Fundamental and Essential Matrix

From figure 2.4 we can see that the vectors $\overrightarrow{\boldsymbol{O}_c'\boldsymbol{X}}$, $\overrightarrow{\boldsymbol{O}_c''\boldsymbol{X}}$ and $\overrightarrow{\boldsymbol{O}_c'\boldsymbol{O}_c''}$ forms a plane. Where $\boldsymbol{O}_c'$ and $\boldsymbol{O}_c''$ is the position of the pinhole of the left and right camera respectively. The coplanarity constraint can be written as the triple product of the vectors.

$$\overrightarrow{\boldsymbol{O}'_c\boldsymbol{X}} \cdot (\overrightarrow{\boldsymbol{O}'_c\boldsymbol{O}''_c} \times \overrightarrow{\boldsymbol{O}''_c\boldsymbol{X}}) = 0 \tag{2.78}$$

By rewriting equation 2.73 for both cameras.

$$\boldsymbol{x}' = \boldsymbol{K}'\boldsymbol{R}' \begin{bmatrix} \boldsymbol{I}_3 & -\boldsymbol{O}'_c \end{bmatrix} \boldsymbol{X}, \quad \text{since} \quad -\boldsymbol{R}'\boldsymbol{O}'_c = \boldsymbol{t}' \tag{2.79}$$

This equation is true for all images taken, and the subscript denoting the image number is therefor excluded. We can now explicitly write the vectors that defines the plane

$$\begin{aligned}
\overrightarrow{\boldsymbol{O}'_c\boldsymbol{X}} &= \boldsymbol{R}'^T\boldsymbol{K}'^{-1}\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{I}_3 & -\boldsymbol{O}'_c \end{bmatrix} \boldsymbol{X} \\
\overrightarrow{\boldsymbol{O}''_c\boldsymbol{X}} &= \boldsymbol{R}''^T\boldsymbol{K}''^{-1}\boldsymbol{x}'' = \begin{bmatrix} \boldsymbol{I}_3 & -\boldsymbol{O}''_c \end{bmatrix} \boldsymbol{X} \\
\overrightarrow{\boldsymbol{O}'_c\boldsymbol{O}''_c} &= \boldsymbol{b} = \boldsymbol{O}''_c - \boldsymbol{O}'_c
\end{aligned}$$

By substituting the definitions of the vectors into 2.78, we get a fundamental result:

$$\boldsymbol{x}'^T\boldsymbol{K}'^{-T}\boldsymbol{R}'\boldsymbol{S_b}\boldsymbol{R}''^T\boldsymbol{K}''^{-1}\boldsymbol{x}'' = 0 \tag{2.80}$$

where the matrix

$$\boldsymbol{F} = \boldsymbol{K}'^{-T}\boldsymbol{R}'\boldsymbol{S_b}\boldsymbol{R}''^T\boldsymbol{K}''^{-1} \tag{2.81}$$

is called the fundamental matrix. The fundamental matrix contains information about the relative orientation of two images from two uncalibrated cameras and satisfies the equation $\boldsymbol{x}'^T\boldsymbol{F}\boldsymbol{x}'' = 0$ for corresponding points. For calibrated cameras we can obtain the directions of the vectors from the two camera centers to the world point as:

$$\boldsymbol{x}'_K = \boldsymbol{K}'^{-1}\boldsymbol{x}', \quad \boldsymbol{x}''_K = \boldsymbol{K}''^{-1}\boldsymbol{x}''. \tag{2.82}$$

By substituting equation 2.82 into 2.80 we get:

$$\boldsymbol{x}'^T_K\boldsymbol{R}'\boldsymbol{S_b}\boldsymbol{R}''^T\boldsymbol{x}''_K = 0 \tag{2.83}$$

where the essential matrix is defined as:

$$\boldsymbol{E} = \boldsymbol{R}'\boldsymbol{S_b}\boldsymbol{R}''^T \tag{2.84}$$

The essential matrix contains the same information as the fundamental matrix, but only for calibrated cameras. The coplanarity constraint can be simplified for calibrated cameras:

$$\boldsymbol{x}'^{T}_{K}\boldsymbol{E}\boldsymbol{x}''_{K} = 0 \tag{2.85}$$

The reader is referred to [5, 15] for a more detailed description of the derivation and properties of the essential and fundamental matrix. These equations forms the basis for the theory of epipolar geometry.

### 2.3.4 Epipolar Geometry

Epipolar geometry describes the relation between corresponding points in the images taken from two cameras in a stereo rig, and is therefore essential for solving the correspondence problem. It will reduce the search space form 2D to 1D. Lets go through some important elements of the epipolar geometry and how they are defined, stated in [5].

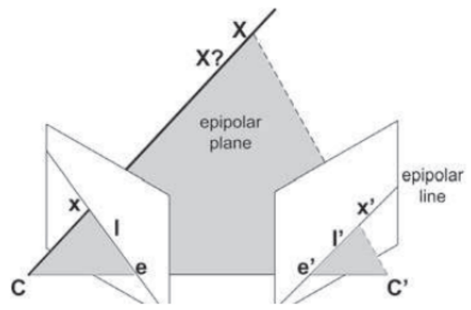


Figure 2.5: An illustration of the the epipolar plane, epipoles and the epipolar lines [15].

- The epipolar axis is the line connecting the two projection centers.

$$\boldsymbol{\beta} = (\boldsymbol{O}'_c\boldsymbol{O}''_c) \tag{2.86}$$

- The epipolar plane depends on the positions of the projection centers and the world point $\boldsymbol{X}$.

$$\varepsilon(\boldsymbol{X}) = (\boldsymbol{O}'_c\boldsymbol{O}''_c\boldsymbol{X}) \tag{2.87}$$

- The epipoles are the image points of the other projection centers. The epipoles in the left and right image plane are defined by:

$$\boldsymbol{e}' = (\boldsymbol{O}''_c)', \quad \boldsymbol{e}'' = (\boldsymbol{O}'_c)'' \tag{2.88}$$

where $(\cdot)'$ and $(\cdot)''$ is the projection of $(\cdot)$ in the left and right image plane respectively.

- The epipolar lines are the image of the line connecting the projection center and the world point $X$ in the other image.

$$\boldsymbol{l}'(\boldsymbol{X}) = (\boldsymbol{O}''_c\boldsymbol{X})', \quad \boldsymbol{l}''(\boldsymbol{X}) = (\boldsymbol{O}'_c\boldsymbol{X})'' \tag{2.89}$$

The projection centers, the epipolar lines, the world point $\boldsymbol{X}$ and the $(\boldsymbol{X})'$ and $(\boldsymbol{X})''$ projections lies on the same epipolar plane. Say that we have the projection $(\boldsymbol{X})'$ then we define the epipolar plane $\boldsymbol{\varepsilon}((\boldsymbol{X})') = (\boldsymbol{O}'_c\boldsymbol{O}''_c(\boldsymbol{X})')$. The intersection of the epipolar plane $\boldsymbol{\varepsilon}((\boldsymbol{X})')$ and the right image plane is the epipolar line $\boldsymbol{l}''((\boldsymbol{X})')$. This means that $(\boldsymbol{X})''$ must lie somewhere on the epipolar line $\boldsymbol{l}''((\boldsymbol{X})')$. Lets look into how we can obtain the epipolar lines. We have two corresponding points:

$$\boldsymbol{x}' = \boldsymbol{P}'\boldsymbol{X}, \quad \boldsymbol{x}'' = \boldsymbol{P}''\boldsymbol{X} \tag{2.90}$$

where $\boldsymbol{P}'$ and $\boldsymbol{P}''$ is the projection matrix for the left and right camera respectively. If $\boldsymbol{x}'$ and $\boldsymbol{x}''$ lies on the epipolar lines $\boldsymbol{l}'$ and $\boldsymbol{l}''$, respectively, the following must hold:

$$\boldsymbol{x}'^T\boldsymbol{l}' = 0, \quad \boldsymbol{x}''^T\boldsymbol{l}'' = 0 \tag{2.91}$$

The fundamental matrix $\boldsymbol{F}$ has the property for corresponding points such that:

$$\boldsymbol{x}'^T\boldsymbol{F}\boldsymbol{x}'' = 0, \quad \boldsymbol{x}''^T\boldsymbol{F}^T\boldsymbol{x}' = 0 \tag{2.92}$$

By looking at equations 2.91 and 2.92 we can easily see that the epipolar lines are given by:

$$\boldsymbol{l}' = \boldsymbol{F}\boldsymbol{x}'', \quad \boldsymbol{l}'' = \boldsymbol{F}^T\boldsymbol{x}' \tag{2.93}$$

### 2.3.5 Rectification

From last section we describe the epipolar lines. It would be useful if the epipolar lines are horizontal, e.g.there is no shift in the y-parallax between the image pair. This will simplify the correspondence problem even further, the search space for corresponding points will now be a horizontal 1D line in the image pair. If we have two corresponding points in pixel coordinates, $\boldsymbol{x}'$ and $\boldsymbol{x}''$, assuming that the images are rectified, the two corresponding pixel points have the following relation:

$$\boldsymbol{x}' = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \boldsymbol{x}'' = \begin{bmatrix} x+d \\ y \\ 1 \end{bmatrix} \tag{2.94}$$

where the scalar $d \in \mathbb{R}$ is the disparity. The disparity is the difference between the corresponding points along the $x$-axis in the normalized image pair. Lets look into how we can achieve this relation between image points for our stereo camera rig. We define the origin of the stereo camera coordinate system as the projection center of the left camera. Earlier we calculated the the rigid transformation, $\boldsymbol{R}$ and $\boldsymbol{t}$, of the right camera relative to the left. As a result of this we can define the extrinsics of the two cameras in the new coordinate system as:

$$\boldsymbol{R}' = \boldsymbol{I}_3, \quad \boldsymbol{t}' = \boldsymbol{0} \tag{2.95}$$

$$\boldsymbol{R}'' = \boldsymbol{R}, \quad \boldsymbol{t}'' = \boldsymbol{t} \tag{2.96}$$

$$\boldsymbol{x}' = \boldsymbol{K}' \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0} \end{bmatrix} \boldsymbol{X}, \quad \boldsymbol{x}'' = \boldsymbol{K}'' \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \boldsymbol{X} \tag{2.97}$$

where the left camera coordinate system are define as the world coordinate system. The new image points in the stereo normal case are given by:

$$\boldsymbol{x}'_m = \boldsymbol{K}_m \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{0} \end{bmatrix} \boldsymbol{X}, \quad \boldsymbol{x}''_m = \boldsymbol{K}_m \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{t} \end{bmatrix} \boldsymbol{X} \tag{2.98}$$

where $\boldsymbol{K}_m = \text{diag}(c, c, 1)$ is a common calibration matrix to ensure that the normalized image planes are at the same distance from their respective projection center. From figure 2.6 you can see the normalized image pair shaded in gray. The goal of this transformation is that corresponding image points have no $y$-parallax. We want to apply a homography to each image that maps them over to images that are normalized. In [5] the homography's are defined as:

$$\boldsymbol{x}'_m = \boldsymbol{H}'_m \boldsymbol{x}', \quad \boldsymbol{x}''_m = \boldsymbol{H}''_m \boldsymbol{x}'' \tag{2.99}$$

$$\boldsymbol{H}'_m = \boldsymbol{K}_m \boldsymbol{R}_m \boldsymbol{K}'^{-1}, \quad \boldsymbol{H}''_m = \boldsymbol{K}_m \boldsymbol{R}_m \boldsymbol{R}^T \boldsymbol{K}''^{-1}.$$

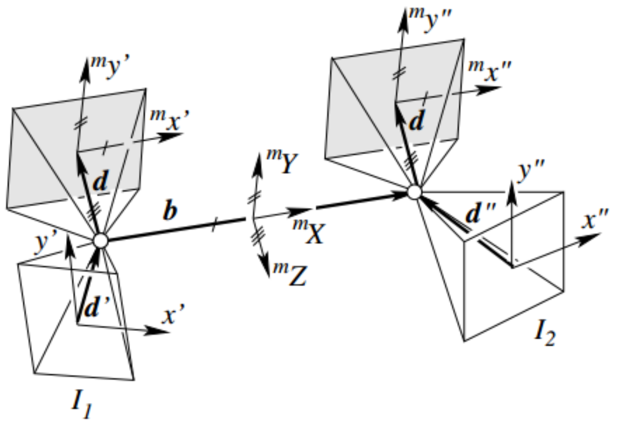


Figure 2.6: Normalized stereo pairs [5]

The rotation matrix $\boldsymbol{R}_m$ are defined as

$$\boldsymbol{R}_m = \begin{bmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{bmatrix} \tag{2.100}$$

and is chosen such that the following three conditions hold, stated in [5]:

- The direction of the $x'_m$ and $x''_m$ axes are parallel to the baseline $t$. This results in:

$$\boldsymbol{r}_1 = \frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} \tag{2.101}$$

- The new viewing directions are perpendicular to the baseline and are as close as possible to the original viewing direction. We assume that the viewing direction of each camera don't differ significantly, therefor we choose the new viewing direction with respect to the left camera only. The original viewing direction of the left camera is $\boldsymbol{d}' = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ direction.

$$\boldsymbol{r}_2 = \frac{\boldsymbol{S_t d'}}{\|\boldsymbol{S_t d'}\|} \tag{2.102}$$

- The direction of the $y'_m$ and $y''_m$ are orthonormal to plane generated by the new viewing direction and the $x'_m$ and $x''_m$ axes.

$$\boldsymbol{r}_3 = \frac{\boldsymbol{S}_{\boldsymbol{r}_1}\boldsymbol{r_2}}{\|\boldsymbol{S}_{\boldsymbol{r}_1}\boldsymbol{r_2}\|} \tag{2.103}$$

The new common viewing direction for the normalized cameras is given by:

$$\boldsymbol{d} = \boldsymbol{r}_3 \tag{2.104}$$

By arbitrarily choosing the world coordinate system to have the same orientation as the left normalized camera coordinate system, we can simplify the projection matrices for both the normalized cameras since they have the same orientation in space. Note that the origin of the world coordinate system doesn't change because the rectification only apply a rotation. Be rewriting equation 2.98 after the reorientation of the world coordinate system we get:

$$\boldsymbol{x}'_m = \boldsymbol{K}_m \begin{bmatrix}\boldsymbol{I}_3 & \boldsymbol{0}\end{bmatrix} \boldsymbol{X}, \quad \boldsymbol{x}''_m = \boldsymbol{K}_m \begin{bmatrix}\boldsymbol{I}_3 & \boldsymbol{b}\end{bmatrix} \boldsymbol{X} \tag{2.105}$$

where $\boldsymbol{b} = \begin{bmatrix}\|\boldsymbol{t}\| & 0 & 0\end{bmatrix}^T$. We can now prove that the epipolar lines are horizontal for normalized cameras using the coplanarity constraint. The essential matrix for the normalized stereo pair is given by:

$$\boldsymbol{E} = \boldsymbol{R}'\boldsymbol{S}_{\boldsymbol{b}}\boldsymbol{R}''^T = \boldsymbol{S}_{\boldsymbol{b}} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -\|\boldsymbol{t}\|\\ 0 & \|\boldsymbol{t}\| & 0\end{bmatrix} \tag{2.106}$$

where $\boldsymbol{R}' = \boldsymbol{R}'' = \boldsymbol{I}_3$. We obtain the following coplanarity constraint for corresponding points in the normalized image planes:

$$\boldsymbol{x}_c'^T \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -\|\boldsymbol{t}\|\\ 0 & \|\boldsymbol{t}\| & 0\end{bmatrix} \boldsymbol{x}''_c = c\|\boldsymbol{t}\|(y''_c - y'_c) = 0 \tag{2.107}$$

where

$$\boldsymbol{x}'_c = \begin{bmatrix}x'_c & y'_c & c\end{bmatrix}^T = c\boldsymbol{K}_m^{-1}\boldsymbol{x}'_m, \quad \boldsymbol{x}''_c = \begin{bmatrix}x''_c & y''_c & c\end{bmatrix}^T = c\boldsymbol{K}_m^{-1}\boldsymbol{x}''_m.$$

The vectors $\boldsymbol{x}'_c$ and $\boldsymbol{x}''_c$ are the mapping of image points $\boldsymbol{x}'_m$ and $\boldsymbol{x}''_m$ in their respective normalized image plane and also contain the direction from their respective projection center to the world point $\boldsymbol{X}$. The result of the coplanarity constraint in equation 2.107 prove that the normalized image pair don't have any shift in the $y_c$-parallax in the image planes for corresponding points.

$$p_{y_c} = y''_c - y'_c = 0 \tag{2.108}$$

We can also show that the $y_m$-parallax is zero for corresponding image points $\boldsymbol{x}'_m$ and $\boldsymbol{x}''_m$ by looking at the coplanarity constraint given by the fundamental matrix $\boldsymbol{F}$.

$$\boldsymbol{F} = \boldsymbol{K}'^{-T}\boldsymbol{R}'\boldsymbol{S_b}\boldsymbol{R}''^{T}\boldsymbol{K}''^{-1} = \boldsymbol{K}_m^{-T}\boldsymbol{S_b}\boldsymbol{K}_m^{-1} \tag{2.109}$$

$$\boldsymbol{F} = \begin{bmatrix} \frac{1}{c} & 0 & 0 \\ 0 & \frac{1}{c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\|\boldsymbol{t}\| \\ 0 & \|\boldsymbol{t}\| & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{c} & 0 & 0 \\ 0 & \frac{1}{c} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.110}$$

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{\|\boldsymbol{t}\|}{c} \\ 0 & \frac{\|\boldsymbol{t}\|}{c} & 0 \end{bmatrix} \tag{2.111}$$

We get the following coplanarity constraint for corresponding normalized image points:

$$\boldsymbol{x}'^{T}_m\boldsymbol{F}\boldsymbol{x}''_m = \frac{\|\boldsymbol{t}\|}{c}(y''_m - y'_m) = 0 \tag{2.112}$$

where $\boldsymbol{x}'_m = \begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix}^T$ and $\boldsymbol{x}''_m = \begin{bmatrix} x''_m & y''_m & 1 \end{bmatrix}^T$ are the corresponding normalized image points. The result of the constraint in equation 2.112 prove that the $y_m$-parallax must is zero for corresponding normalized image points.

$$p_{y_m} = y''_m - y'_m = 0 \tag{2.113}$$

### 2.3.6 Triangulation

To solve the reconstruction problem we use triangulation to compute the 3D coordinates of a world point based on the corresponding points. The system have been transformed into a normalized stereo pair, this simplifies the reconstruction problem. The detected corresponding image points of the unknown world point $\boldsymbol{X}$ are given by equation 2.105. Since we are working with normalized image points, the detected image points have the relation:

$$\boldsymbol{x}'_m = \begin{bmatrix} x'_m \\ y'_m \\ 1 \end{bmatrix}, \quad \boldsymbol{x}''_m = \begin{bmatrix} x'_m + d \\ y'_m \\ 1 \end{bmatrix} \tag{2.114}$$

where $d \in \mathbb{R}$ is the disparity. We will look into how we can estimate the disparity in the next chapter. For now we assume that $d$ is known. In figure 2.7 $B$ is

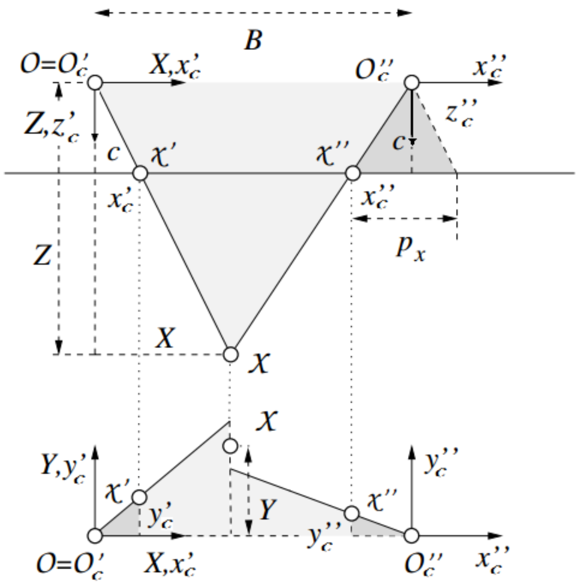


Figure 2.7: View of the stereo normal cameras in the $XZ$ and $XY$ plane in world coordinates. The world coordinate system with its origin $\boldsymbol{O}$ is identical to the left camera coordinate system [5].

the metric distance of the baseline and $p_x$ is the $x$-parallax in the image planes given by:

$$B = \|\boldsymbol{t}\|, \quad p_x = x''_c - x'_c \tag{2.115}$$

where

$$\boldsymbol{x}'_c = \begin{bmatrix} x'_c \\ y'_c \\ c \end{bmatrix} = c\boldsymbol{K}_m^{-1}\boldsymbol{x}'_m, \quad \boldsymbol{x}''_c = \begin{bmatrix} x''_c \\ y''_c \\ c \end{bmatrix} = c\boldsymbol{K}_m^{-1}\boldsymbol{x}''_m.$$

The vectors $\boldsymbol{x}'_c$ and $\boldsymbol{x}''_c$ are the mapping of image points $\boldsymbol{x}'_m$ and $\boldsymbol{x}''_m$ in the left and right image plane, respectively. The scalar $c$ is the common focal length for the stereo normal cameras, defined in the previous sub chapter. By examining figure 2.7, we can express the coordinates for world point $\boldsymbol{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ using the rule of similar triangles. The coordinates are given by:

$$X = x'_c \frac{B}{p_x}, \quad Y = \frac{y'_c + y''_c}{2} \frac{B}{p_x}, \quad Z = c\frac{B}{p_x}. \tag{2.116}$$

Since $y'_c$ should be equal to $y''_c$ in the stereo normal case, we can express the $Y$ coordinate as:

$$Y = y'_c \frac{B}{p_x} \tag{2.117}$$

We can write the equation for the inhomogeneous world coordinate $\boldsymbol{X}$ on linear form using equations 2.116 and 2.117.

$$\boldsymbol{X} = \begin{bmatrix} \frac{B}{p_x} & 0 & 0 \\ 0 & \frac{B}{p_x} & 0 \\ 0 & 0 & \frac{B}{p_x} \end{bmatrix} \boldsymbol{x}'_c \tag{2.118}$$

By converting to homogeneous coordinates we can use the $x$-parallax as input.

$$\begin{bmatrix} U \\ V \\ W \\ T \end{bmatrix} = \begin{bmatrix} B & 0 & 0 & 0 \\ 0 & B & 0 & 0 \\ 0 & 0 & B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'_c \\ y'_c \\ c \\ p_x \end{bmatrix} \tag{2.119}$$

The transformation between inhomogeneous and homogeneous coordinates is given by:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{T} \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \tag{2.120}$$

The error of depth, derived in [5], is given by:

$$\Delta Z = \frac{Z^2}{cB} \Delta p_x \tag{2.121}$$

where $\Delta p_x$ is the expected error of the $x$-parallax. We can see that the error of $Z$, $\Delta Z$, is increasing by the square of $Z$. The error of $Z$ is proportional to $p_x$, we can reduce the error of the $x$-parallax by estimating the disparity with sub-pixel accuracy. We will show how we can estimate the disparity with sub-pixel accuracy in the next chapter. The depth error can be reduced by increasing the focal length and baseline. This also have its drawbacks, because increasing the baseline will make the matching harder. Increasing the focal length will reduce the field of view, as a result of equation 2.122. The horizontal and vertical field of view is given by:

$$\theta_H = 2\arctan\left(\frac{h}{2f}\right), \quad \theta_V = 2\arctan\left(\frac{v}{2f}\right) \tag{2.122}$$

where $\theta_H$ and $\theta_V$ are the horizontal and vertical field of view in degrees, respectively. The width and height of the image sensor are given by $h$ and $v$, and $f$ is the focal length. From 2.122 we can also see that by increasing the size of the image sensor, we can increase the field of view. A consequence of having a larger image sensor is that it increases the number of pixels, higher resolution resulting in more processing time. It is important to have these relations in mind when designing a stereo camera system. The design procedure involves a number of performance trade-offs and are chosen according to the application requirements of the system. Stereo camera systems will typically work within a limited range, depending on the application, to have the best possible accuracy.

# 3 Approach

In this chapter we will present a solution for solving the correspondence problem. The main goal is to accurately find the corresponding point of the docking slot given by a tracker. When the corresponding point is found we can estimate the disparity used for calculate the three dimensional world coordinate of the docking slot. First we will have short description of the problem and the challenges that follows. The approach of finding the corresponding point is mostly based on the assumptions given in the introduction and of course theory presented in subsection 2.3. A review of the different computer vision algorithms used for finding corresponding points between the image pair are given in the end of this chapter.

## 3.1 Problem Description and Challenges

Since the platform is painted grey the surface is homogeneous and texture less. The fence poles all look the same and can give us ambiguous matches. There is no hallmark on the docking slot and is only painted grey as you can i see in 3.1.

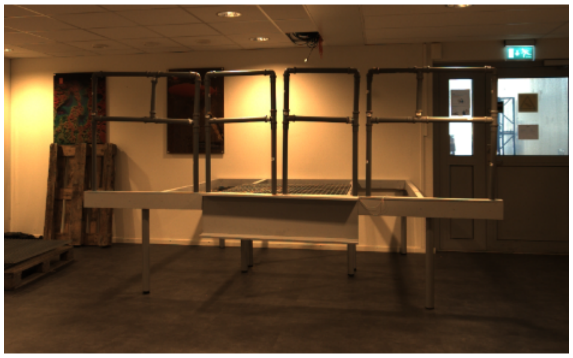


Figure 3.1: The platform built for experimental testing of the different algorithms and is a copy of an entrance of a windmill. The docking slot is positioned under the entrance in the center of the platform.

These two factors makes it challenging when searching for corresponding points in the image pair. therefore we will look into how we can narrow down the search region and how we can create a reasonable sized template to remove most of the scene. The template and search region should contain enough texture so we can get a accurate match between the image pair. When template and search region is created we will apply the different computer vision algorithms on them to find corresponding points and estimate the disparity value. The disparity value is key factor when calculating the three-dimensional coordinate of the docking slot relative to the camera.

## 3.2 Camera Setup

For this application Kongsberg designed their own stereo camera system. It consists of two IDS GigE uEye CP industrial cameras, [10], with a resolution of 4.19Mpix. The reason for choosing this camera is that it have a global shutter. A global shutter turns all the pixels on and off at the same time.

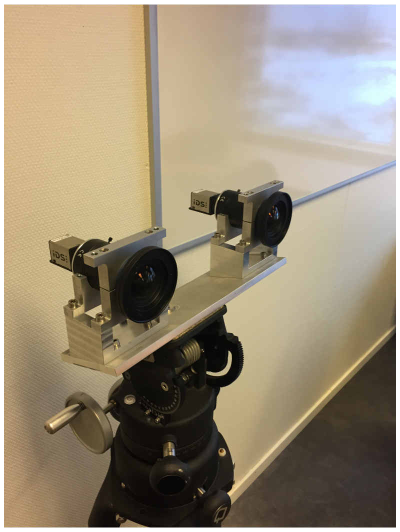
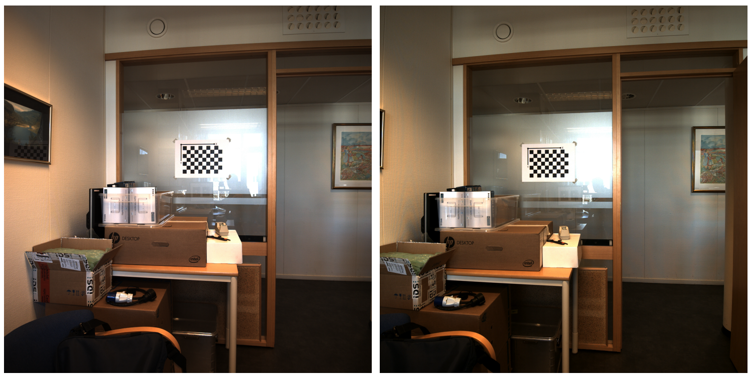


Figure 3.2: The stereo camera rig.

Since the camera will move when taking photos due to the sea sate and that the bridge will move when docking, a global shutter will not suffer from any skew in the image. But, you would get some motion blur depending on the exposure time. If the images suffered from skew it would be hard to match them and the pixel positions may not be at their true position resulting in a wrong depth measurement. From figure 3.2 you can see that the cameras are mounted in parallel to each other. Both cameras use the same lens with a focal length of $8[mm]$. The stereo rig was calibrated in Matlab using the stereo camera calibration app in the computer vision toolbox. The calibration app uses a planar based calibration approach that is reviewed in 2.2. To be able to perform the calibration, the app only need a set of image pairs containing a checkerboard, see figure 3.3, with known dimensions as input. Both the camera



(a) Left (b) Right

Figure 3.3: One of the image pairs containing a checker board taken by the stereo camera rig used for the calibration.

intrinsic, $\boldsymbol{K}$, and extrinsic matrix, $\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}$, are estimated for each camera. It also estimates the rigid transformation between the cameras. The stereo rig was only calibrated in a range of $2-3[m]$. The system is supposed to work in a range of $2-20[m]$, to improve accuracy of the rig it is advantageous to calibrate the rig in its operating range. The reprojection error, for each image used in the calibration, of the stereo rig is plotted in figure 3.4.

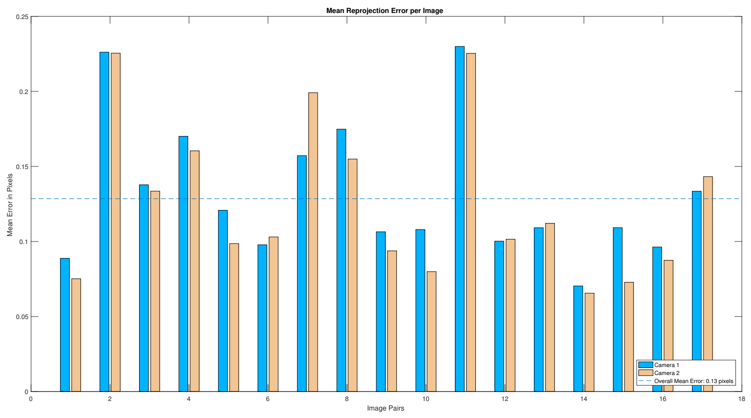


Figure 3.4: The reprojection error of the calibrated stereo camerea system. The blue and yellow bars represent the error for the left and right camera, respectively.

The mean reprojection error is estimated to be 0.13 pixels. By rewriting equation 2.121 with the focal length in $[pix]$, we can substitute the expected error in the $x$-parallax with the expected disparity error, $\Delta d$.

$$\Delta Z = \frac{Z^2}{fB}\Delta d \tag{3.1}$$

The estimated baseline of stereo rig is $B = 265.8[mm]$ and the average of the estimated focal lengths for each camera is $f = 7.93[mm]$. It is expected that the estimated focal length will deviate from the focal length given in the data sheet for the lens because of small manufacturing errors. All pixels for IDS camera is square and have a side length of $5.5[\frac{\mu m}{pix}]$ and the focal length in pixels is then $f = 1441.8[pix]$. If we have a perfect match between the image pair, i.e. the disparity value is correct, a reasonable expected error for the disparity can be set to two times the mean of the reprojection error, $\Delta d = 2 \times 0.13 = 0.26[pix]$. The expected depth error, assuming a perfect match, by using the estimated values of $f$, $B$ and $\Delta d$ in equation 3.1 for $Z \in [0, 20]$ is plotted in figure 3.5. If

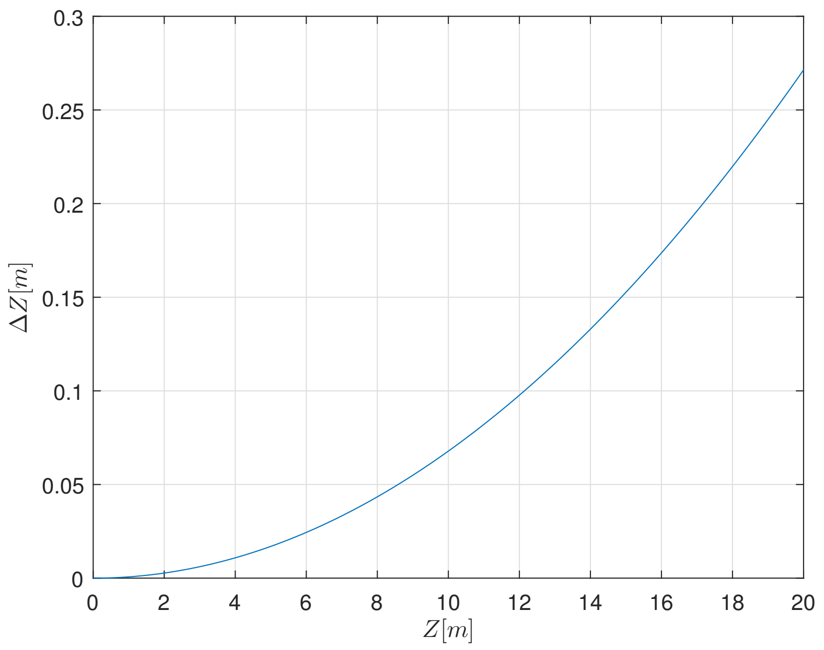


Figure 3.5: The expected depth error at different ranges. Parameters: $f = 1441.8[pix]$, $B = 256.8[mm]$ and $\Delta d = 0.26[pix]$

we always have a perfect match the stereo rig will have good accuracy as you can see from the plot, but we can expect that $\Delta d$ is much higher for the different algorithms we are gonna test later. To improve accuracy we can increase the baseline, but it is important that the docking slot is visible for both cameras in the working area of the bridge. As mentioned earlier this system should work at range of $2-20[m]$ and also within 10 degrees to the left and right of the center of the docking slot. If we position the center of the camera rig at 10 degrees to the left or right from center of the slot in the horizontal direction at $2[m]$ depth we can estimate the maximum baseline where the whole docking slot is visible for both cameras at these positions. When the docking slot is visible for both cameras at this range and angle, it also will be visible for both at longer ranges with same angle or smaller. The field of view of the cameras in the horizontal and vertical direction is equal because of a square image sensor and is:

$$\theta_h = \theta_v = 2\arctan\left(\frac{h}{2f}\right) = 2\arctan\left(\frac{11.264}{2\times 7.93}\right) = 70.76^\circ.$$

The width of the docking slot is $w_{slot} = 1.2[m]$. The maximum we can move the left or right camera from these two positions by adjusting the baseline where the docking slot is visible for both cameras can be described with the following equation.

$$2\tan\left(\frac{\theta_h}{2}\right) = 2\tan(10) + \frac{w_{slot}}{2} + \frac{B}{2} \tag{3.2}$$

The right side of equation represents the horizontal displacement of camera that is furthest away from the left or right edge of the docking slot and must be equal to the horizontal viewing distance of the camera at $2[m]$.Note that this equation only yields correct answer when the baseline of the rig is parallel to the face of the docking slot. Solving for the baseline we get $B = 0.9275[m]$. This is only a theoretical value and should be tested, but there shouldn't be a problem to increase the current baseline with twice its size and increase the accuracy of the stereo rig even further.

### 3.2.1 Template and Search Region

Before creating template and search region we undistort and rectify the image pair. This will simplify the problem when searching for corresponding points, now we know that corresponding points must lie on the same horizontal epipolar line as discussed in the previous chapter. It is assumed that we get the pixel position of the center of the docking slot from a tracker. In this thesis we define that we get this information in the left image, i.e. $\boldsymbol{x}'_m$ is known. If we have some information of how far we are from the docking slot we can estimate a disparity range where the corresponding point should lie by using the equations from 2.3.6. We can get this information from the previous measurement produced by this algorithm, GPS, IMU or the mathematical model of the bridge used by the control system and we can make prediction of what the disparity range should be. When estimating the disparity range, creating template and search region we are only interested in the $Z_c$-coordinate. Lets define the depth measurement we get from some source in the system as $Z_{est}$. Since the bridge most likely will move in some direction between each image pair taken we need to make a depth range prediction, $Z_c \in [Z_{min}, Z_{max}]$. We define the limits as:

$$Z_{min} = (1-\alpha)Z_{est}, \quad Z_{max} = (1+\alpha)Z_{est}, \quad \alpha \in [0,1]. \tag{3.3}$$

The depth is inverse proportional with the disparity value, thereforee $Z_{max}$ corresponds to $d_{min}$ and opposite. We can calculate these disparity values with use of equations 2.114, 2.115 and 2.116.

$$p_{xmin} = \frac{cB}{Z_{max}}, \quad p_{xmax} = \frac{cB}{Z_{min}} \tag{3.4}$$

And we can express the disparity values that corresponds to each distance in the image plane as:

$$\begin{bmatrix} d_{min} \\ 0 \\ 1 \end{bmatrix} = \frac{1}{c}\boldsymbol{K}_m \begin{bmatrix} p_{xmin} \\ 0 \\ c \end{bmatrix}, \quad \begin{bmatrix} d_{max} \\ 0 \\ 1 \end{bmatrix} = \frac{1}{c}\boldsymbol{K}_m \begin{bmatrix} p_{xmax} \\ 0 \\ c \end{bmatrix}. \tag{3.5}$$

Hopefully will the corresponding point in the right image, $\boldsymbol{x}''_m$, lie somewhere on the horizontal interval:

$$d_{min} + x'_m \leq x''_m \leq d_{max} + x'_m. \tag{3.6}$$

We have the option to tune $\alpha$ to create a wider or tighter disparity range. In this thesis this parameter will be set to a static value when running the algorithms on test photos, but it would be an idea to have $\alpha = f(Z_{est})$ because the uncertainty of the measurement depends on depth. Now that we have established a way to narrow down the search range we will have a look on how we can create a reasonable size for the template surrounding $\boldsymbol{x}'_m$. The docking slot will have the same dimension on every windmill or platform. The size of the object is known and we define the width and height as $X_{obj}$ and $Y_{obj}$ respectively. With this information we can calculated how many pixels we need to move in the horizontal and vertical direction in the image to create a template that have the same size as the object depending on $Z_{min}$ and $Z_{max}$. By use of equations 2.114, 2.115 and 2.116 again, we can calculate how much we can move in the horizontal and vertical direction in the image, to reach half of the size of the objects width and height in the real world.

$$\frac{1}{2}\begin{bmatrix} X_{obj} \\ Y_{obj} \\ 2Z_{max} \end{bmatrix} = \frac{B}{p_{xmin}} \begin{bmatrix} w'_c \\ h'_c \\ c \end{bmatrix} \tag{3.7}$$

where $w'_c = w'_c - x'_0$ and $h'_c = h'_c - y'_0$ are the distances between the principal point, $\boldsymbol{x}'_0$, and the two points in the image plane, $w'_c$ and $h'_c$, that corresponds to half the object size in the horizontal and vertical direction. Note that the principal point have the coordinates $\boldsymbol{x}'_0 = \begin{bmatrix} 0 & 0 & c \end{bmatrix}^T$ in the image plane. To find the number of pixels we need to move in both direction, first we need to find the pixel positions of $w'_c$ and $h'_c$ . By multiplying both sides of equation 3.7 with $\frac{\boldsymbol{K}_m}{c}$ we get the pixel positions $w'_m$ and $h'_m$.

$$\frac{p_{xmin}}{2cB}\boldsymbol{K}_m \begin{bmatrix} X_{obj} \\ Y_{obj} \\ 2Z_{max} \end{bmatrix} = \begin{bmatrix} w'_m \\ h'_m \\ 1 \end{bmatrix} \tag{3.8}$$

We need to find the principal point in pixel position and subtract it from $\begin{bmatrix} h'_m & w'_m & 1 \end{bmatrix}^T$ to get the distance in pixels. The principal point in pixels is given by:

$$\begin{bmatrix} x'_{m_0} \\ y'_{m_0} \\ 1 \end{bmatrix} = \frac{1}{c}\boldsymbol{K}_m \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix}. \tag{3.9}$$

The distance of half the size of the objects width and height in pixels is then:

$$\begin{bmatrix} w'_m \\ h'_m \end{bmatrix} - \begin{bmatrix} x'_{m_0} \\ y'_{m_0} \end{bmatrix} = \begin{bmatrix} w'_{min} \\ h'_{min} \end{bmatrix} \tag{3.10}$$

where $w'_{min}$ and $h'_{min}$ is the half size of the object in pixels given that we are at $Z_{max}$ away from the target. This can also be done for $Z_{min}$ and we can calculate $w'_{max}$ and $h'_{max}$ the same way. The template size is chosen as the average of the maximum and minimum size.

$$\frac{1}{2}\begin{bmatrix} w'_{min} \\ h'_{min} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} w'_{max} \\ h'_{max} \end{bmatrix} = \begin{bmatrix} w'_{avg} \\ h'_{avg} \end{bmatrix} \tag{3.11}$$

The pixels in the left image contained in the template is given by:

$$\boldsymbol{T} = \begin{bmatrix} (x'_m - w'_{avg}, y'_m - h'_{avg}) & \cdots & (x'_m + w'_{avg}, y'_m - h'_{avg}) \\ \vdots & (x'_m, y'_m) & \vdots \\ (x'_m - w'_{avg}, y'_m + h'_{avg}) & \cdots & (x'_m + w'_{avg}, y'_m + h'_{avg}) \end{bmatrix} \tag{3.12}$$

where

$$\boldsymbol{T} \in \mathbb{R}^{p \times q}, \quad p = 2h'_{avg} + 1, \quad q = 2w'_{avg} + 1.$$

It is important to have a template off odd size to ensure that the point of interest is in the center of the template. For the search region we want to have the same height as the template. We need to extend the width by the disparity range and center the interval from equation 3.6 in middle of the search region. The pixels in the right image contained in the search region is defined as:

$$\boldsymbol{S} = \begin{bmatrix} (d_{min} + x'_m - w'_{avg}, y'_m - h'_{avg}) & \cdots & (d_{max} + x'_m + w'_{avg}, y'_m - h'_{avg}) \\ \vdots & & \vdots \\ (d_{min} + x'_m - w'_{avg}, y'_m + h'_{avg}) & \cdots & (d_{max} + x'_m + w'_{avg}, y'_m + h'_{avg}) \end{bmatrix} \tag{3.13}$$

where

$$\boldsymbol{S} \in \mathbb{R}^{p \times r}, \quad r = 2w'_{avg} + d_{max} - d_{min}.$$

In figure 3.6 both the template and search region are drawn in the left and right image respectively. The true depth value was used, $Z_{est} = 2.26[m]$, and $\alpha = 0.25$. Further the object size was set to $X_{obj} = 1.5[m]$ and $Y_{obj} = 0.5[m]$ which is a bit larger than the real size of the docking slot. This is done with the purpose to get some more texture in the template. The cross in the template is the point given by the tracker and the line drawn within the search region is the disparity range given by equation 3.6. As mentioned earlier it is important to reduce the template and search region to remove potential ambiguous matches, but we will also lower the computation time for the different algorithms we are going to

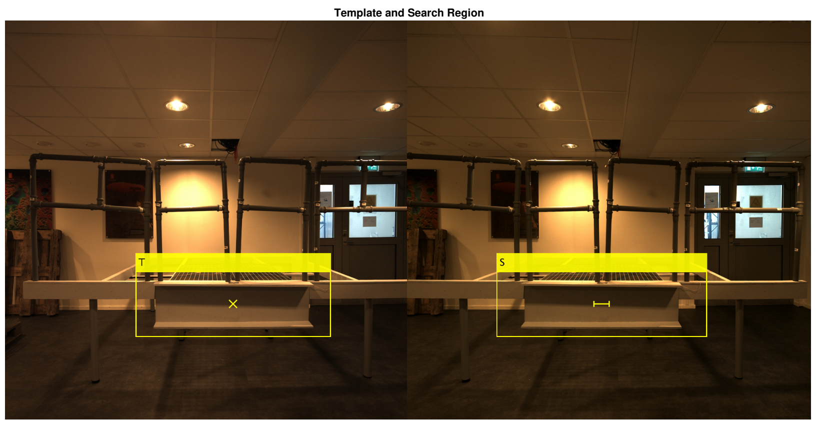


Figure 3.6: Template and search region generated by the equations in section 3.2.1. Parameters used: $Z_{est} = 2.26[m]$, $\alpha = 0.25$, $X_{obj} = 1.5[m]$ and $Y_{obj} = 0.5[m]$. The cross in the template is the point given by a tracker and the line drawn in the search region is the disparity range.

use when finding the corresponding point. Also note that the template size and disparity range depends on the depth, the template size and disparity range are inversely proportional with depth. This will result in increasing computation time as we approach the platform. Since we use the true depth value of the point given by the tracker, the corresponding point will always lie somewhere on the interval given by equation 3.6. We cannot be sure if this is the case when we use the posterior measurement. It can be solved analyzing the depth accuracy of the different algorithms and we can set a value for $\alpha$ based on the results.

## 3.3 Algorithms and Used Methods

In this section a theoretical review of the different algorithms used for solving the correspondence problem. The theory presented for all the correlation based methods is based on [15]. For the feature based methods, Harris corner detector, scale invariant feature transform and speeded up robust features, the reader is referred to [8], [11] and [3], respectively.

### 3.3.1 Preliminaries

Before we begin the review of the different algorithms, we will cover some basic image processing theory. A short introduction to image filtering and image pyramids is given to make the review of the algorithms, especially the feature-based methods, more compact and easier to read. The theory presented in this subsection is taken from [17]. Also a method for achieving sub pixel accuracy for the correlation based methods is presented.

#### 3.3.1.1 Image Filtering

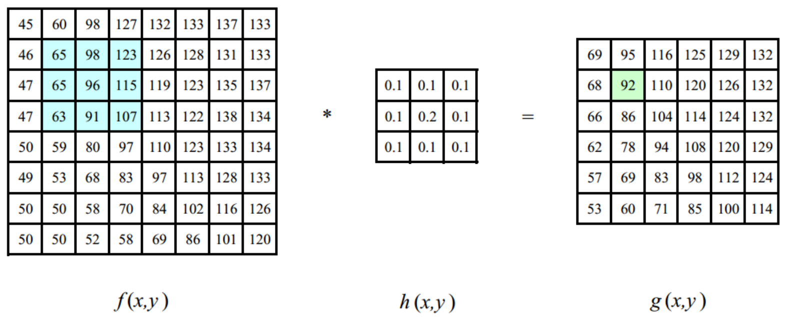
Image filters perform a wide range of image transformations such as bluring, sharpening or edge enhancements are just a few examples. An image filter is a neighborhood operator where the output pixel value is a weighted sum of the input pixels in a neighborhood around the pixel of interest. With different weights we can apply different image transformations. Let $g(i,j)$ be the output value of pixel $(i,j)$ and $f(i,j)$ is the input pixel value, the weighted sum over an area $(k,l)$ is given by:

$$g(i,j) = \sum_k \sum_l f(i-k, j-l)h(k,l). \tag{3.14}$$

The weight of each pixel is given by $h(k,l)$ and is a matrix often called kernel or convolution matrix. The entries of the kernel depends on the filter type and are called filter coefficients. All kernels are weighted by the absolute sum of the filter coefficients to ensure that the intensity values, after applying a kernel, is mapped to the same range. Equation 3.14 is the convolution between the kernel and the image and can by written as:

$$g = f * h \tag{3.15}$$

where $*$ is the convolution operator. From figure 3.7 you can see that the output



|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 45 | 60 | 98 | 127 | 132 | 133 | 137 | 133 |
| 46 | 65 | 98 | 123 | 126 | 128 | 131 | 133 |
| 47 | 65 | 96 | 115 | 119 | 123 | 135 | 137 |
| 47 | 63 | 91 | 107 | 113 | 122 | 138 | 134 |
| 50 | 59 | 80 | 97 | 110 | 123 | 133 | 134 |
| 49 | 53 | 68 | 83 | 97 | 113 | 128 | 133 |
| 50 | 50 | 58 | 70 | 84 | 102 | 116 | 126 |
| 50 | 50 | 52 | 58 | 69 | 86 | 101 | 120 |

|  |  |  |
|---|---|---|
| 0.1 | 0.1 | 0.1 |
| 0.1 | 0.2 | 0.1 |
| 0.1 | 0.1 | 0.1 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 69 | 95 | 116 | 125 | 129 | 132 |
| 68 | 92 | 110 | 120 | 126 | 132 |
| 66 | 86 | 104 | 114 | 124 | 132 |
| 62 | 78 | 94 | 108 | 120 | 129 |
| 57 | 69 | 83 | 98 | 112 | 124 |
| 53 | 60 | 71 | 85 | 100 | 114 |

Figure 3.7: An illustration of the convolution of image $f(x, y)$ with kernel $h(x, y)$. The area of the image in blue is convolved with the kernel and results in the output pixel value in green [17].

images shrinks depending on the size of the kernel. This can be compensated by padding the input image, for example with zeros. There exist many padding alternatives, but we will not go into detail on this theme. More interesting are the different types of kernels that are used in the algorithms we are going to review later. The Sobel operator is a discrete differentiation kernel and is used to compute a approximation of the pixel intensity derivatives in the horizontal and vertical direction. The Sobel operator for the horizontal and vertical direction is defined as:

$$\boldsymbol{D}_x = \frac{1}{8}\begin{bmatrix}1\\2\\1\end{bmatrix}\begin{bmatrix}-1 & 0 & 1\end{bmatrix} = \frac{1}{8}\begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix}, \quad \boldsymbol{D}_y = \boldsymbol{D}_x^T \in \mathbb{R}^{3\times 3}. \tag{3.16}$$

The vector $\frac{1}{2}\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$ is a 1D kernel that represents the central difference formula and $\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix}^T$ is a approximation of a Gaussian. The Gaussian is used to smooth out the pixel intensities in the vertical direction when computing the horizontal direction and vice versa. It is important to smooth out the image since the computation of the derivative is really sensitive to noise. The Gaussian kernel is used in computer vision algorithms for blurring(smoothing) or weighting an area in an image. The 2D Gaussian is given by:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2}e^{\frac{-x^2+y^2}{2\sigma^2}} \tag{3.17}$$

and the $\mathbb{R}^{5\times5}$ discrete approximation of the 2D Gaussian function with $\sigma = 1$ and zero mean results in the Gaussian kernel:

$$\boldsymbol{G} = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}.$$

The degree of blurring depends on the standard deviation of the Gaussian, larger standard deviation increases the amount of blur in the output image. The dimension of the kernel depends also on the standard deviation of the Gaussian because larger $\sigma$'s requires a higher dimension of the kernel in order to approximate the Gaussian accurately.

#### 3.3.1.2 Image Pyramids

An image pyramid is a multi-scale representation of an image and is introduced because it is a good approach to look for features of various scales. The process of creating an image pyramid is by repeatedly smooth and sub-sample the image, usually by a factor of two, at each level. The smoothing and sub-sampling can

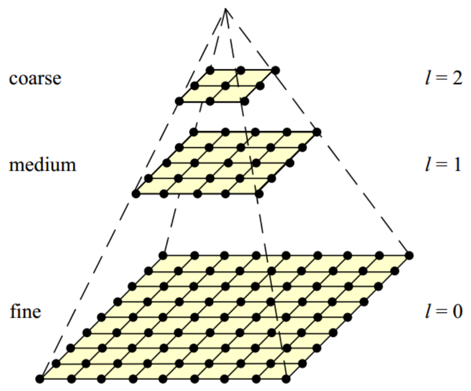


Figure 3.8: Illustration of a image pyramid with three levels [17].

be done in one operation with a decimation kernel. One of the most common decimation kernel, is the binomial filter.

$$\boldsymbol{b}^T = \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}, \quad \boldsymbol{B} = \boldsymbol{b}\boldsymbol{b}^T \tag{3.18}$$

Since the output image is smaller by a factor of two, $r = 2$, we only evaluate the convolution between the image and kernel at every second sample.

$$g(i,j) = \sum_k \sum_l f(ri - k, rj - l)h(k,l). \tag{3.19}$$

A illustration of equation 3.19 with the 1D binomial kernel is given in figure 3.9.

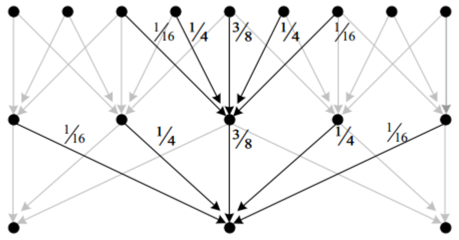


Figure 3.9: Smoothing and sub-sampling with the binomial kernel at each level in the pyramid. [17].

### 3.3.2 Sub-pixel Estimation

A problem when estimating the disparity is that we only get disparity values in natural numbers, $d \in \mathbb{N}$, because of the affine image sensor. A way to smooth out these edges is to use sub-pixel estimation to get $d \in \mathbb{R}$. The idea is to fit a parabola on the best match position and its nearest neighbors to get a better estimation of the disparity $d$, this is one of many methods presented in [2]. In

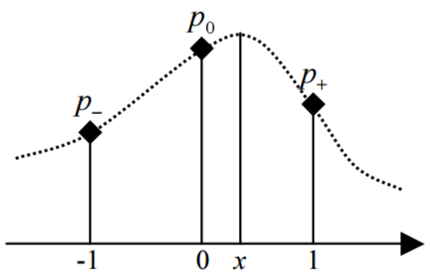


Figure 3.10: Pixel definitions [2]

figure 3.10 $p_0$ is the best match score from a correlation method with disparity $d$. Further $p_-$ and $p_+$ corresponds to the correlation scores at the neighbouring pixels with disparity $d-1$ and $d+1$ respectively. The parabola is defined as:

$$y = ax^2 + bx + c. \tag{3.20}$$

By differentiating and setting the derivative to zero we find the local maximum $x$.

$$\frac{dy}{dx} = 2ax + b = 0 \implies x = -\frac{b}{2a} \tag{3.21}$$

The values for the three known pixels can be written as:

$$\begin{aligned} y(-1) &= p_- = a - b + c \\ y(0) &= p_0 = c \\ y(1) &= p_+ = a + b + c. \end{aligned}$$

By substitution we can rewrite 3.21 and solve for location of the local maximum $x$.

$$x = \frac{p_- - p_+}{2(p_- + p_+) - 4p_0} \tag{3.22}$$

When $x$ is solved for we can improve the estimate of the disparity.

$$d_{est} = d + x \tag{3.23}$$

Sub-pixel estimation is implemented and used for all algorithms to improve the estimate of the location of corresponding points. When the corresponding points locations is estimated with sub pixel accuracy the disparity value will be calculated more precisely.

### 3.3.3 Correlation Based Methods

We are going to test three different correlation based methods, common for the three methods are that they use block-matching when comparing the images. For a given pixel in the left image we generate a window around it as done in the previous subsection. For further calculations we define the window for pixel

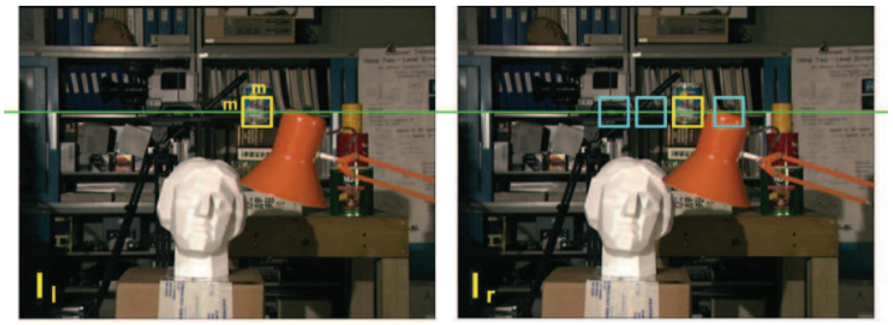


Figure 3.11: Block-matching illustration. The template is defined in the left image and moving along the the epipolar line in the right images to find the best match [15].

position $(x'_m, y'_m)$ as a set of pixels in following way:

$$W(x'_m, y'_m) = \left\{ (u,v) \middle| x'_m - \frac{(q-1)}{2} \le u \le x + \frac{(q-1)}{2}, \right. \\ \left. y'_m - \frac{(p-1)}{2} \le v \le y'_m + \frac{(p-1)}{2} \right\} \tag{3.24}$$

where $p$ and $q$ are the dimension of the template. By comparing the template with a window, of same size, in the search region we can calculate the correlation between them for each disparity value, $d \in [d_{min}, d_{max}]$, and conclude on the disparity value that corresponds to the highest correlation. Block-matching is a costly operation since we calculate the correlation between all pixels in the template with a an area of same size in the search region for each disparity value. therefore it is important, as mentioned earlier, that we try to narrow down search to reduce the number of calculations. The correlation based methods use the intensity of the pixels when calculating the correlation between them. In a colored image, each pixel contains three components of intensity such as red, blue and green. Instead of comparing three intensity's for each pixel we take a average or a weighted average over the intensity components. By doing this we convert a colored image into a greyscale image, containing only one intensity component for each pixel. In this thesis we have calculated the greyscale intensity as:

$$I = 0.2989R + 0.5870G + 0.1140B, \quad I, R, G, B \in [0, 255] \tag{3.25}$$

where $R$, $G$ and $B$ are the red, green and blue components of a pixel.

#### 3.3.3.1 Sum of Squared Differences (SSD)

The dissimilarities between the left and right block can be measured by the sum of squared differences. The cost function for SSD is given by [15]:

$$\min_{d \in [d_{min}, d_{max}]} \quad SSD(x'_m, y'_m, d) = \sum_{(u,v) \in W(x'_m, y'_m)} \big(I_l(u,v) - I_r(u+d,v)\big)^2 \quad (3.26)$$

where $I_l$ and $I_r$ are the intensities in the left and right image. Here we want to find the disparity $d$ that minimizes the cost function for a given pixel position $(x, y)$. If two image blocks correspond to the same world object, the intensities of the pixels should be similar, resulting in a small SSD value.

#### 3.3.3.2 Sum of Absolute Differences (SAD)

The sum of absolute differences is a slight variation of SSD. The cost function for SAD is given by [15]:

$$\min_{d \in [d_{min}, d_{max}]} \quad SAD(x'_m, y'_m, d) = \sum_{(u,v) \in W(x'_m, y'_m)} |I_l(u,v) - I_r(u+d,v)| \quad (3.27)$$

The SAD cost is less computational expensive than SSD because of the squaring operation, but the SSD cost function penalizes the difference of the intensities more. Due to illumination and non-Lambertian reflections, both the SSD and SAD may not give a low value even for correct matches. These effects can occur even if the images are taken simultaneously and will cause variations in the intensities. Illumination have to do with where the light source is and the shading of the scene in the image pair can be different, depending on the position of the light source. A shiny surface, a non-Lambertian surface, in the scene will reflect light creating specular highlights in the images. The angle of the reflected light depends on the direction of incoming light and the direction of the viewer. These specular highlights cause a change in the intensities in the image pair. In [15] they recommend to normalize the pixel intensities in the window by the purpose of filtering out the difference in intensities caused by these effects. This have been implemented for SSD and SAD. The normalized pixel intensities are given by:

$$I_n = \frac{I - \overline{I}}{\sigma_I} \quad (3.28)$$

where $\overline{I}$ and $\sigma_I$ are the mean and standard deviation of the intensities within the window. The mean and standard deviation of the intensity's in the window are given by:

$$\overline{I} = \frac{1}{pq} \sum_{(u,v)\in W(x'_m,y'_m)} I(u,v), \quad \sigma_I = \sqrt{\frac{1}{pq} \sum_{(u,v)\in W(x'_m,y'_m)} (I(u,v) - \overline{I})^2}.$$

When normalizing the pixel intensities we ensure that the they have zero-mean and unit variance.

#### 3.3.3.3 Normalized Cross Correlation (NCC)

The normalized cross correlation measures the similarities between the blocks- the disparity value that generate the highest score indicates the best match.The NCC measure is calculated as:

$$\max_{d\in[d_{min},d_{max}]} \quad NCC(x'_m, y'_m, d) = \frac{1}{pq} \sum_{(u,v)\in W(x'_m,y'_m)} \frac{(I_l(u,v) - \overline{I}_l)(I_r(u+d,v) - \overline{I}_r)}{\sigma_{I_l}\sigma_{I_r}}. \tag{3.29}$$

As you can see from equation 3.29 NCC uses normalized pixels and thereforee compensates for illumination and non-lambertian effects.

### 3.3.4 Feature Based Methods

Instead of using block-matching we want to apply some operations to the template and search region to extract distinct features in the two images and search for correspondences. For each of the feature-based algorithms we can divide this process into three main steps. First we find and select interest points at distinctive locations in the images. The most important property of a interest point detector is its repeatability, it should reliably find the same interest points under different viewing conditions. The second is to create a descriptor vector, also called feature vector, for each interest point. The descriptor vector must be distinctive and should be robust to noise, detection errors and photometric deformations. The third and final step is to match the descriptor vectors between the template and search region. When matching we look at the Euclidean distance between the descriptor vectors and also use the epipolar criteria on the matched interest points location in the image.

#### 3.3.4.1 Harris Corner Detector(HCD)

The Harris corner detector is an interest point detector that responds well to image corners. The idea of the corner detector is to consider a local window in the image and determining the average change of intensity that result from shifting the window by a small amount in various directions, [8]. If the window contains a corner or an isolated point, then all shifts will result in a large change. HCD uses a weighted sum of squared differences to measure the change of the intensities when applying a shift. By creating a window over the area $(u, v)$ of odd size centered at pixel $(i, j)$ in the image and shifting it by $(x, y)$, the change of the weighted SSD caused by a shift is written as:

$$E(x, y; i, j) = \sum_u \sum_v w(u, v)(I(u + x, v + y) - I(u, v))^2 \tag{3.30}$$

where $E(x, y; i, j)$ is the change of intensities caused by a displacement $(x, y)$ of the window at pixel position $(i, j)$ and $w(u, v)$ is the weight of the pixel $(u, v)$ in the window. We cover all possible shifts by performing a Taylor expansion of $I(u + x, v + y)$. The first order Taylor expansion is given by:

$$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y \tag{3.31}$$

where $I_x$ and $I_y$ are the partial derivatives of $I$. The partial derivatives for each pixel can by approximated by filtering an image patch surrounding the pixel with the Sobel operator. The approximation of the partial derivatives in both directions are calculated as the convolution of the two operators with an image patch of same size.

$$I_x(u, v) = \boldsymbol{D}_x * \boldsymbol{I}(u, v), \quad I_y(u, v) = \boldsymbol{D}_y * \boldsymbol{I}(u, v), \quad I_x, I_y \in \mathbb{R} \tag{3.32}$$

By combining equation 3.30 and 3.31 we get an approximation for the weighted SSD.

$$E(x, y; i, j) \approx \sum_u \sum_v w(u, v)(I_x(u, v)x + I_y(u, v)y)^2 \tag{3.33}$$

We can write equation 3.33 in linear form:

$$E(\boldsymbol{x}; i, j) \approx \boldsymbol{x}^T \boldsymbol{M}_{ij} \boldsymbol{x} \tag{3.34}$$

where $\boldsymbol{M}_{ij}$ is the structure tensor for pixel $(i, j)$ and $\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}^T$. The structure tensor is given by:

$$\boldsymbol{M}_{ij} = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2(u, v) & I_{xy}(u, v) \\ I_{xy}(u, v) & I_y^2(u, v) \end{bmatrix}.$$

where $I_{xy} = I_x I_y$. To be able to calculate the structure tensor, we must define the weight function. HCD uses a two dimensional Gaussian function with zero mean to weight the derivatives, typically with $\sigma = 1$. We can then write the equation for the structure tensor in following way:

$$\boldsymbol{M}_{ij} = \begin{bmatrix} \boldsymbol{G} * \boldsymbol{I}_x^2(u, v) & \boldsymbol{G} * \boldsymbol{I}_{xy}(u, v) \\ \boldsymbol{G} * \boldsymbol{I}_{xy}(u, v) & \boldsymbol{G} * \boldsymbol{I}_y^2(u, v) \end{bmatrix} \in \mathbb{R}^{2\times 2} \tag{3.35}$$

where $\boldsymbol{G}$ is the Gaussian kernel. $\boldsymbol{I}_x^2(u, v)$, $\boldsymbol{I}_y^2(u, v)$ and $\boldsymbol{I}_{xy}(u, v)$ contains all values of $I_x^2(u, v)$, $I_y^2(u, v)$ and $I_{xy}(u, v)$, respectively, in the window. When the structure tensor is calculated for all pixels in the image we need to decide which of them we should extract as possible feature points. Lets denote the elements of structure tensor as:

$$\boldsymbol{M}_{ij} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \quad E(\boldsymbol{x}; i, j) = ax^2 + 2bxy + cy^2 \tag{3.36}$$

If $a$ and $c$ are large and $b = 0$, we have a good minimum in $E$. Any shift in the $x$ and $y$ direction will produce a large change in $E$ which implies that we are at a corner or isolated point. The only problem is if $b \neq 0$, then we can have a flat surface along the diagonal in $E$. This can be resolved by using a rotationally invariant description of $\boldsymbol{M}_{ij}$,[8].

$$\begin{aligned} \boldsymbol{M}_{ij} \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} &= \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} \\ \boldsymbol{M}_{ij}\boldsymbol{R} &= \boldsymbol{\Lambda R} \\ \boldsymbol{R}^T \boldsymbol{M}_{ij} \boldsymbol{R} &= \boldsymbol{\Lambda} \end{aligned} \tag{3.37}$$

where $\lambda_1$ and $\lambda_2$ are the eigenvalues of $\boldsymbol{M}_{ij}$ and $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ are the corresponding normalized eigenvectors. By rotating the coordinate system to:

$$\boldsymbol{x} = \boldsymbol{R}\boldsymbol{z} \tag{3.38}$$

where $\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$. By substituting equations 3.38 and 3.37 into 3.33, $E$ takes the form:

$$E(\boldsymbol{z}; i, j) = \boldsymbol{z}^T \boldsymbol{R}^T \boldsymbol{M}_{ij} \boldsymbol{R} \boldsymbol{z} = \boldsymbol{z}^T \boldsymbol{\Lambda} \boldsymbol{z} \tag{3.39}$$

So if both eigenvalues are large we get a good minimum in $E$, a corner point. If one eigenvalue is low the surface is flat along that direction, an edge. HCD proposes a corner/edge response function to find possible feature points, also called an interest measure.

$$f(\boldsymbol{\Lambda}) = \mathrm{Det}(\boldsymbol{\Lambda}) - 0.06\, \mathrm{Tr}(\boldsymbol{\Lambda})^2 \tag{3.40}$$

The exact computation of the eigenvalues is expensive and in [8] they found that it is sufficient to evaluate the determinant and trace of $\boldsymbol{M}_{ij}$ to find interest points and the response function 3.40 becomes:

$$f(\boldsymbol{M}_{ij}) = \mathrm{Det}(\boldsymbol{M}_{ij}) - 0.06\, \mathrm{Tr}(\boldsymbol{M}_{ij})^2. \tag{3.41}$$

The interest measure is a useful indicator of which windows that can be reliably matched. If $f(\boldsymbol{M}_{ij}) > 0$ we have a corner or isolated point. The corner response function downweights edge-like features where $\lambda_2 \gg \lambda_1$. When the interest measure is calculated for all points you find the local maxima above a certain threshold and choose them as your feature points. HCD have not developed their own descriptor for their corner features, therefore you are free to choose any suitable descriptor. Often the keypoints are matched between the images by creating an image patch around the keypoint and match them with other patches by using normalized cross correlation.

#### 3.3.4.2 Scale Invariant Feature Transform(SIFT)

When looking for interest points you usually dont know what scales good features have. Using a fine scale may not be appropriate when an image contains of different homogeneous regions. SIFT solves this problem by extracting features at different scales and is achieved by searching for features at different levels in an image pyramid and matching them at the same level. This is a suitable algorithm for stereo camera systems since the images being matched won't suffer from large scale differences. Instead of looking for corner, the SIFT looks for blob like features. The keypoints are found by looking for scale-space extrema in the difference-of-Gaussians. The scale-space of an image is defined as the function, given in [11], as:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.42}$$

where $L(x, y\sigma)$ is the scale-space of the input image $I(x, y)$ at scale $\sigma$. By convolving the input image multiple times with Gaussians with incremental scale, the difference-of-Gaussians $D(x, y, \sigma)$ is calculated by taking the difference of two nearby scales separated by a constant factor $k$.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{3.43}$$

This procedure is done at each octave in an image pyramid with a down sample rate of two. When one octave is complete the scale is set to twice the initial value of $\sigma$ for the next octave and so forth. An illustration of this process is shown in figure 3.12. The reason for choosing the difference-of-Gaussians

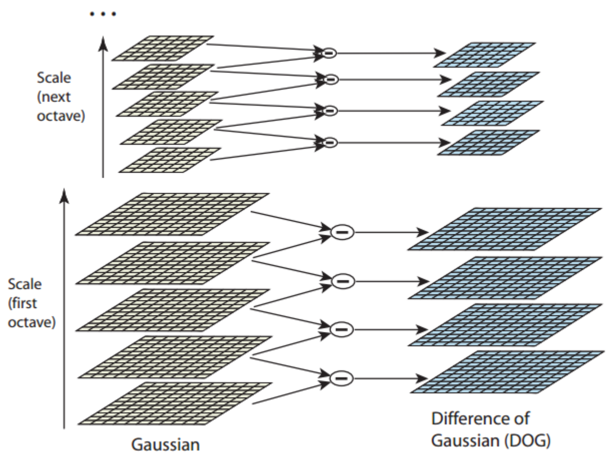


Figure 3.12: On the left we have the scale space of the input image and on the right the difference-of-Gaussians, [11].

when looking for possible feature points is because it is a close approximation of the scale-normalized Laplacian-of-Gaussians, $\sigma^2\nabla^2 G$, where $\nabla^2$ is the Laplacian operator. The Laplacian-of-Gaussians is the first and most commonly used blob detector. It have been shown that the normalization of the Laplacian with a factor of $\sigma^2$ is required for true scale invariance [11], this allows for detection of interest points with their own characteristic scale. Also the maxima and minima of $\sigma^2\nabla^2 G$ produce the the most stable image features compared to other interest functions. The relation between $D(x, y, \sigma)$ and $\sigma^2\nabla^2 G$ can be found by exploiting a property of the Gaussian. The Gaussian satisfy the heat equation and is given by:

$$\frac{\delta G}{\delta \sigma} = \sigma \nabla^2 G. \tag{3.44}$$

We can approximate $\frac{\delta G}{\delta \sigma}$ with the finite difference between two nearby scaled Gaussians.

$$\frac{\delta G}{\delta \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{3.45}$$

This is also a faster and more stable way of calculating the double derivatives of the Gaussian, remember that the computation of derivatives is sensitive to noise. By substituting equation 3.44 into 3.45 we get:

$$(k-1)\sigma^2\nabla^2 G \approx G(x, y, k\sigma) - G(x, y, \sigma). \tag{3.46}$$

As we can see from equation 3.46 the difference-of-Gaussians separated by a constant factor $k$ have the property of scale invariance and the approximation error will go to zero as $k \to 1$. In [11] they found that the approximation have little impact on the stability of detection and localization of feature points for even differences in scale and suggests using $k = \sqrt{2}$. When finding local maxima and minima of $D(x, y, \sigma)$ we compare each sample point with its eight neighbors in the scale and its nine neighbors in the scale above and below shown in figure 3.13. The sample point is only selected as a keypoint candidate if its higher or lower than all of its 26 neighbors. For all sample points that satisfy this criteria

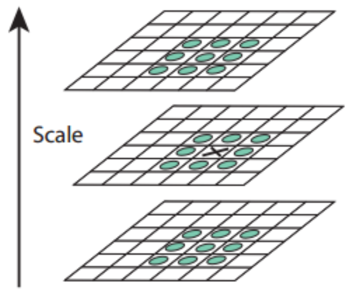


Figure 3.13: The sample point marked with $X$ is compared with its 26 neighbors coloured in green, [11].

we want to fit a quadratic function of the nearby points, depending on location and scale, to find the true extremum in $D(x, y, \sigma)$ at each octave. The second order Taylor expansion of $D(x, y, \sigma)$ where the origin is shifted and evaluated at the location of a keypoint candidate can be written as:

$$D(\boldsymbol{x}) = D + \frac{\delta D}{\delta \boldsymbol{x}}^T \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T \frac{\delta^2 D}{\delta \boldsymbol{x}^2}\boldsymbol{x} \tag{3.47}$$

where the shift from this point is $\boldsymbol{x} = \begin{bmatrix} x & y & \sigma \end{bmatrix}^T$. The derivatives of $D$ can be approximated by convolving $D$ with the Sobel operator. By taking the derivative of 3.47 with respect to $\boldsymbol{x}$ and setting it to zero we find the location of the extremum, $\hat{\boldsymbol{x}}$.

$$\hat{\boldsymbol{x}} = -\frac{\delta^2 D^{-1}}{\delta \boldsymbol{x}^2}\frac{\delta D}{\delta \boldsymbol{x}} \tag{3.48}$$

If any elements contained in the offset vector $\hat{\boldsymbol{x}}$ is larger than 0.5 the keypoint candidate is discarded because the extremum lies closer to another sample point, otherwise it is kept. By adding $\hat{\boldsymbol{x}}$ with the discarded keypoint location we can do the same process until we find a point that produce an offset vector that satisfy the criteria. In [11] they suggest to check the value of $D(\hat{\boldsymbol{x}})$ to reject unstable extrema with low contrast. If the intensity of $|D(\hat{\boldsymbol{x}})| < 7.65$, the sample point is rejected. This is not enough since the difference-of-Gaussian will have a strong response along edges. By looking at the principal curvature at a keypoint we can eliminate the edge response by looking at the ratio between the eigenvalues of the Hessian matrix $\boldsymbol{H}$. The Hessian is computed at the location and scale of the keypoint and is given by:

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{3.49}$$

The eigenvalues of the hessian are proportional to the principal curvature of the surface $D$. If one of the eigenvalues are sufficiently larger than the other indicates that the keypoint is located at an edge. Since SIFT only is concerned by the ratio between the eigenvalues, they avoid calculating the eigenvalues explicitly by borrowing the approach from HCD. The sum and product of the eigenvalues of the Hessian is:

$$\text{Tr}(\boldsymbol{H}) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2, \quad \text{Det}(\boldsymbol{H}) = D_{xx}D_{yy} - D_{xy}^2 = \lambda_1\lambda_2. \tag{3.50}$$

We define that $\lambda_1$ is always equal to the largest eigenvalue of $\boldsymbol{H}$ and the ratio between them is $r$, resulting in $\lambda_1 = r\lambda_2$.

$$\frac{\mathrm{Tr}(\boldsymbol{H})^2}{\mathrm{Det}(\boldsymbol{H})} = \frac{(r\lambda_2 + \lambda_2)^2}{r\lambda_2^2} = \frac{(r+1)^2}{r} \tag{3.51}$$

The function in equation 3.51 only the depends on the ratio of the curvatures and have minima when the eigenvalues are equal. SIFT only choose keypoints that satisfy the criteria:

$$\frac{\mathrm{Tr}(\boldsymbol{H})^2}{\mathrm{Det}(\boldsymbol{H})} < \frac{(r+1)^2}{r} \tag{3.52}$$

with $r = 10$ and eliminates the edge like feautures. Now we have a set of scale invariant and reliable keypoints, the next step is to create a descriptor vector for these keypoints. First we assign each keypoint with a consistent orientation. By choosing the Gaussian smoothed image $L$ that have the closest scale to the keypoint, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ for each image sample $L(x, y)$ is computed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y)^2 + (L(x,y+1) - L(x,y-1))^2} \tag{3.53}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \tag{3.54}$$

The gradient orientations in a region around each keypoint forms an orientation histogram with 36 bins, covering the 360 degree range. The gradient orientation of each sample point in the region is weighted by its magnitude and a Gaussian kernel centered at the keypoint. The scale of the Gaussian is set to be 1.5 times the scale of the keypoint. The highest peak in the orientation histogram diagram is detected and corresponds to the most dominant direction of the local gradients. If we have other peaks within 80% of the highest peak in the histogram, we create new keypoints assigned with their orientation. This means that we can have multiple keypoints at the same location and scale, but with different orientation will contribute to the stability of matching. To improve accuracy of the orientation we fit a parabola to the closest histogram values of each peak, using the same approach as in section 3.3.2. All keypoints have now been assigned with an orientation and will be needed to achieve orientation invariance. The coordinates of the descriptor and the gradient orientations is rotated relative to the orientation of the keypoint. The discriptor is created by first collecting the gradient magnitude and orientation of each sample point around a keypoint in a $16 \times 16$ array, aligned with the keypoint orientation, in $L$ with closest scale to the keypoint. The array is weighted with a Gaussian, centered at the keypoint, of scale 1.5 times the width of the descriptor. The sample array is then divided into 16 regions of size $4 \times 4$ where a orientation diagram consisting of 8 bins is calculated. This results in a $4 \times 4$ descriptor with

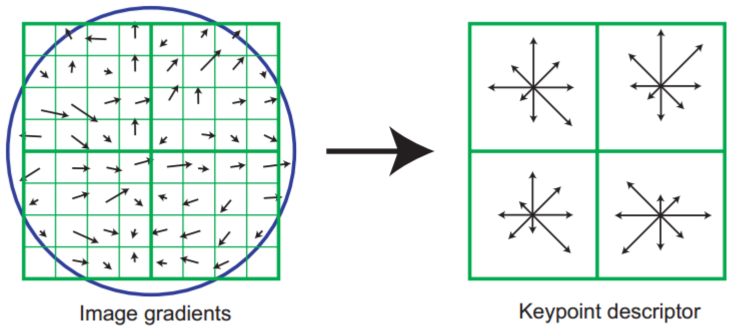


Figure 3.14: On the left we have a $8 \times 8$ sample array of gradient orientations weighted with their magnitude and by a Gaussian. The Gaussian weight is centered at the keypoint and is illustrated by the blue circle. The resulting $2 \times 2$ descriptor on the right have 8 orientation bins each, [11].

8 orientation bins each and experiments done in [11] shows that the matching stability does not improve more by having larger descriptors. The process of creating the descriptor is illustrated in figure 3.14, only with sample array of $8 \times 8$. A feature vector of $4 \times 4 \times 8 = 128$ elements is created for each keypoint. The feature vector is normalized to unit length to be invariant to changes in illumination. The change in image contrast, where all pixels are multiplied by a constant, the gradient magnitude will be proportional to this constant and will be canceled out by the normalization. A change in brightens, where all pixels are added by a constant, will not affect the values of the gradient since we calculate them with pixel differences. The keypoints of same scale orientation are matched against each other by looking at the Euclidean distance between the feature vectors.

#### 3.3.4.3 Speeded Up Robust Features(SURF)

This algorithm was developed after SIFT's success with the approximation of the Laplacian-of-Gaussians and is also a blob detector. One of the drawbacks of SIFT is that it is slow for larger images, the creators of SURF wanted to speed up the process for detection, description and matching by use of simple kernels and integral images. SURF takes the approximation of the Gaussian second order derivatives even to reduce the computation time. SURF uses a fast Hessian detector and only rely on the determinant of the Hessian for selecting the location and scale of a feature, [3]. For a point $\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ in the input image $I(x, y)$ at scale $\sigma$ the Hessian matrix $\boldsymbol{H}(\boldsymbol{x}, \sigma)$ is defined as:

$$\boldsymbol{H}(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix} \tag{3.55}$$

where $L_{xx}(\boldsymbol{x}, \sigma) = \frac{\delta^2}{\delta x^2} G(x, y, \sigma) * I(x, y)$ evaluated at point $\boldsymbol{x}$, and similarly for $L_{xy}(\boldsymbol{x}, \sigma)$ and $L_{yy}(\boldsymbol{x}, \sigma)$ [3]. The Gaussian second order partial derivative with scale $\sigma = 1.2$, lowest scale used by SURF, results in a $9 \times 9$ kernel and can be approximated with the box filters shown in figure 3.15. A box filter is a kernel with equal filter coefficients. The convolution of box filters with the input image

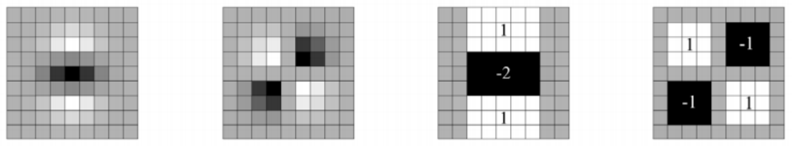


Figure 3.15: The two filters on the left is the Gaussian second order partial derivatives in the $y$ and $xy$ direction and the approximation on the right where the kernel only consist of box filters. The elements that are coloured grey equals to zero, [3].

$I(x, y)$ can be computed quickly by converting the image into a integral image, $I_\Sigma(x, y)$.

$$I_\Sigma(x, y) = \sum_{i \le x} \sum_{j \le y} I(i, j) \tag{3.56}$$

You only need to calculate the integral image once and can be used to calculate the sum of intensities in a rectangular area in the image with constant computation time independent of size. For example a $3 \times 3$ box filter with filter coefficients equal to one convolved with the image evaluated at pixel $\begin{bmatrix} x & y \end{bmatrix}^T$ can be written in terms of the integral image in the following way:

$$\frac{1}{9}(I_\Sigma(x+1, y+1) - I_\Sigma(x-1, y+1) - I_\Sigma(x+1, y-1) + I_\Sigma(x-1, y-1)). \tag{3.57}$$

This concept can be extended when calculating the convolution of the approximated kernels in figure 3.15, that only consists of box filters, with the input image. The $9 \times 9$ approximations of a Gaussian with $\sigma = 1.2$ is denoted $A_{xx}$, $A_{xy}$ and $A_{yy}$. To ensure that the determinant of the Hessian stays close to its true value the expression of $\mathrm{Det}(\boldsymbol{H}_{approx})$ is weighted.

$$\mathrm{Det}(\boldsymbol{H}_{approx}) = A_{xx}A_{yy} - (wA_{xy})^2 \tag{3.58}$$

Where $w$ makes the ratio of the approximated Gaussiam kernels, $A_{xy}$ to $A_{xx}$, the same as the Gaussian kernels, $L_{xy}$ to $L_{xx}$, and is calculated in [3] as:

$$w = \frac{\|L_{xy}(1.2)\|_F \|A_{xx}(9)\|_F}{\|L_{xx}(1.2)\|_F \|A_{xy}(9)\|_F} = 0.912 \approx 0.9 \tag{3.59}$$

The weight depends on the scale of the Gaussian, but is kept constant since it did not have a significant impact on the results in experiments done in [3]. Instead of using a image pyramid, SURF applies approximation of the Gaussian second derivatives of different scale to original image. The use of box filters and integral images allows us to calculate the convolution of the Gaussian second order derivative kernels of any size with the image at the same speed, as mentioned earlier. For each octave SURF use four different scales. The first octave SURF use the filter sizes $9 \times 9$, $15 \times 15$, $21 \times 21$ and $27 \times 27$. The scale of a $9 \times 9$ is $\sigma = 1.2$ and for $27 \times 27$ it is $\sigma = 3 \times 1.2 = 3.6$. For each new octave the filter size increase is doubled, going from 6 to 12 to 24. Also the sampling interval for extracting interest points can be doubled. The size of the kernels for each octave is shown in figure 3.16. The location and scale of interest points are

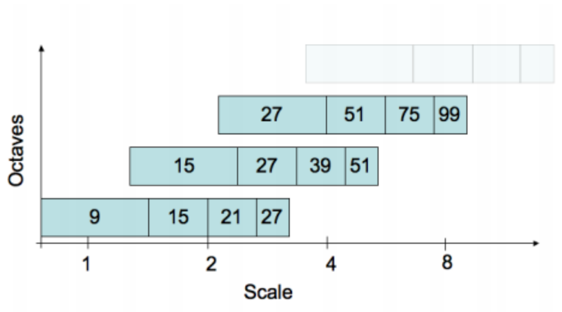


Figure 3.16: The size of the kernels at each octave is plotted against the logarithmic horizontal axis that represents the scale, [3].

found by finding the maxima in a $3 \times 3 \times 3$ neighborhood, in scale and location, of each sample point. They find the local maximum in the same manner as done in SIFT, equation 3.47 and 3.48, but SURF are looking for the location and scale that have a maxima in the determinant of the Hessian. The local maximums are then selected as keypoints. Each keypoint is assigned with an orientation and is based on calculations of the Haar wavelet response. First we calculate the Haar wavelet response, in the $x$ and $y$ direction in the image, in a circular window with radius of 6 times the scale of the keypoint. The Haar wavelet kernel for both directions is shown in figure 3.17. Note that this kernel also only consists of box filters and can thereforee also be calculated with integral images. The size of the Haar wavelet kernel is four times the scale of

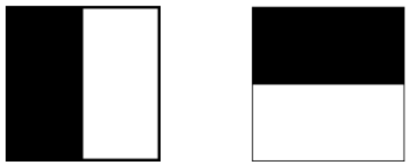


Figure 3.17: The Haar wavelet kernels used to compute the response in the horizontal (left) and vertical direction (right). The black and white regions have filter coefficients of $-1$ and $+1$, respectively, [3].

the keypoint and the sampling step is equal to the scale. When all the responses are computed they are then weighted with a Gaussian, centered at the location, with two times the scale of the keypoint. The local orientation vector is found by summing up the responses, in both directions separately and contained in a vector, in each sector of 60 degrees in the circle. The dominant orientation of the keypoint is set to be the longest local orientation vector. The descriptor is created by first placing a square region centered around the keypoint and is also orientated along its dominant orientation. The square is set to be 20 times the scale. The Haar wavelet response is computed for each point in the square in both direction relative to the orientation, with kernel size two times the scale and weighted with a Gaussian of 3.3 times the scale. We denote the responses in the horizontal and vertical direction as $d_x$ and $d_y$ respectively. Then the square is divided into $4 \times 4$ sub-regions where the responses $d_x$, $|d_x|$, $d_y$ and $|d_y|$ are summed up in each region. The procedure of creating the descriptor is shown

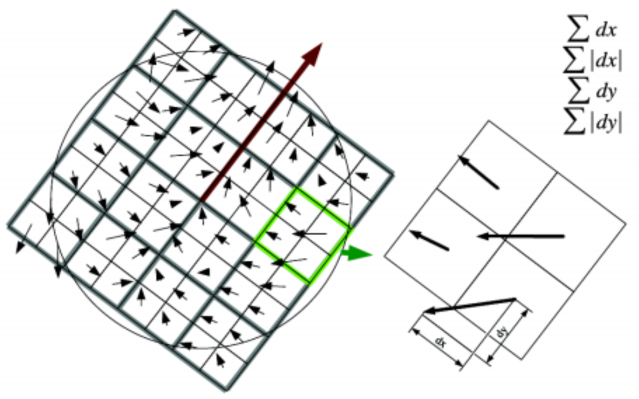


Figure 3.18: The square centered is sentered at the keypoint and oriented along the dominant orientation and divided into $4 \times 4$ sub regions. The $2 \times 2$ subdivision in each region represents the sum of $d_x$, $|d_x|$, $d_y$ and $|d_y|$. The circle around the square illustrates the Gaussian weight, [3].

in figure 3.18. Each sub-division has descriptor vector equal to:

$$\boldsymbol{v} = \left[ \sum d_x \quad \sum |d_x| \quad \sum d_y \quad \sum |d_y| \right]^T, \tag{3.60}$$

by concatenating the vectors from each sub-region we get a descriptor vector of $4 \times 4 \times 4 = 64$ elements for each keypoint. The Haar wavelet response is invariant to brightness changes and by normalizing the descriptor vector into a unit vector it also becomes invariant to contrast changes. Finally the keypoints of same scale, orientation and sign of the Laplacian($\text{Tr}(\boldsymbol{H}(\boldsymbol{x}, \sigma))$) are matched against each other by looking at the Euclidean distance between the descriptor vectors. The reason for dividing the points into groups depending of the sign of the Laplacian is because it distinguishes bright blobs on dark backgrounds and vice versa.

#### 3.3.4.4 Disparity Estimation

The feature based methods don't output the disparity value explicitly, they only return a set of corresponding points between the template and search region. We denote the location of the center of the docking slot in pixels as $\boldsymbol{x}'_m$, assumed given by a tracker in the left image. The vectors of all corresponding points returned by a feature based algorithm in the left and right image is denoted:

$$\begin{bmatrix} \boldsymbol{x}'_{f1} \\ \vdots \\ \boldsymbol{x}'_{fn} \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{x}''_{f1} \\ \vdots \\ \boldsymbol{x}''_{fn} \end{bmatrix} \tag{3.61}$$

We want to find the location of the corresponding point $\boldsymbol{x}''_m$. Since the images are rectified we only need to find the horizontal coordinate $x''_m$. We assuming that the relative horizontal distance between $x'_m$ and $x'_{fi}$ is the same as the distance between $x''_m$ and $x''_{fi}$ for all $i \in [1, \ldots, n]$. Because of wrong matches and that the scene is viewed from two different locations this assumption will not hold, therefore we calculate all possible locations of $x''_m$.

$$\begin{bmatrix} x''_{m1} \\ \vdots \\ x''_{mn} \end{bmatrix} = (x'_m \mathbf{1}_n - \begin{bmatrix} x'_{f1} \\ \vdots \\ x'_{fn} \end{bmatrix}) + \begin{bmatrix} x''_{f1} \\ \vdots \\ x''_{fn} \end{bmatrix}. \tag{3.62}$$

Most of the entries in the vector, $\begin{bmatrix} x''_{m1}, \ldots, x''_{mn} \end{bmatrix}^T$, will be a good estimate of the location of $x''_m$. If we sort the vector in ascending order the good estimates will lie in the middle. Therefore the final value of $x''_m$ is estimated to by the median of the sorted vector and the disparity value is then calculated as $d = x''_m - x'_m$.

# 4 Implementation and Experimental Results

The main objective with this thesis is to conclude on the best algorithm for solving the correspondence problem and also consider the accuracy of the stereo camera for estimating the three dimensional coordinate of the corresponding points. All algorithms are tested on the same set of images and their performance are compared. A description of the field experiment, i.e on how the images were taken, conditions, measurements, etc., is included in this chapter along with a short note on how the different algorithms were implemented. Then the results for each algorithm of the field experiment are presented followed by a discussion.

## 4.1 Implementation

All algorithms were implemented in Matlab, also the stereo rig is calibrated in Matlab. The correlation based methods with use of block matching and sub-pixel estimation was made from scratch. For the feature based methods, both HCD and SURF was implemented by using their respective functions for finding and matching interest points in the Computer System Vision Toolbox [13]. Since SIFT doesn't exist in Matlab, a open source library called VLFeat [18], with SIFT implemented, was used. The outline of the program can be summed up in four stages:

1. The image pair taken by the stereo rig is undistorted, rectified and converted to grey scale.
2. Template and search region are created in the left and right image, respectively, with $Z_{est}$ equal to the measured depth, $\alpha = 0.25$, $X_{obj} = 1.5$, $Y_{obj} = 0.5$ and the location of the docking slot in left image $\boldsymbol{x}'_m$ is given. See section 3.2.1.
3. All algorithms run on the template and search region and returns their estimate of the disparity value.
4. The 3D-coordinate of the docking slot is calculated with triangulation relative to the left camera for each algorithm based on their disparity estimate.

Some small necessary adjustments to the feature based methods were added. When each algorithm have computed and matched interest points between the template and search region, matched corresponding points that deviates more than 3 pixels from the epipolar line were removed. This indicates a wrong match. The value of 3 pixels was experimentally found and is a result of a near perfect rectification. The window size used for HCD depends on $Z_{est}$ and decreases as $Z_{est}$ increases. With a fixed window size the HCD would only work well in a limited range. For SIFT and SURF the number of octaves is restricted depending on the height of the template, because we will not find features larger than the smallest dimension of the image. The height of the template is equal to the search region and is always smaller than the width.

## 4.2 Test Setup and Measurements

Images of the test platform were taken indoor in different locations in a range of $2 - 9[m]$. Since we are in a controlled environment all images were taken under constant illumination which was measured by a lux meter to be $70[lx]$. The stereo rig is stationary for each photo taken. To get acceptable images of the scene under these conditions the exposure time for each camera was set to $2[ms]$. The stereo rig was mounted to a tripod with a rotating head which allowed for taking images in different orientations at each location, see figure 4.1. The head of the tripod was leveled out by adjusting the tripods legs. The position of the left camera's center of projection with respect to the center of rotation of the tripod was estimated through a series of test photos. By only measuring the 3D coordinate of the docking slot with respect to the tripods center of rotation at each location we can easily estimate the left camera coordinates of the docking slot for any configuration of the tripods head. The 3D coordinate of the docking slot relative to the rotation center of the tripod was measured by laser and measuring tape. At each location we took a series of five photos. One straight on, i.e. the baseline of the stereo rig was parallel to the face of the docking slot, and then four images where the head of tripod was rotated $-10$ and 10 degrees around the $x$-axis and $y$-axis separately. To be able to consistently measure the pixel position of the center of the docking slot, a second series of images was taken at the same location and configurations where the center of the slot was marked. This also made it simple to find the true disparity for each image pair which was found by analyzing the images manually afterwards and should give pretty accurate estimate. We were supposed to test the algorithms in different conditions. Images were taken outside as well with varying degree of illumination, but it turned out to be harder than expected to

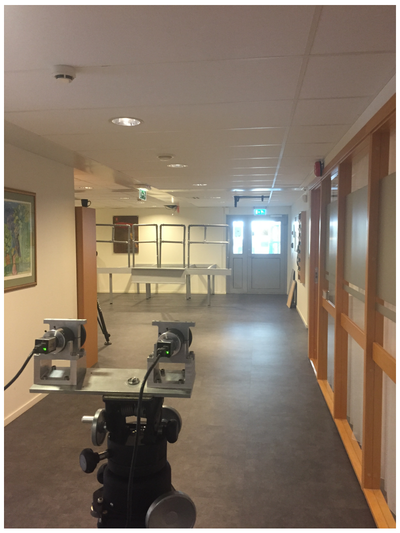


Figure 4.1: Taking images of the platform with the stereo camera at one of many locations.

precisely measure the coordinates of the docking slot and therefor these image series were discarded from further analysis. A field experiment like this is time consuming and unexpected scenarios occur. Often a few different approaches are needed to figure out the best way to get it right. The approach described here is a result of a couple of failed attempts. Even if every thing is going as planned suddenly something unexpected happens. During the experiment one of the camera houses started to unscrew itself from the lens, this was detected after a couple of image series and all of them were discarded because the calibration and rectification were not valid for these images. In total we ended up with 29 image pairs that were not affected by the misadventure.

## 4.3 Experimental Results

All the algorithms were used on the 29 image pairs taken indoors. For each algorithm the estimate of the disparity , three dimensional coordinate of the docking slot and computation time were stored. Also for the feature based methods the number of corresponding points was saved. To give the reader a view of the output of the program, a visual solution of the correspondence problem for each algorithm for the image pair in figure 3.6 are shown in figure 4.3. The template used for this particular image pair can bee seen in figure 4.2.

In each sub figure in 4.3 the values in the yellow boxes represents the estimate of the left camera's coordinate of the docking slot, the line of sight distance and the number of corresponding points are included for the feature based methods. Here the camera coordinate system is oriented such that the positive direction of $X_c$ is to the right, $Y_c$ downward and $Z_c$ inward in the image.

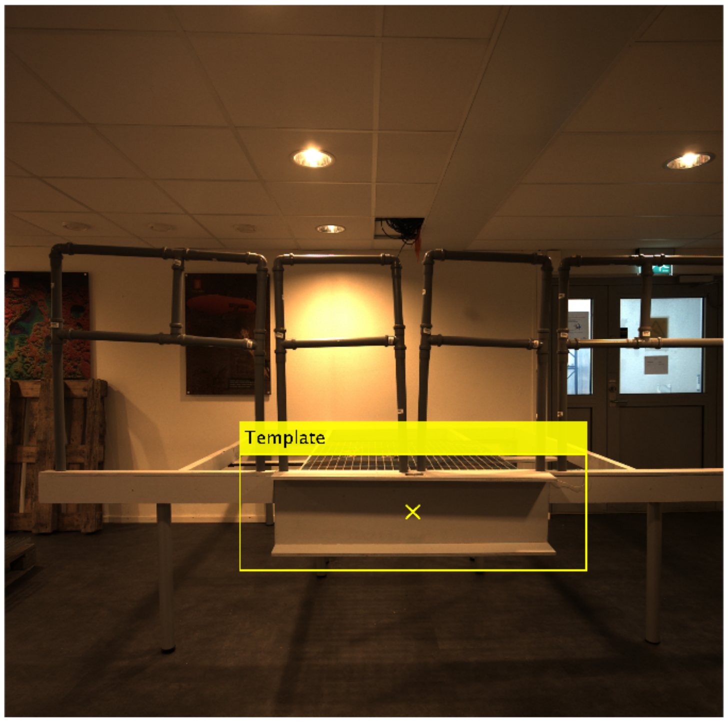
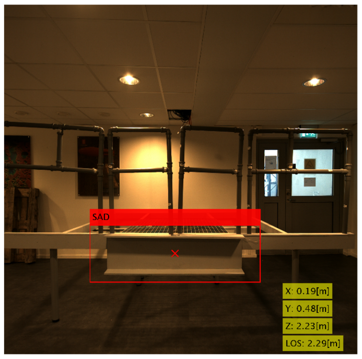
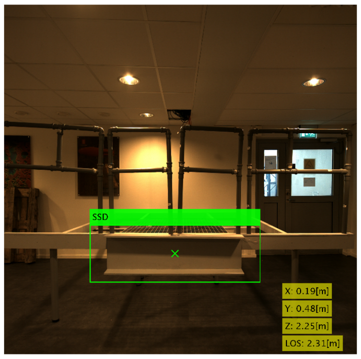


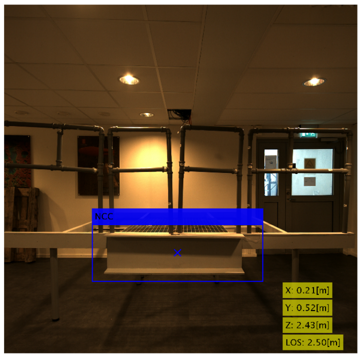
Figure 4.2: The template generated and used for the image pair. The point of interest, assumed given by a tracker, is marked with a cross.



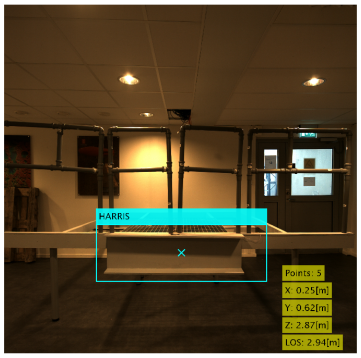
(a) SAD



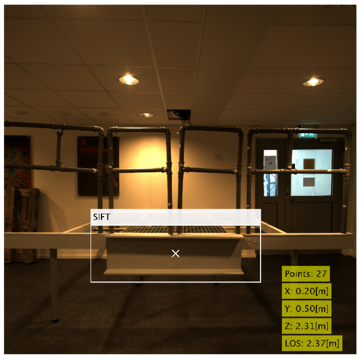
(b) SSD



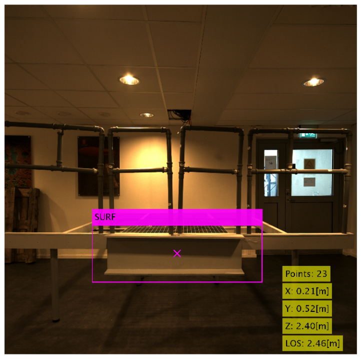
(c) NCC



(d) HCD



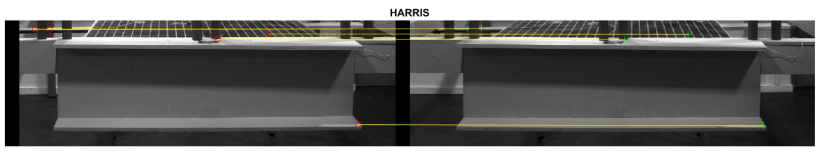
(e) SIFT



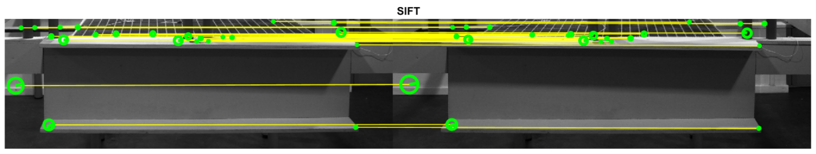
(f) SURF

Figure 4.3: The solution of the correspondence problem of each algorithm is marked with a cross. The measured values of the docking slot location is $X_c = 0.18[m]$, $Y_c = 0.52[m]$ and $Z_c = 2.26[m]$.

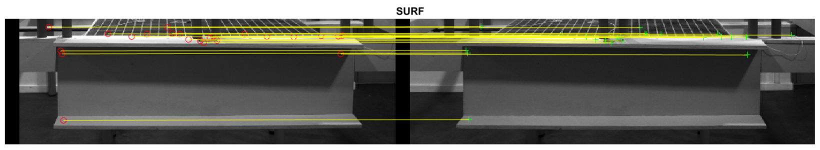
For the feature based methods we have plotted the all corresponding feature points found by each algorithm between the template and the search region in figure 4.4. As you can see from the from the sub figures all the corresponding points lie on the same horizontal epipolar line. For SIFT and SURF we get a lot of matches on steel grating floor panels and the fence poles. There will most likely occur some ambiguous matches and this is the reason why we estimate the disparity as in 3.3.4.4.



(a) HCD



(b) SIFT



(c) SURF

Figure 4.4: Corresponding feature points between the template and search region found by the three different feature based methods.

SIFT and SURF will always find more points than HCD since they search for features in many scales, HCD only use one window size depending on depth for each image as mentioned earlier. The number of feature points will decay as we move away from the platform since the template and search region gets smaller. Because of the outline of the program the correlation based methods will always output an estimate of the disparity, but the feature based methods need to find at least one set of corresponding feature points to be able to estimate the disparity. Throughout the experiment both SURF and HCD did not find any feature points for one image pair each. This was in the depth of $6 - 8[m]$. When comparing the performance of the algorithms these images were removed, but the failure of HCD and SIFT are taken into consideration for the final evaluation. For the following plots in figure 4.5-4.10 we have calculated the absolute error of the three dimensional coordinate and the disparity estimate between the measured values for each algorithm. The errors are defined in the following way:

$$\Delta X = |X_a - X_t|, \quad \Delta Y = |Y_a - Y_t|, \quad \Delta Z = |Z_a - Z_t|, \quad \Delta d = |d_a - d_t| \tag{4.1}$$

where the subscript $a$ and $t$ represents the estimate of the algorithm and the true measured value, respectively. All the errors are plotted against the measured depth. To remove some noise from the plots the results are smoothed with a moving average filter with a span of 5. This is reasonable since the images are taken close to each other in depth and the evolution of the errors as the depth increases are shown more clearly. The axis limits and aspect ratios are equal for each error plot for the different algorithms so the results are easier to analyze. Peaks in $\Delta d$ will cause a peak in the $\Delta Z$ at the same depth as result of equation 3.1. For $\Delta X$ we can see that it is a peak around $3.8[m]$ that is not correlated with the disparity error for most of the algorithms and clearly indicates a measuring error, same for $\Delta Y$ at $5.8[m]$. The evolution of $\Delta X$ and $\Delta Y$ should be the same since we have a square image sensor. For all the methods the plot for $\Delta Y$ looks pretty much the same. This is natural since the estimate of $Y$ is mostly dictated by the vertical pixel coordinate of the interest point.

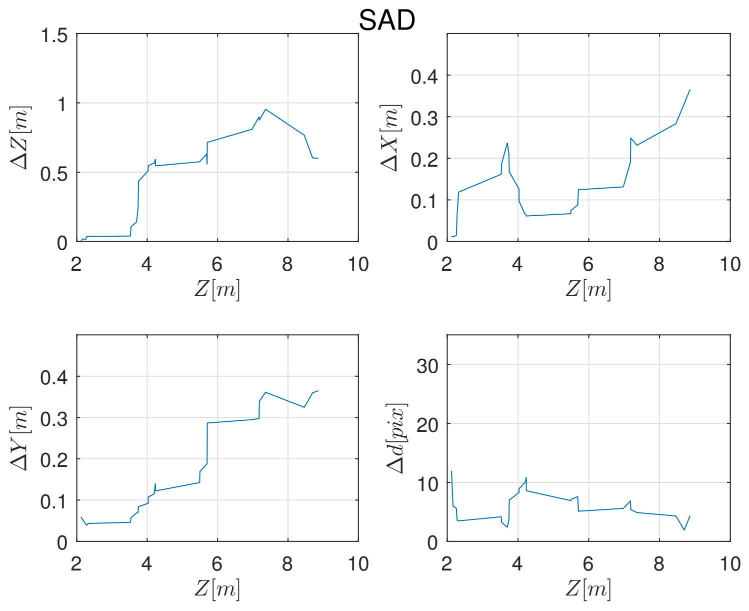


Figure 4.5: SAD

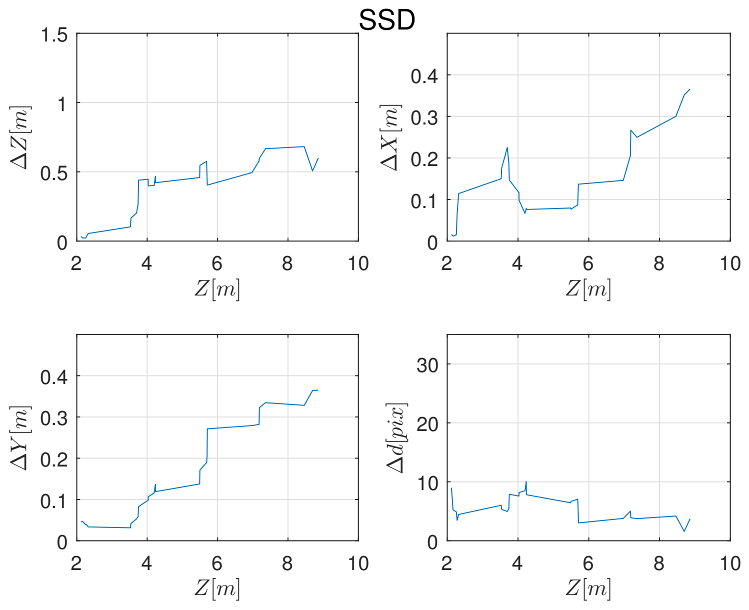


Figure 4.6: SSD

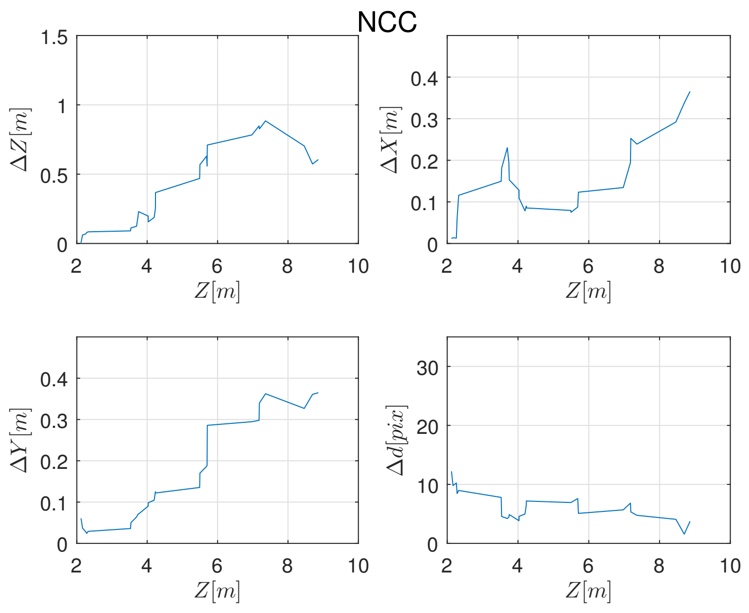


Figure 4.7: NCC

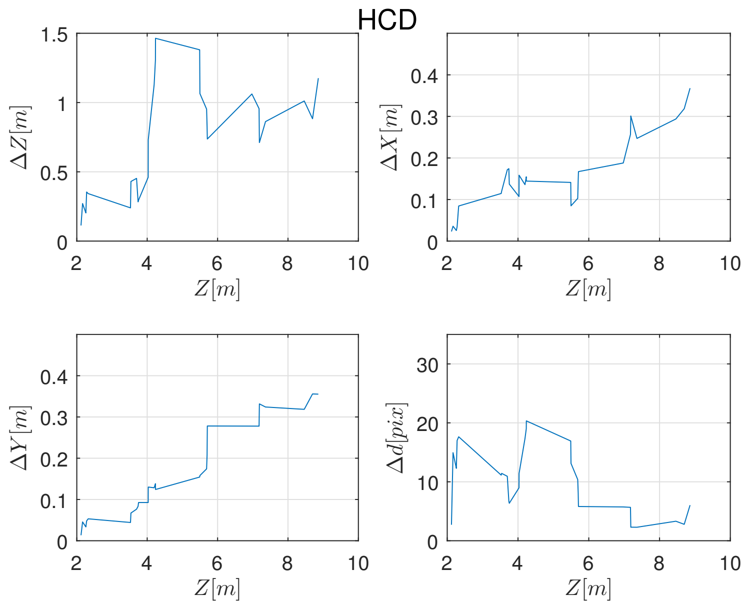


Figure 4.8: HCD

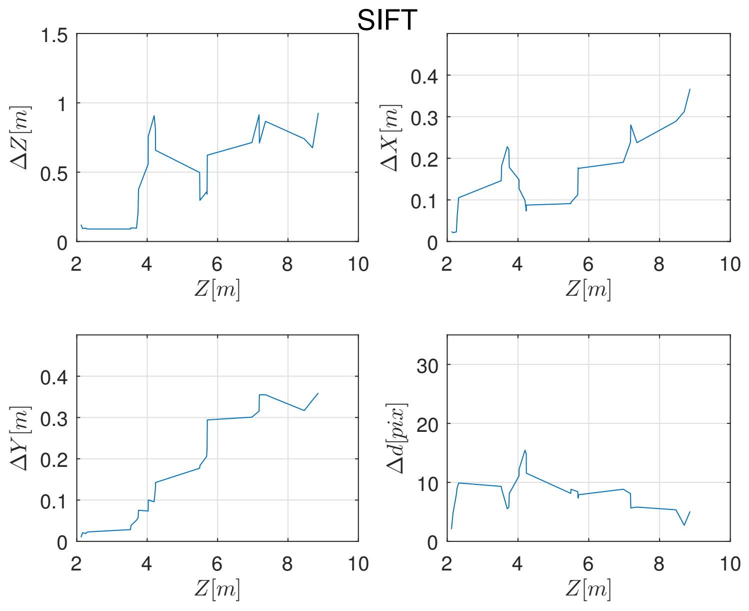


Figure 4.9: SIFT

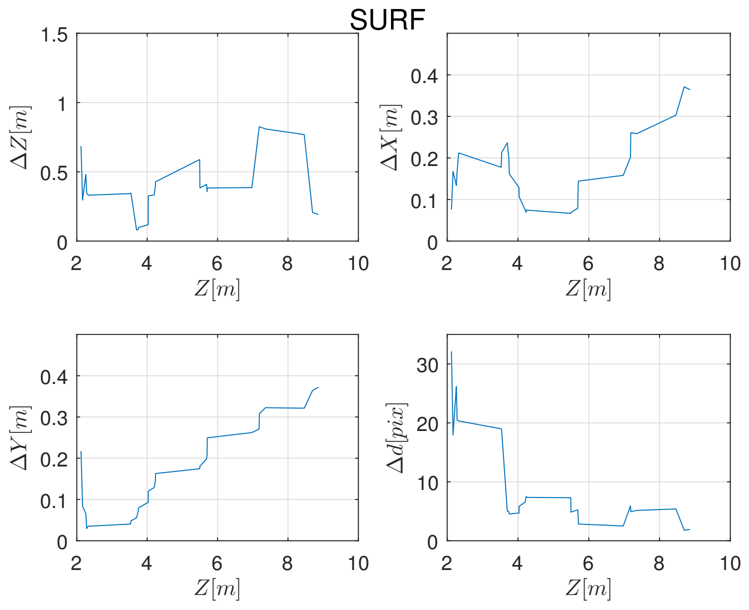


Figure 4.10: SURF

In figure 4.11 we have plotted the computation time for each algorithm for solving the correspondence problem against the measured depth where the image pair was taken. The machine used for this experiment have an Intel Core i7 processor running at 2.85GHz. The size of the template and search region depends on depth and we can clearly see that the computation time decreases as we move away from the platform. For HCD the computation time is almost constant and high because the ratio between the dimension of the window and template is around the same for all depths. At close range, $2 - 3[m]$, we can see that SURF are twice as fast than SIFT as expected when the size of the template is large, but SIFT turns out to be faster as the depth increases. The correlation based methods have pretty much the same computation time, where NCC is a bit faster than the others at close range. Note that these time estimates only consider stage 3 in the description of the program in subsection 4.1. The total computation time for stage 2 and 3 is around $0.09[s]$. For stage 1 the computation time was around $4.5[s]$ and is of course not good enough when running in real time. The results presented in [14] they computed stage 1, the process of undistorting and rectifying a image pair of same dimensions, in $12[ms]$ using a Intel Core i5 processor running at 3.3GHz. All the calculations will be speeded up even further using data parallelism and a graphics processing unit, so running in real time is indeed achievable.

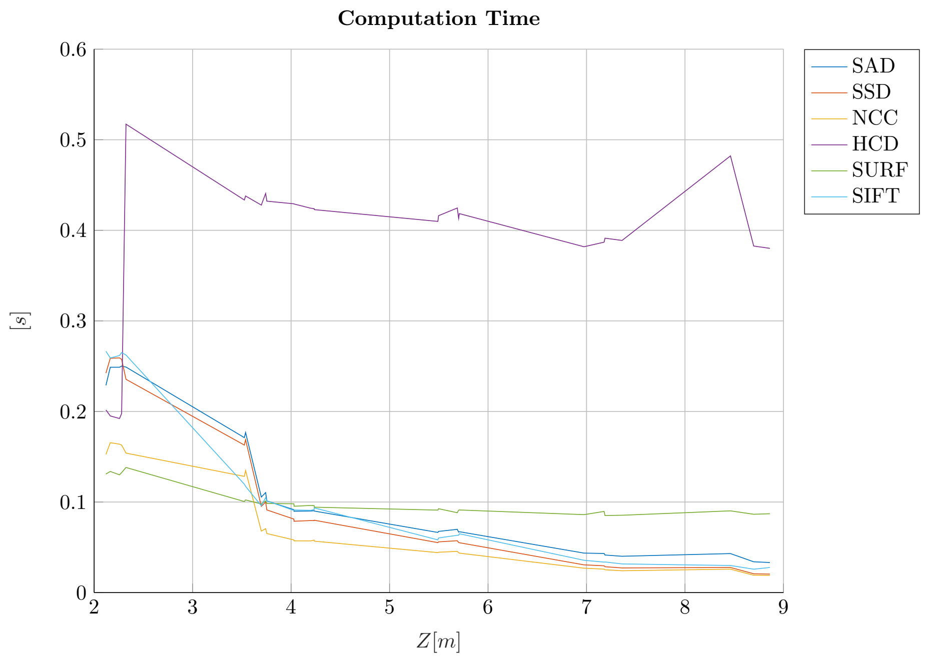


Figure 4.11: Computation time for each algorithm at various depths.

To summarize the performance of each algorithm for this experiment the mean and standard deviation of $\Delta d$ in pixels and expected computation time for all methods in seconds are calculated in table 4.1.

| Alg. | $E[\Delta d]$ | $\sigma_{\Delta d}$ | $E[t]$ |
|---|---|---|---|
| SAD | 5.92 | 4.87 | 0.11 |
| SSD | 5.59 | 4.82 | 0.10 |
| NCC | 6.15 | 4.36 | 0.07 |
| HCD | 10.15 | 10.17 | 0.39 |
| SIFT | 8.20 | 5.82 | 0.10 |
| SURF | 9.03 | 13.62 | 0.10 |

Table 4.1: Mean and standard deviation of $\Delta d$ in pixels and the expected computation time in seconds for each algorithm.

## 4.4 Discussion

Solving for the corresponding point of the docking slot are not an easy task. There are no known unique signs or hallmarks on the platform that are easy to detect. As we can see from the results, especially from table 4.1, the feature based methods really struggle to solve the correspondence problem mostly because they get a lot of ambiguous matches. HCD and SURF are neither accurate or consistent in their disparity estimate. SIFT had the overall best performance of the feature based methods, but cannot match the accuracy of the correlation based methods. The biggest difference between the two groups of algorithms are amount information they extract in the template and search region for solving the correspondence problem. The correlation based methods use the whole template when matching where the feature based methods only use parts of the template where the feature points are found. This is one of the main reasons, along with the problem of ambiguous matches, that the correlation based methods outperforms the feature based methods in accuracy of the disparity estimate. Even if the correlation based methods use block matching they have around the same computation time as SIFT and SURF. All in all this field experiment have shown that the feature based methods are not suited for this particular application. When selecting the best algorithm we are only left with the correlation based methods. They did all performe well where SSD was the most accurate and NCC was the fastest by a small margin according to the values in table 4.1. Since they all have around the same computation time, and all of them can run in real time, the most important quality of the best algorithm is its accuracy of the disparity estimate. On that premise the SSD was the best algorithm overall for solving the correspondence problem. As mentioned earlier this field experiment was done indoors so the algorithms are not compared in different light conditions, but score for SAD, SSD and NCC are calculated with normalized intensity values and should therefor be invariant to illumination differences. The outcome may change if a new field experiment was executed outside in different weather conditions, but we can almost be certain that the correlation based methods will perform better than the feature based. To improve the accuracy of the stereo rig we did come up with a optimal theoretical value of the baseline for this application in 3.2. By choosing SSD as the algorithm for computing the disparity we will have a expected disparity error of $\Delta d = 5.59[pix]$. We have again plotted the expected depth error in figure 4.12 to see the true and potential accuracy of the stereo camera rig with the original and theoretical value of the baseline when using SSD. As we can some from

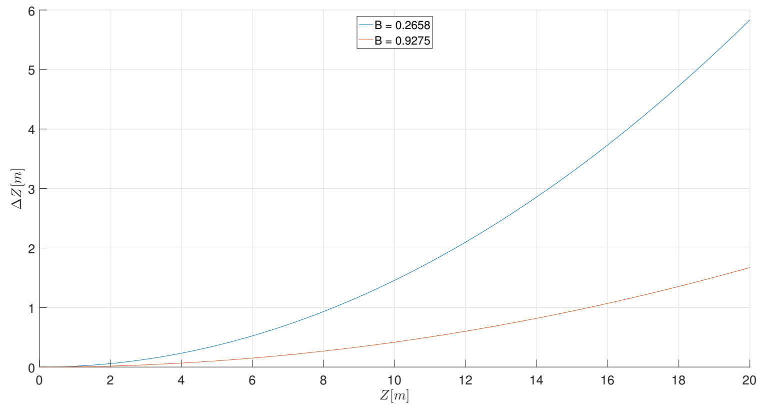


Figure 4.12: The expected depth error at different ranges for two different baseline lengths $B = 256.8[mm]$ and $B = 927.5[mm]$ with the parameters: $f = 1441.8[pix]$ and $\Delta d = 5.59[pix]$.

figure 4.12 that increasing the baseline to the theoretical optimal value we get a really good accuracy for the stereo rig for the whole range. In 3.2 we estimated the expected disparity error to be $\Delta d = 0.26$ for a perfect match, using SSD the expected disparity error is more than 20 times higher and the difference in accuracy can been seen by comparing figure 3.5 with 4.12. This program need an estimate of depth to work, see section 3.2.1. The idea was that the previous measurement of depth should be used as the estimate of depth for the next iteration. Increasing the baseline allows us to use a smaller value of $\alpha$ because the measurements are more accurate. This means that the disparity range we search over for the correlation based methods gets shorter and the computation time will decrease. Using the optimal base length we could set a reasonable value for $\alpha$ to be in the interval $\alpha \in [0.1, 0.15]$. This is only a theoretical value ans should be tested.

# 5 Conclusion

Two groups of algorithms, correlation and feature based methods, have been tested for solving the correspondence problem. The feature based methods, HCD, SIFT and SURF, were found not to be a suitable approach for this application. There may be more clever and robust ways of using them, but in any case the set of matched feature points will be influenced by ambiguous matches and it will be difficult to accurately estimate the disparity. The correlation based methods, SAD, SSD and NCC, proved to be the most accurate and consistent algorithms based on the results of the field experiment. They extract more information of the scene and will be more robust against ambiguous matches. When deciding on the superior algorithm the accuracy of the disparity estimate is the most important property since it dictate the accuracy of the stereo camera system's depth estimate. Sum of absolute differences, SSD, did have the overall best accuracy in the field experiment and was concluded to be the best algorithm for this application. The implementation is simple, but it works well and the computation time of the whole program will be low enough to be run in real time. To improve the accuracy of the stereo camera system even further we have proposed a theoretical optimal length of the baseline in 3.2. The algorithms were not tested in different light condition and the theoretical length for the baseline have not been used. Some suggestion for future work:

- Evaluate the performance of the correlation based methods in different light conditions.
- Experiment with different baseline lengths around the optimal theoretical value.
- Run the program in real time and experiment with different values of $\alpha$.

# 6 Appendix

## 6.1 Homogeneous Coordinates

Some properties of homogeneous coordinates, also called projective coordinates [9].

### 6.1.1 Cartesian → Homogeneous

$$P = (x, y) \in \mathbb{R}^2 \to \tilde{P} = (x, y, 1) \in \mathbb{P}^2 \tag{6.1}$$

$$\mathbb{P}^n := \mathbb{R}^{n+1} \setminus \{0\}$$

The origin is represented by $\begin{bmatrix} 0_1 \cdots 0_n & 1_{n+1} \end{bmatrix} \in \mathbb{P}^n$

### 6.1.2 Homogeneous → Cartesian

$$\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z}) \to P = (x, y) \tag{6.2}$$

where

$$x = \frac{\tilde{x}}{\tilde{z}}, \quad y = \frac{\tilde{y}}{\tilde{z}}$$

### 6.1.3 A line in homogeneous form

In homogeneous coordinates we can represent a line and a point as $\tilde{l} = (l_1, l_2, l_3)$ and $\tilde{p} = (\tilde{x}, \tilde{y}, \tilde{z})$, respectively. The point equation of line is given by:

$$\tilde{l}^T \tilde{p} = 0 \tag{6.3}$$

The advantage of this is that you can form a line of any slope.

### 6.1.4 Line joining points

If you have two points $\tilde{p}_1 = (a, b, c)$ and $\tilde{p}_2 = (d, e, f)$ in homogeneous coordinates. The homogeneous line that goes trough them is:

$$\tilde{l} = \tilde{p}_1 \times \tilde{p}_2 \tag{6.4}$$

### 6.1.5 Intersecting lines

The point of intersection of two arbitrary homogeneous lines, $\tilde{l}_1 = (a, b, c)$ and $\tilde{l}_2 = (d, e, f)$ is:

$$\tilde{p} = \tilde{l}_1 \times \tilde{l}_2 \tag{6.5}$$

## 6.2 Singular Value Decomposition (SVD)

The singular value decomposition is a factorization of a real or complex matrix [16]. The singular value decomposition of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is given by:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{6.6}$$

where

- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$, the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ make up the columns of $\boldsymbol{U}$. We say that the columns of $\boldsymbol{U}$ are the left singular vectors of $\boldsymbol{A}$. The matrix $\boldsymbol{U}$ is orthogonal, $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}_m$.

- $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$ make up the columns of $\boldsymbol{V}$. We say that the the columns of $\boldsymbol{V}$ are the right singular vectors of $\boldsymbol{A}$. The matrix $\boldsymbol{V}$ is orthogonal, $\boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}_n$.

- $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$, is a rectangular diagonal matrix. The elements along the diagonal are the square root of the eigenvalues of both $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^T$ in descending order. The elements along the diagonal of $\boldsymbol{\Sigma}$ are called the singular values of $\boldsymbol{A}$.

## References

[1] Digital focal plane. URL `http://www.uglyhedgehog.com/t-99555-2.html`.

[2] D. G. Bailey. Sub-pixel estimation of local extrema. Technical report, Institute of Information Sciences and Technology, Massey University, November 2003.

[3] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. Technical report, ETH Zurich, 2008.

[4] R. Collins. Camera projection. URL `http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf`. Lecture 12 for course CSE486, Penn State University.

[5] W. Förstner and B. P. Wrobel. *Photogrammetric Computer Vision: Statistics, Geometry, Orientation and Reconstruction*, volume 11. Springer International Publishing, 1 edition, 2016.

[6] H. P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. Technical report, Department of Civil and Environmental Engineering, Duke University, May 2016.

[7] K. Grauman. Epipolar geometry and stereo vision, 2009. URL `http://www.cs.utexas.edu/~grauman`. University of Texas at Austin.

[8] C. Harris and M. Stephens. A combined corner and edge detector. Technical report, Plessey Research Roke Manor, United Kingdom, 1988.

[9] Homogeneous coordinates. Homogeneous coordinates — Wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/wiki/Homogeneous_coordinates`.

[10] IDS. Gige ueye cp industrial camera. URL `https://en.ids-imaging.com/store/products/cameras/gige-cameras/ueye-cp.html`.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. Technical report, Computer Science Department, University of British Columbia Vancouver, January 2004.

[12] N. Lucia. Toolbox for camera - self calibration. Technical report, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, September 2009. URL `http://www.posterus.sk/?p=2071`.

[13] Matlab. Computer vision system toolbox. URL `https://se.mathworks.com/products/computer-vision.html`.

[14] C. D. Pantille, I. Haller, M. Drulea, and S. Nedevschi. Real-time image rectification and stereo reconstruction system on the gpu. Technical report, Technical University of Cluj-Napoca, January 2011.

[15] S. Se and N. Pears. Passive 3d imaging. In *3D Imaging, Analysis and Applications*, chapter 2, pages 35–89. Springer-Verlag London, 2012.

[16] Singular Value Decomposition. Singular value decomposition — Wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/wiki/Singular_value_decomposition`.

[17] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer-Verlag London, 1 edition, 2011.

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

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