Stereo vision using local methods for autonomous ferry

TTK4550 - Engineering Cybernetics, Specialization Project, 10th of January 2020

Trine Ødegård Olsen

NTNU
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

Sensors in autonomous vehicles are essential for situation awareness. By perceiving obstacles and other vehicles, fusion of sensors can track and predict motion to avoid collisions. Range information can be achieved with active sensors like LiDAR or RADAR, or passive sensors like cameras in a stereo configuration. Cameras have an increasing pixel resolution and possible long-range detection, which are indispensable for advanced collision avoidance and driver assistance systems.

This thesis explores how stereo cameras combined with computer vision can calculate range information and create a 3D-map of the environment. Theory on stereo vision and associated algorithms are given. An experiment was conducted with a fixed scene and ground truth measured by a LiDAR calibrated with the stereo cameras. Nine local stereo algorithms, both correlation- and featured-based, were implemented to create depth-maps; SAD, SSD, NCC, RT, CT, HCD, SIFT, SURF, and ORB. They were all tested against the ground truth, giving an average error on 3 centimeters. Five of the algorithms were discarded, while four algorithms remained with the potential to be implemented on the autonomous ferry prototype ”MilliAmpere”. Overall, CT yields the best result considering both accuracy and computation cost.

Preface

The thesis is part of the specialization project in the fifth year of the author's Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The overall goal is to specialize in selected areas based on scientific methods, collect supplementary information based on literature search and other sources. Combining the latter with own knowledge into a project report to extend knowledge into the chosen subject area.

The project was proposed by my supervisor Edmund F. Brekke and co-supervisors Annette Stahl and Øystein Kaarstad Helgesen, each of them working under the Department of Engineering Cybernetics. It is part of the Autoferry research project, whose aim is to develop cutting-edge technology for autonomous all-electric passenger ferry for urban water transport. This report specializes in computer vision, more precisely in depth estimation using binocular stereo cameras.

The first part of the thesis is written in collaboration with Lina Theimann, who writes about LiDAR and Stereo calibration; *Comparison of depth information from stereo camera and LiDAR*. Together, implementing, calibrating, and creating the stereo vision setup. This includes Chapter 2, Section 4.1 and 4.3. Together we would like to thank our supervisors for their help and knowledge throughout the project. We thank Glenn Angel and all the technical staff at ITK for their help with the stereo setup. We would also like to thank Stefano Brevik Bertelli for always being helpful in times of need, and Egil Eide for taking the time to show us the ferry MilliAmpere.

Table of Contents

Abbreviations

SAD	=	Sum Absolute Difference
SSD	=	Sum of Squared Difference
NCC	=	Normalized Cross Correlation
RT	=	Rank Transform
CT	=	Census Transform
HCD	=	Harris Corner Detector
SIFT	=	Scale Invariant Feature Transform
SURF	=	Speeded Up Robust Feature
ORB	=	Oriented FAST and Rotation Aware BRIEF
BF	=	Brute-Force
FLANN	=	Fast Library for Approximate Nearest Neighbors
FOV	=	Field Of View

Chapter 1

Background

The project is motivated by Gizmodo's article ”Elon Musk Was Right: Cheap Cameras Could Replace LiDAR on Self-Driving Cars, Researchers Find” [21]. At Tesla's Autonomy Day 2019, Elon Musk stated that stereo vision is the new LiDAR, and subsequently, stereo vision is becoming an extending research area for autonomous vehicles.

> *“LiDAR is a fool's errand. Anyone relying on LiDAR is doomed. Doomed! They are expensive sensors that are unnecessary”*
>
> —Elon Musk, Tesla's Autonomy Day 2019

Today, most major developers and startups of autonomous cars use LiDARs in combination with cameras and radar. Musk and Tesla, on the other hand, is unique in the sense that they only use cameras and radar. Light Detection and Ranging (LiDAR) is an active remote sensing system, giving an exact image of the world. With distance accuracy of around one centimeter on long distances, it acquires reliable depth information on objects on sufficient size and range. It is trivial to isolate objects, works independent of ambient light, and is robust against interference [43]. Hence, is there any support in Musk's statement, can cameras substitute the advantages of using LiDARs?

1.1 Problem description

The primary goal of the specialization project is to explore how stereo vision can improve visual vessel detection. Compared to LiDARs, passive optical recognition like cameras is in the low-price range. Combined with their increasing pixel resolution and possible long-range detection, they are indispensable for advanced driver assistance systems [43]. The object is to prepare a stereo vision setup for visual vessel detection, and look into how the resulting 3D world map differs with various algorithms. Overall, the goal is to achieve an understanding of the setup and calibration needed, and to end up with one or several algorithms which are applicable for stereo vision on the autonomous ferry prototype ”Milliampere”. The tasks include

- Familiarize with necessary literature
- Binocular stereo setup. Physically fixing cameras and optics relative to each other
- Get the cameras up and running with driver, and ROS
- Performing stereo camera calibration
- Perform rectification and undistort the images
- Familiarize with algorithms for creating disparity map
  - Correlation based matching
  - Feature based matching
- Creating a Ground Truth environment by constructing a measured 3D scene
- Testing the algorithms on the scene as a Ground Truth

1.2 Literature review

Detection of the world is an essential task in autonomous vehicles. A key ingredient for vision-based 3D object detection is a reliable depth estimation approach. Today 3D approaches based on a cheap camera or stereo cameras have resulted in drastically lower accuracy compared to solutions based on LiDAR. An expanding number of researchers are trying to reduce this gap.

Wang, Chao, Garg, et al. [47] presented in June 2019 an article that supports Musk's view on stereo vision. The article is one among others, bridging the gap between image- and LiDAR-based 3D object detection. They improved the accuracy of visualizing surroundings by stereo cameras, putting their approach on par with LiDAR solutions. By changing the representation of 3D information from cameras, the image-based performance on 30meters was improved from state-of-the-art 22 percent accuracy to 74 percent.

Related papers delve into sensor fusion and how a fusion of LiDAR and stereo vision can complement the sensors' weaknesses. LiDARs are expensive, range limited, and is for example, not able to detect traffic lights. Computer vision contrarily is not robust considering reliability, variations in light, and the high CPU usage [43]. In January 2019, Sadakuni et al. published the article ”Construction of Highly Accurate Depth Estimation Dataset Using High Density 3D LiDAR and Stereo Camera” [41]. By sensor fusion, they integrate point clouds based on the relative positioning of the two sensors to compensate for the LiDAR's short ranging distance and to get depth information corresponding to each pixel of the stereo image. Both LiDAR and stereo vision have an extensive amount of research, and the choice of using either LiDAR, cameras, or both is more a question about which technology one can reliably bet will be best in the future [43].

Computer vision as a research area is more than 50 years old, and today improving depth estimates for stereo cameras is a flourishing topic with numerous publications. Articles used throughout the project like [24], [16], [36], [48], [44], [42] and [3] are only a narrow representation of relevant publications in the stereo matching problem. However, the majority of articles looking into the correlation problem tend to focus on controlled environments detecting objects on a distance around one to two meters. Hence, long-distance object depth estimate with binocular cameras is something that needs to be further examined. This report will look into the stereo setup required and algorithms for estimating depth on long distanced objects in challenging areas.

Literature that examines various algorithms was used as a base for solving the correspondence problem. This includes [12], [33], [18] and [14], all the latter subordinated either stereo matching techniques or stereo vision disparity map. They examine different algorithms for various application areas. In general, the methods can be classified into local and global approaches, where global methods achieve more accurate disparity maps, while the local methods are more efficient. The experiment is meant to be implemented on an autonomous ferry, having higher real-time demands than accuracy. Taking the four latter papers into account, it appears that local methods have the most suitable approaches for creating a disparity map in use on the ferry, achieving lower computational complexity. Local methods are usually divided into two categories, namely Correlation based methods, and Feature based methods. For the project, the most promising algorithms were chosen, expanded with some acknowledged ones, to end up with nine different algorithms to be analyzed.

- Correlation Based Methods
  - Sum of Absolute Differences [44]
  - Sum of Squared Differences [27]
  - Normalized Cross Correlation [42]
  - Rank Transform [48]
  - Census Transform [11]
- Feature Based Methods
  - Harris Corner Detector [9]
  - Scale Invariant Feature Transform [23]
  - Speeded Up Robust Feature [2]
  - Oriented FAST and Rotation Aware BRIEF [38]

The most promising ones before implementation would be the Sum of Absolute Differences, Census Transform, and Oriented FAST and Rotation Aware BRIEF. SAD is known to yield good results compared to its simplicity. CT for its performance under challenging lighting conditions and ORB because it is known to be robust, accurate, and real-time.

1.3 Report outline

This report is divided into seven chapters. Following the background, the theory of camera and stereo vision is presented in Chapter 2, subsequently the foundation of the algorithms used in Chapter 3. Chapter 4 looks into the hardware and software used, and some theory and result of the chosen setup and calibration. In Chapter 5, the experiment performed is presented and discussed, while Chapter 6 summarizes the result. The final chapter state a conclusion and suggest further work.

Chapter 2

Theory

The chapter aims to provide the theoretical principles and theory used throughout the report. The focus is on the geometric camera model, epipolar geometry, calibration, and the creation of a disparity map. A disparity map is an image indirectly containing the distance to the surface of objects from a given viewpoint. The theory is mainly based on [7], [10] and [40].

2.1 Pinhole model

The pinhole model is the most commonly used geometric camera model. The mathematical model describes the correspondence between real-world 3D points and their projection onto the image plane. In an ideal pinhole camera, the camera aperture is described as a point; no lenses used to focus.

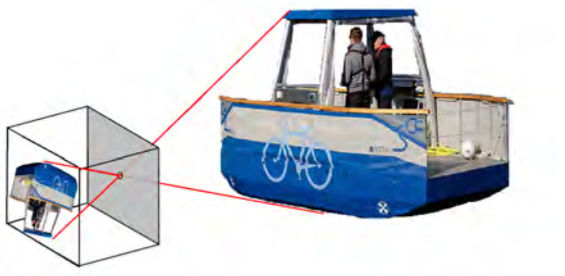


Figure 2.1: Pinhole camera model

Figure 2.1 displays the camera body as a rectangular box. The camera film, or image plane which captures the light emitted through the pinhole, is placed in the rear inside of the camera body. The pinhole let light rays from only one direction through at a time. This filter out the light rays coming from the wrong direction, and the light rays will move through the hole in a straight line. So light from the bottom of the object will end up at the top of the image plane, and likewise, light from the top of the scene will end up at the bottom. In the images captured, everything will be in focus. Hence, to avoid blur in the image, it is necessary to have prolonged exposure time or absence of moving objects. In practice, cameras are equipped with a lens to produce useful images. The pinhole is mathematically convenient, and the geometry of the model is an adequate approximation of the imaging process.

2.1.1 Camera parameters

The camera parameters are what describe the relationship between the camera coordinate system and real-world coordinates. They can be divided into two groups; intrinsic and extrinsic parameters. The intrinsic parameters represent the focal length and the optical center of a camera, described as a mapping between camera coordinates and pixel coordinates in the image plane. The extrinsic matrix represent the position and orientation of the camera in real-world coordinates, and it is described as a mapping between the camera coordinates and the real-world coordinates. The image plane is parallel to the camera

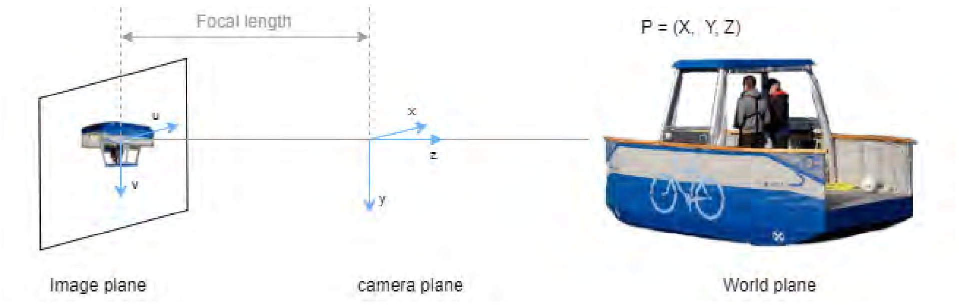


Figure 2.2: Intrinsic and extrinsic geometry

plane at a fixed distance from the pinhole. This distance is called the focal length, f. The geometry is shown in Figure 2.2.

When representing geometric transformations, it is useful to think in terms of projective geometry. Homogeneous coordinates in the projective space is an alternative way to describe Cartesian coordinates in Euclidean space. The homogeneous coordinates allow affine transformations and projective transformations to be represented by a matrix easily. As well, they represent points at infinity using finite coordinates. Mapping a Cartesian vector to a homogeneous space and homogeneous to Cartesian is given by (2.1) and (2.2) respectively.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \longrightarrow \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \longrightarrow \mathbf{x} = \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix} \tag{2.2}$$

Intrinsic parameters

The intrinsic parameters describe the transformation between camera coordinates and pixel coordinates in the image frame. The world coordinates are assumed given in the camera coordinate frame, where the origin is in the optical center. Considering the reference frames showed in Figure 2.2, the mapping between the Euclidean 3D space, (X, Y, Z), to Euclidean 2D space (u, v) is expressed as follows

$$u = f\frac{X}{Z} \quad v = f\frac{Y}{Z}$$

The corresponding mapping in homogeneous coordinates is thus given by

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The homogeneous mapping does not take into consideration that the origin of the image plane may not coincide with the principal point, the point where the gray line in Figure 2.2 meets the image plane. To compensate, the mapping is adjusted to

$$u = f\frac{X}{Z} + p_x \quad v = f\frac{Y}{Z} + p_y$$

and equivalent to homogeneous coordinates:

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The point $(p_x, p_y)^T$ are the coordinate of the principal point and

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

is the camera calibration matrix or intrinsic camera matrix.

The pinhole model assumes pixels to be square, which may not be the case for other camera models like e.g., CCD cameras. In CCD cameras, there is the possibility of having non-square pixels. This may cause an unequal scale factor in each direction if image coordinates are measured in pixels. To account for this, m_x and m_y are defined as number of pixels per unit distance in image coordinates, in x and y directions respectively.[13]

The adapted intrinsic camera matrix is given by

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} \alpha_x &= m_x f \\ \alpha_y &= m_y f \\ x_0 &= m_x p_x \\ y_0 &= m_y p_y \end{aligned} \tag{2.4}$$

where (α_x, α_y) represents the focal length in pixel units, and (x_0, y_0) represents the principal point in pixel units. The number s is called the skew parameter. It is non-zero in cameras where the image axes are not perpendicular, usually not the case for today's cameras.

Extrinsic parameters

Via translation and rotation, the camera frame and the world reference frame are related. With two coordinate frames, $\mathcal{F}_a$ and $\mathcal{F}_b$ with different positions and orientations, the transformation between the two frames can be described by the homogeneous transformation matrix $\mathbf{T}_{ab}$.

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{t}^a_{ab} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \text{where} \quad \mathbf{x}^a = \mathbf{T}_{ab}\mathbf{x}^b,$$

The matrix $\mathbf{R}_{ab}$ is the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, and $\mathbf{t}^a_{ab}$ the translation vector given in the coordinate frame of $\mathcal{F}_a$.

When describing the intrinsic parameters, the world coordinates were assumed to be given in the camera frame. To describe the extrinsic parameters, the scene points are expressed in the world coordinate frame, the one to the far right in Figure 2.2. A point in the world frame can be expressed in the camera frame by

$$\mathbf{x}_c = \mathcal{R}\mathbf{x}_w + \mathbf{t}, \tag{2.5}$$

where $\mathcal{R}$ and $\mathbf{t}$ is thus the extrinsic parameters of the camera.

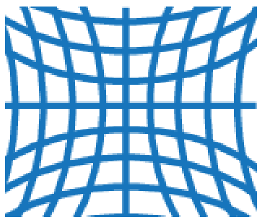
Combining the intrinsic parameters K with the extrinsic $[\mathcal{R}, \mathbf{t}]^T$ gives the camera matrix. It relates world coordinates with pixel coordinates, and can be written as follows.

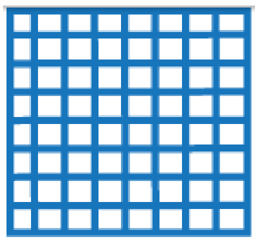
$$\mathbf{P} = \begin{bmatrix} \mathcal{R} \\ \mathbf{t} \end{bmatrix} K \tag{2.6}$$

Distortion

Distortion makes straight lines in a scene appear bent in the image. The optical distortion is derived in the lens and occurs when lens elements are used to reduce aberrations. An ideal pinhole camera does not have a lens, and thus the camera matrix does not account for lens distortion.

There are different kinds of distortion; tangential distortion and radial distortion. The latter is the most important to take into account with nowadays lenses. It occurs when light rays are bent more near the edges of the lens than in the center. Smaller lenses have greater distortion. Figure 2.3 shows how different types of radial distortion looks like in


(a) Negative radial distortion


(b) No distortion

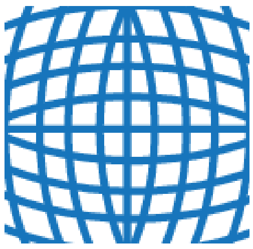

(c) Positive radial distortion

Figure 2.3: Radial distortion

an image. Let (x, y) be the ideal points, the radial distortion (x_d, y_d) can be modeled as follows

$$x_d = x + x\left[k_1(x^2+y^2) + k_2(x^2+y^2)^2 + k_3(x^2+y^2)^3 + \ldots\right]$$
$$y_d = y + y\left[k_1(x^2+y^2) + k_2(x^2+y^2)^2 + k_3(x^2+y^2)^3 + \cdots\right]$$

Where k_1, k_2, k_3, ... are the radial distortion coefficients of the lens. [49]

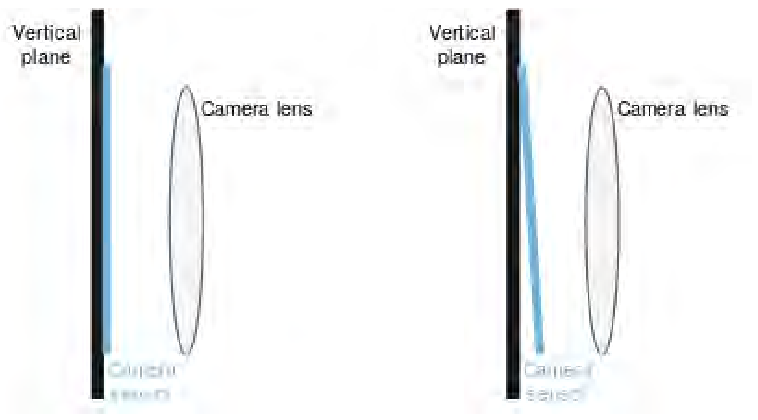


Figure 2.4: Left: No tangential distortion. Right: Tangential distortion

Tangential distortion occurs when the lens and the image plane are not parallel, as shown in Figure 2.4. When modeling tangential distortion (x_d, y_d), the undistorted pixel locations (x, y) are normalized and thus dimensionless. Expressing the model with the tangential distortion coefficients of the lens, p_1 and p_2;

$$\begin{aligned} x_d &= x + \left[2p_1xy + p_2(x^2+y^2+2x^2)\right] \\ y_d &= y + \left[p_1(x^2+y^2+2y^2) + 2p_2xy\right] \end{aligned} \tag{2.7}$$

2.1.2 Calibration

Camera calibration is the process where intrinsic and extrinsic parameters are estimated. The intrinsic and extrinsic parameters are what relate the world points to the camera coordinate system. The parameters can be used to correct for lens distortion, get the size of an object in the real world, or determine where in the scene the camera is located. There exist different techniques, where the two main ones are:

- **Photogrammetric calibration**: The calibration is performed by using an object with known 3D world points. The correspondences between the real world points and the corresponding 2D image points can be found by having multiple images of a known calibration pattern. The most commonly used pattern is the checkerboard, where the number of squares and their size is known.
- **Self-calibration**: No calibration object used, and therefore the camera is moved in a static scene. The correspondence between three images is used to estimate the intrinsic and extrinsic parameters.

Both methods have advantages.

Zhangs method

Zhangs method is a calibration technique developed by Zhengyou Zhang at Microsoft Research [49]. This method lies between the two main techniques described above, exploiting the advantages of both methods. The only requirement is that the camera has to observe a planar pattern, typically a checkerboard, in at least two different orientations. The algorithm then detects feature points in the images. Since the plane $z = 0$ is given by the pattern itself, only 2D metric information is used. The linear transformation between planes is computed, and by using singular value decomposition, an initial estimation of the intrinsic parameters are computed. Furthermore, the parameters are refined by applying the Levenberg-Marquardt algorithm to the algebraic error. The same approach is used to make an estimate of the extrinsic parameters. When estimating the lens distortion, the tangential distortion is ignored. An estimate of the radial distortion parameters is calculated by comparing the real pixel values and the ideal ones given by the pinhole model.

2.2 Stereo Camera

Stereo vision is a computer vision technique used to perceive depth in a scene by combining two or more sensors. With the second camera of known relative position, the 3D position of points projected on the camera plane can be estimated through triangulation. Working with binocular cameras requires an understanding of the calculations and setup, which influence the 3D points probability distribution. As well, the application area and the choice of algorithms used for computing and solving the correspondence problem will affect the resulting depth-map and the associated point cloud.

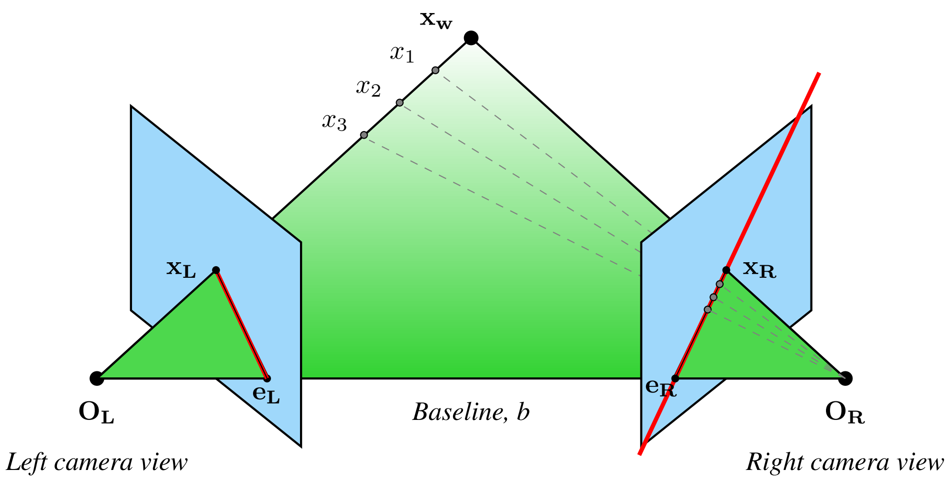


Figure 2.5: Epipolar geometry of two cameras

2.2.1 Stereo geometry

Observing the same world point $\mathbf{x_w}$ from two relative views gives a strong geometric constraint on the observed point called the epipolar constraint. By using the geometric pinhole model, the depth is calculated by triangulation. Figure 2.5 gives an illustration of the epipolar geometry between two cameras. The image planes are illustrated in blue, and the point $\mathbf{x_w}$ is the world coordinate desired to reconstruct depth.

The green triangular in Figure 2.5 connects the two optical centers and the point $\mathbf{x_w}$; it is called the epipolar plane. The baseline, fixed-length depending on the stereo setup, is the line connecting the two optical centers, $\mathbf{O_L}$ and $\mathbf{O_R}$. The epipolar plane intersects through the two blue image planes forming the red epipolar line in each plane. The epipolar line in the left camera view intersects the epipole $\mathbf{e_L}$ and the pixel value $\mathbf{x_L} = (u_L, v_L)$.

The epipolar constraint uses the epipolar line to find pixel correspondences. It can be stated:

"A point in one image generates a line in the other on which its corresponding point must lie. The search for correspondences is thus reduced from a region to a line"

—Bob Fisher, University of Edinburgh

The constraint arises by the reason that when two cameras capture the same world object, the pixel points, the world point, and the optical centers are all coplanar. The depth is thus directly proportional to the distance between the two pixels capturing the world object and the baseline.

The algebraic mapping between a pixel in one image and the epipolar line in the second image is described by a matrix called the fundamental matrix. See Chapter 8 in [13] for a further mathematical explanation. The theory is illustrated in Figure 2.5, where the pixel $\mathbf{x_L}$ in the left view is mapped to all the pixels on the epipolar line of the right camera. In practice, every pixel on the epipolar line in the right camera view capturing the world points $\mathbf{x_w}$, x_1, x_2, or x_3 can correspond to the one pixel $\mathbf{x_L}$ in the left view. The problem of selecting the correct matching pixel in the right view is termed the correspondence problem and is further explained in Section 2.2.3.

2.2.2 Stereo setup

The field of view, blind-spot, and uncertainty all depends on the stereo mounting. The choice of baseline and camera vergence angles set constraints on the scene used.

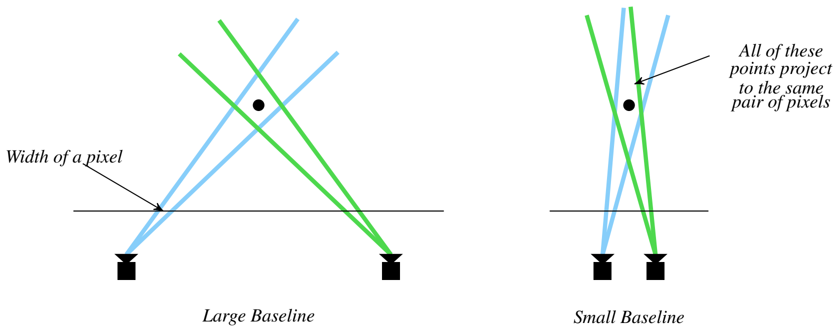


Figure 2.6: Illustrating uncertainty depending on the baseline

Representing the 3D world with a discrete camera plane consisting of pixels gives rise to uncertainty in the depth estimate. In Figure 2.6 the uncertainty with different baselines is demonstrated. In general, the uncertainty will decrease with expanding baseline. In addition to the baseline, the vergence angle of the cameras will affect the shape of the uncertainty region. Changing the angle between the cameras will result in a similar effect, as illustrated in Figure 2.6, akin parallel rays give less precisely localization of 3D points. The accuracy of reconstructed 3D points is thus dependent on both the baseline and the angle, where the width of the probability distribution will decrease with increasing baseline and vergence angle.

Even though the accuracy may increase with increasing baseline and vergence angle, the search problem becomes more difficult, and the field of view shrinks. The field of view of the system is directly linked to the distance between the two cameras, baseline, and the angle the cameras are adjusted towards each other [25]. Depth estimation is possible only for the overlapping field of view between the two camera views, and in practice, minimum $\frac{2}{3}$ of each image should overlap. Three different setups are demonstrated in Figure 2.7, where the overlapping field of view is illustrated in green. The overlapping field of view will diminish with a more substantial baseline. Additionally, the blind spot will expand. A solution for lessening the blind spot is demonstrated in the rightmost figure titled *Angled*. When the cameras are angled towards each other, the field of view on short distance is increased and withal decreasing the blind spot.

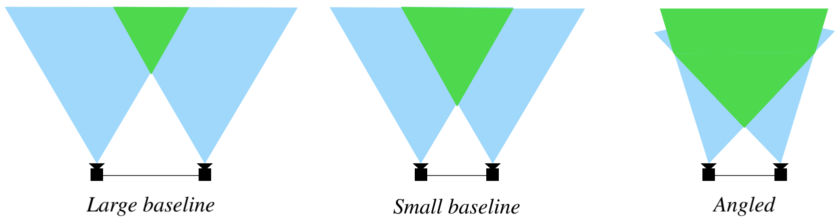


Figure 2.7: Illustrating field of view in stereo setup

Another benefit of angled cameras is that the vergence angle determines a particular fixation point in the scene. When the fixation point is near a target object, it can help separate the target objects from distracting surroundings. The working distance, i.e., how far the setup is from a given object or fixation point, can be calculated depending on the camera optics.

$$\mathcal{WD} = \frac{f * FOV_h}{h}, \qquad \begin{aligned} FOV_h &= \textit{horizontal field of view} \\ h &= \textit{horizontal sensor size} \\ \mathcal{WD} &= \textit{working distance} \\ f &= \textit{focal length} \end{aligned}$$

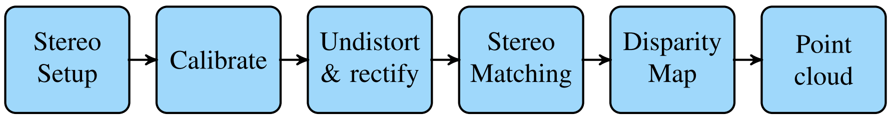
The baseline is directly proportional to the horizontal distance from the mounting to the object. Thus stereo systems with smaller baseline achieve better results when detecting objects on average distances, whereas broader baseline is preferred with greater distances. As a general rule, the fraction $\frac{1}{30}$ can be used to estimate the dimension of the baseline [5]. For a working distance on 30 meters, the baseline should be around one meter, depending on the cameras' angling.

Eventually, the setup is a matter of the distance one wants to detect, the width and blind spots, which are all correlated. The factors all define what level of depth-accuracy the system will achieve. Further, the camera specifications will affect the result, e.g., focal length, and if the objects to be detected are small, consider cameras with higher resolution.

2.2.3 Correspondence problem

Depth is inversely proportional to the disparity between pixels in stereo captured images, implying that calculating depth is a single trivial equation. However, in order to do the calculation, one must first determine for each pixel in one image which pixel in the second originated from the same 3D world point. This issue is known as the correspondence problem and is a central part of binocular vision.

Before stereo matching and solving the correspondence problem, the images should be undistorted and rectified to ease the computation of finding corresponding pixels. The pipeline for retrieving depth from stereo vision can be illustrated in a flowchart;



When the rigid body transformation between the two cameras is known from the calibration, it can be used to undistort and rectify the images. Rectification is the process of transforming the two images onto a common image plane, aligning the epipolar lines. It ensures that the corresponding pixels in the image pair are on the same row. The concept

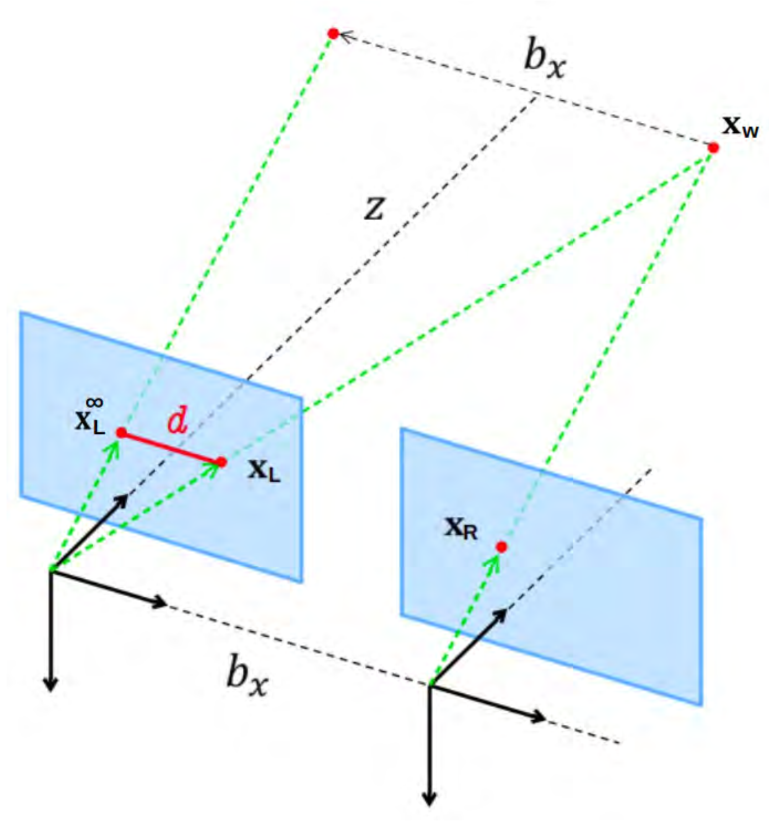


Figure 2.8: Ideal Stereo Geometry

is shown in Figure 2.8. By defining new camera matrices (2.6), the rotation matrices is set to the identity matrix and the translation only along the x-axis, the correspondence problem is mapped to ideal stereo geometry. The ideal stereo geometry is a special case of the epipolar geometry where the epipoles are at infinity, meaning $\mathbf{x_L}^\infty = \mathbf{x_R}$. The cameras will thus have identical orientation and baseline along the x-axis with distance b_x. Which gives the following pixel correspondence;

$$\mathbf{x_L} = \mathbf{x_R} + \begin{bmatrix} \frac{f * b_x}{d} \\ 0 \\ 0 \end{bmatrix}$$

This means that the epipolar lines become horizontal. Looking for corresponding pixels in the two images boils down to a one-dimensional search.

The matching of corresponding pixels can be performed by various stereo matching algorithms. A selection of the various methods is explained in Chapter 3. They are all local methods, matching conjugate points directly based on intensity or indirectly by feature extraction and matching.

When the corresponding pixels are matched and the disparities achieved, they can be displayed in a disparity map. Disparity map, also commonly named range- or depth-map, is a representation of the displacement between conjugate pixels in the stereo pair image. It represents the motion between the two views, where small disparities are shown as

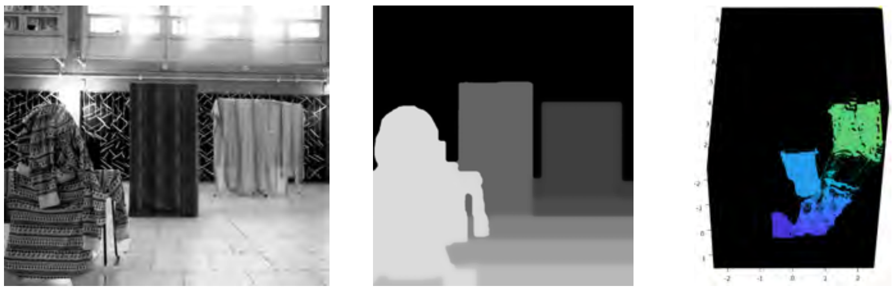


Figure 2.9: Example of scene with corresponding grey-scale disparity map and Point cloud

darker pixels, therefore less motion and longer distance. An example of a disparity map is illustrated in Figure 2.9.

Once the disparity map is extracted, the 3D world position of a given object only depends on straightforward mathematics. The triangulation between pixels in the left- (u_L, v_L) and right-image (u_R, v_R) is given in the consecutive equations. The position is given in the cameras coordinate frame, where b_x is the baseline of the stereo camera, f, the focal length, and d corresponds to the disparity between the two images.

$$Z = \frac{f * b_x}{d}$$

$$X = \frac{u_L - u_R}{f} Z \tag{2.8}$$

$$Y = \frac{v_L - v_R}{f} Z$$

The 3D world coordinates calculated can further be plotted together, creating a point cloud [39]. Point clouds are a means of collating a large number of single spatial measurements into a dataset that can then represent a whole. One can say that the point cloud displays the back-projection of the stereo image pair; an example is displayed in Figure 2.9.

Pre- and post-processing

In general, the less noisy the disparity map, the better results of triangulation can be achieved [26]. Stereo matching algorithms tend to create noise in the disparity map under challenging scenes. Depending on the algorithm, a challenging area can be regions with depth discontinuities, texture-less areas, and half-occlusions. There is possible to filter the images before processing a disparity map or filter the disparity map afterward to remove noise. Traditionally, pre-processing involves contrast enhancement, noise removal, and deblurring to improve the image quality, which helps in future processing [17]. This may include filters like median filtering, Wiener filter, and Histogram Equalization. After creating the disparity map, the map can be further post-processed to achieve a less noisy result. This may be smoothing-based or segmentation based approaches. Examples can be Hough Transform, the Weighted Least Squares filter, a left-right consistency check, or a cross-check-test. [26]

Chapter 3

Stereo matching

Stereo matching is the task of establishing correspondences between two images of the same scene. The distance between matching points is used to create a disparity map. Disparity maps are used in various applications with a broad range of input data, and the depth estimated from the disparity can vary remarkably depending upon the properties of the data [33]. Thus a broad range of algorithms has been developed for acquisition corresponding points and subsequent for matching the points to estimate depth. In general, the methods can be classified into local and global approaches. The local methods consider pixels in a local neighborhood, while the global methods use the information presented in the entire image. Global algorithms are primarily based on the optimization of a well-behaved-energy function created form the hole image [22]. Because the energy function is defined on all the pixels of the image, global methods are less sensitive to local ambiguities than local methods [22]. In short, the global methods may achieve more accurate disparity maps, while the local methods yield less computational cost. As the goal of this thesis is to develop a system that can be implemented in real-time, the focus will be on the local methods to achieve lower computational complexity.

There are several steps before outputting a disparity map. The steps may differ, but usually, disparity map methods are implemented using a multistage technique, as shown in Figure 3.1. The input images are obtained from stereo vision cameras. The image

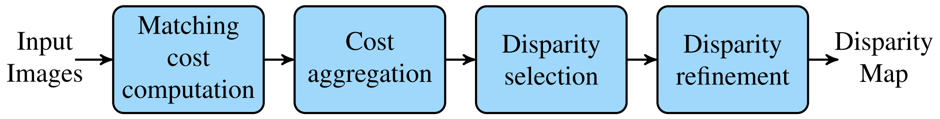


Figure 3.1: Framework for Disparity map

pair will go through all the four steps sequentially before the output should desire to be a noiseless disparity map. The disparity map can be either dense or sparse. Dense methods attempt to reconstruct and use a 3D map for all the pixels in the images, while sparse methods reconstruct and use only a selected set of independent points in the scene [10].

The two last steps in the flowchart, disparity selection, and disparity refinement are not the focus area for this thesis, and thus only briefly discussed.

Local methods broadly prioritize the matching cost computation and cost aggregation. The matching cost is usually optimized by selecting for each pixel the disparity with the smallest value, a winner-take-all strategy, instead of the energy function used in global methods. The methods are known to provide reasonable results with less computational cost than global methods. Local methods are statistical methods, which can be subdivided into two broad categories, namely correlation-based methods and feature-based methods. Different algorithms in both of the two categories are explained in the following two sections.

3.1 Correlation based methods

Correlation based methods are one of the earliest and simplest approaches in stereo matching. While new matching techniques have been proposed in recent years, correlation-based on rectified images still has an advantage due to computational cost and fewer memory requirements. The emphasis is on exploiting the neighborhood of a pixel to find a suitable match on the same horizontal axis in the second image. The neighborhood is a predefined block, or a matrix, therefore often termed block matching or area-based methods.

The neighborhood in the template image is matched with the neighborhood around pixels on the same horizontal axis of the second image. Commonly the matching cost aggregation is achieved by summing or averaging the matching cost in the surrounding window centered by the current pixel. In Figure 3.2, the small reference matrix in the left

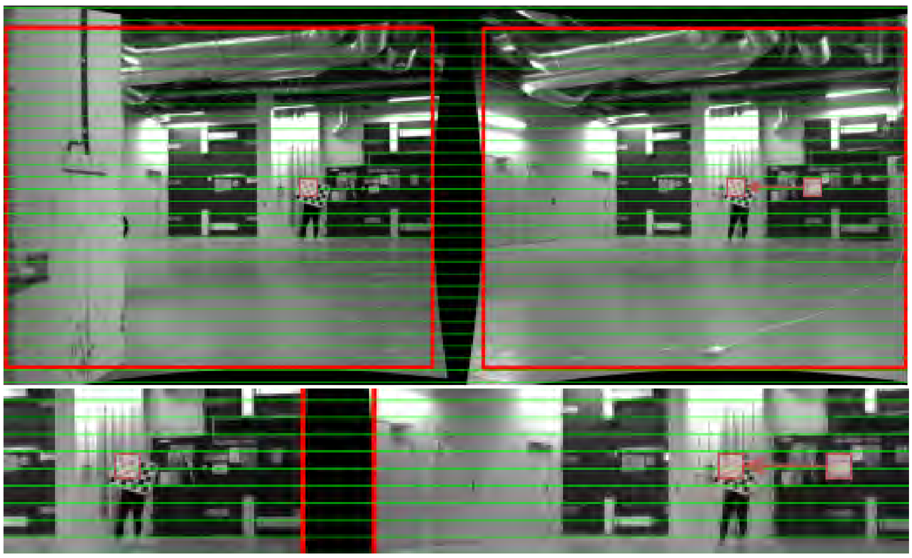


Figure 3.2: Block matching between two rectified images. The reference block in the left image is matched with different blocks on the same horizontal line in the right image.

template image is tried matched to a matrix on the same horizontal line in the right image. When searching the right image, the first block which is tried matched is located at the same coordinates as the reference matrix; this is the rightmost matrix in the right image. By moving the block to the left, the correct matching block is detected. The correct block is defined as the one with the most similarity. The main difference between the correlation based methods is the technique used to measure the similarity of image locations. The mathematical equations are stated with the following nomenclature.

$$\begin{aligned} I_L &- \text{Left image from the stereo setup} \\ I_R &- \text{Right image from the stereo setup} \\ W_m &- \text{Matching window} \\ d &- \text{Disparity measured in pixels} \\ I(u,v) &- \text{Image pixel intensity at location u,v} \end{aligned}$$

3.1.1 Sum of Absolute Differences

Starting with Sum of Absolute Difference (SAD), which is the most common matching criteria in stereo matching algorithms [44]. It is known for its simplicity and low execution time. The SAD equation,

$$C_{SAD}(d) = \sum_{(u,v)\in W_m} |I_L(u,v) - I_R(u-d,v)|$$

calculates the absolute differences between two selected windows or matrices. The value is used as an estimate of how similar blocks are; small values indicate similar blocks. Thaer and Hussein [44] tested the algorithm with Ground Truth to calculate the percentage error. With real and rectified stereo image pairs, the algorithm moves the reference window pixel by pixel over the target windows horizontally aligned with the baseline and give varying results depending on the window size. As stated in [44], error percentage and execution time correlates. In general, smaller window sizes, like 3x3 matrices, gives a high error percentage and short execution time. Moreover, a window size of, e.g., 27x27pixels takes 44 times as long to compute [44]. The choice of window size is thereby a compromise between the accuracy of the result and the execution time. Choosing the right window size depends on the scene, the computational constraints, and the size of the objects to be detected. In Chapter 4, the result with various window sizes is displayed.

3.1.2 Sum of Squared Differences

Sum of Squared Differences, abbreviated SSD, is one of the most naive block-matching algorithms. The algorithm is directly using the equation of sum of squared error, stated in digital form;

$$C_{SSD}(d) = \sum_{(u,v)\in W_m} [I_L(u,v) - I_R(u-d,v)]^2$$

The formula matches the intensity difference, pixel by pixel, between the two images. Calculating SSD over a range of displacements followed by identifying the minimum SSD value will, in theory, correspond to the best alignment offset. [27]

SSD subtracts pixel values in the two images and is used due to its simplicity and relatively low computational cost. In general, SSD minimization is robust against slight deformations but fails in the presence of stronger ones. Considering the algorithm only measures the deviation of data points away from the mean value, it failure occurs when the window contains two depth estimates for the same are, i.e. a depth discontinuity. In the indicated scenario, it is impossible to find a unique correspondence leading to the occlusions and disocclusions artifacts [3]. Therefore, SSD may struggle to give satisfying results when the stereo setup captures multiple objects or objects on long distances.

3.1.3 Normalized Cross Correlation

Commonly Normalized Cross Correlation is known to yield better results than SSD [27]. NCC is one of the most common algorithms to compare the similarity between two matrices. It can be the right choice when using stereo vision because it can compare metrics with different value ranges. In other words, the algorithm can achieve acceptable results if the left and the right camera have variations in brightness.

Assuming the stereo pair is aligned, the following equation is used for calculation:

$$C_{NCC}(d) = \sum_{(u,v)\in W_m} \hat{I}_L(u,v)\hat{I}_R(u-d,v) \quad , \qquad \hat{I}(x,y) = \frac{I(x,y)-\bar{I}}{\|I-\bar{I}\|_{W_m}}$$

Where $\bar{I}$ (calculated only once) is the average intensity value of the pixels in the window W_m. Summarizing, NCC yields good results due to its precision and the high robustness to different illumination conditions between two cameras. The technique presents two significant drawbacks. The first is that the calculation of depth is generally prone to errors close to the surface discontinuities, and due to the underlying mechanism for correlation is based on a series of multiplication operations, it is notorious for high complexity and computational cost. Thereby it rarely meets speed requirements in real-time stereo vision systems [42].

3.1.4 Rank Transform

In contrast to the algorithms presented, the Rank Transform is a non-parametric transform. When non-parametric, the method relies on the relative ordering of local intensity values and not on the intensity values themselves. Due to this, the algorithm can tolerate a significant number of outliers and is thereby more insensitive to image noise and brightness difference of stereo images.

In brief, RT measures the local intensity and is calculated based on the absolute difference between the rank of the reference matrix and the rank from the target matrix. RT analyze the set of pixels in some square neighborhood surrounding the pixel (u, v). The algorithm compares the intensities of a pixel, $I(u, v)$, versus the pixels in the neighborhood.

Equation (3.1) present the algorithm, while (3.2) elaborates.

$$C_{RT}(d) = \sum_{(u,v)\in W_m} |Rank_L(u,v) - Rank_R(u-d,v)| \tag{3.1}$$

where

$$Rank(x,y) = \sum_{(i,j)(x,y)} L(i,j) \quad , \qquad L(i,j) = \begin{cases} 0: & I(i,j) < I(x,y) \\ 1: & \text{otherwise} \end{cases} \tag{3.2}$$

When comparing pixels with its neighborhood, RT only depends on the sign, as shown in (3.2), where the pixel intensity $I(i,j)$ is the neighboring pixel to the matching pixel (x,y).

Compared with conventional area-based methods, RT is shown to be suitable for real-time implementation due to less computational complexity [48]. Looking at conventional normalized correlation methods, such as NCC, RT can result in improved performance near object boundaries. Since the method is efficient in coping with brightness differences, it is reasonable to consider the algorithm for outdoor use.

3.1.5 Census Transform

Equal to Rank Transform, the Census Transform is a non-parametric measure. It summarizes local spatial image structure and is well known for its illumination invariant properties. For every reference pixel, CT computes a binary string, the census signature, by comparing the grey values in its neighborhood. As shown in (3.3) the signature encodes whether the neighbours (i,j) are greater or equal to the reference pixel (x,y). For a 3x3 neighborhood window, the census signature has a length of eight represented efficiently in a computer by a single byte [11]. Subsequently, the Hamming distance is used in (3.4) to compare the similarity between two Census Transformed Images, i.e., between the two matching windows. The Hamming distance is calculated by and XOR-operator, counting the number of 1's in the resulting string.

$$Census(x,y) = Bitstring_{(i,j)\in W_m}(I(i,j) \geq I(x,y)) \tag{3.3}$$

$$C_{CT}(d) = \sum_{(u,v)\in W_m} Hamming(Census_L(u,v) - Census_R(u-d,v)) \tag{3.4}$$

In computer vision, the Census Transform is frequently used in local stereo algorithms due to its decent performance under challenging lighting conditions [36]. It is illumination-robust and invariant under global monotonically increasing grey level rescalings- both crucial advantages in modern stereo vision applications like autonomous vehicles.

3.2 Feature based methods

Compared to the correlation based methods using image windows, the feature-based methods use a sparse set of features. Image features include edges, corners, vertices of linear structures, circles, prominent surface markings, and patches. Summarized, they can be determined as an object, structure, or trends occupying an image. Feature detection algorithms involve searching for features in each image which are suitable to match with features in other images. The region around each feature is further described in a compact and stable manner in the descriptor to make matching easier. The features are thus matched with features in the second image by comparing the feature descriptors. Corresponding feature pairs extracted from the stereo images are the ones associated with the minimum distance often computed by a brute force- or nearest neighbors-technique.

Feature based matching is increasingly used, and technological developments is advancing from correlation block matching methods to the feature based matching techniques [15]. Local features are robust to occlusion and clutter, can differentiate an extensive database of objects, and exploit different types of features in different situations. Nevertheless real-time performance is achievable. In disadvantage,s the methods cannot detect small changes in the stereo images and are therefore not suitable for images with no boundary.

Feature matching is at the base of many computer vision problems, such as object recognition or structure from motion, but still not widely used in stereo matching problems. Traditionally it makes little sense to use feature matching procedures in stereo matching. This is by reason of rectification, by rectifying the image pairs finding correspondences is less prone to errors and more effortless with the correlation-based methods. A key part of feature detection is to find what features one wants to detect, i.e., finding the reoccurring trends of a specific desired feature. In the sequel, an introduction to some of the most popular feature based methods.

3.2.1 Harris Corner Detector

Harris Corner Detector or Harris Operator is one of the earliest detectors proposed in 1988 by Harris and Stephens [9]. It is an intensity-based mathematical operator finding features using corner descriptors in the extraction of features. A corner is the intersection of two edges, a point where the directions of the two edges change. Hence, a corner can be detected by high variation in the intensity gradient (in both directions). To detect corners and edges, the Harris corner detection uses a combination of partial derivatives, Gaussian weighting function, and eigenvalues from the matrix representation of the following equation:

$$E(\Delta u, \Delta v) = \sum_{(u,v)} W(u,v)[I_L(u+\Delta u, v+\Delta v) - I_R(u,v)]^2 \tag{3.5}$$

The equation (3.5) is indirectly searching for corners. By sweeping a window W at position (u,v) the variation of intensity is calculated by computing the intensity between the window $W(u,v)$ and the displacement window $W(u+\Delta u, v+\Delta v)$. The window function $W(u,v)$ is either a rectangular window or Gaussian window, giving weights to pixels underneath. The rectangle window function gives weight 1 to pixels inside the window and 0 otherwise. In practice, one can get a noisy response with a binary window function.

The objective is to determine the positions which maximizes (3.5). By representing the Taylor expansion as a matrix, subsequently calculating the eigenvalues, weak and robust corners are detected with a threshold value. Deriving the mathematical expressions step by step:

$$\begin{aligned}
E(\Delta u, \Delta v) &\approx \sum_{(u,v)} w(u,v)[I_L(u,v) + \Delta u I_{Lu} + \Delta v I_{Lv} - I_R(u,v)]^2 \\
&= \sum_{(u,v)} w(u,v) \left[\Delta u^2 I_{Lu}^2 + 2\Delta u \Delta v I_{Lu} I_{Lv} + \Delta v^2 I_{Lv}\right] \\
&= \begin{bmatrix}\Delta u & \Delta v\end{bmatrix} \left(\sum_{(u,v)} w(u,v) \begin{bmatrix} I_{Lu}^2 & I_{Lu} I_{Lv} \\ I_{Lu} I_{Lv} & v^2 I_{Lv} \end{bmatrix} \right) \begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix} \\
&= \begin{bmatrix}\Delta u & \Delta v\end{bmatrix} M \begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix}
\end{aligned}$$

For each window a score R is calculated for determining if a window is containing a corner or not;

$$R = \det(M) - k(\text{trace}(M))^2 \tag{3.6}$$

where

$$\begin{aligned}
\det(M) &= \lambda_1 \lambda_2 \\
\text{trace}(M) &= \lambda_1 + \lambda_2
\end{aligned}$$

Based on (3.6) the algorithm decide whether a region is a corner, edge or flat. When $|R|$ is small, λ_1 and λ_2 small, the region is flat. If $R < 0$ ($\lambda_1 >> \lambda_2$ or $\lambda_1 << \lambda_2$) the region is a edge. And last if R is large, i.e. λ_1 and λ_2 are large and approximately equal, the region is declared a corner. The threshold of R is determined in advance.

The Harris Corner Detector is simple to compute, relatively fast, and popular because it is rotation and illumination invariant. However, it is non-invariant to image scale. Detecting corners on a specific captured boat will present different results depending on its distance from the cameras(size in the picture). A detected corner in a high-resolution picture may not be a detected corner if the image is scaled down.

3.2.2 SIFT

SIFT was proposed in 2004, is widely used and the basis in many subsequently developed features [23]. The keypoint detector and descriptor have proven remarkably successful in several applications using visual features. The method can be comprehended in four stages briefly explained in order:

Scale-space extrema detection
In contradiction to the Harris Corner Detector, SIFT is scale-invariant. This is accomplished by detecting keypoints under various image sizes. By successively reducing the image size by combining smoothing and subsampling, the well-known pyramid representation is obtained. Briefly explained SIFT searches over all scales and image locations

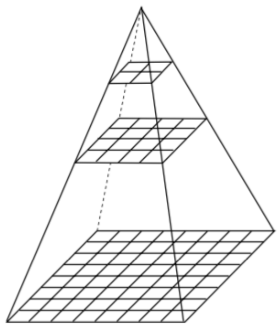


Figure 3.3: A pyramid representation is obtained by successively reducing the image size by combined smoothing and subsampling

to find the local maxima across scale and space. Due to speed constraints, the algorithm is implemented using Difference of Gaussian, which is an approximation of Laplacian of Gaussian. The DoG identifies potential interest points, invariant to scale, and orientation. Outputting a list of (u, v, σ) values, which is the pixel position at the scale σ.

Keypoint localization
At each potential interest-point detected, a Taylor series expansion of the scale space is fit to get a more accurate location and scale. Additionally, a concept similar to the Harris Corner Detector is applied to discard edges. In short, SIFT selects keypoints based on a measure of their stability; the remaining keypoints are the strongest interest points.

Orientation assignment
Based on local image gradient directions, orientations are assigned to each keypoint location. The gradient magnitude and direction of a keypoint is calculated by taking the neighboring pixels. This is accomplished to achieve invariance to image rotation. All the keypoints are at the presence transformed relative to the assigned orientation, scale, and location, providing invariance to these transformations.

Keypoint descriptor
The final stage generates the descriptor by measuring the local image gradients at the selected scale. A total of 16x16 pixels around the keypoint is taken and divided into 16 sub-blocks. At each sub-block, eight orientations are created, represented as a histogram (vector) to form keypoint descriptors. Finally, the eight vectors are assembled to a 128 dimension vector, normalized, and the values thresholded. Thus transforming the vector into

a descriptor allowing significant levels of local shape distortion and change in illumination.

The method is computationally expensive and hard to comprise in a section for a comprehensive explanation look into the paper written by the developer Lowe [23]. It is known to be highly expressive and thus suitable for applications with tracking and recognition. However, it imposes a significant computational burden for real-time systems. It is worth noticing that it is patented, and thereby, permission is needed to use SIFT in commercial algorithms (free for academic and research purposes) [30].

3.2.3 SURF

SURF was introduced as a speeded-up version of SIFT in 2006 [2]. It is a rotation- and scale-invariant descriptor, with the main interest in its fast computation of operators. Two get an explicit explanation of the algorithm, it is recommended to look into the paper written by the creators [2]. The method is separated into two steps; feature extraction and feature description.

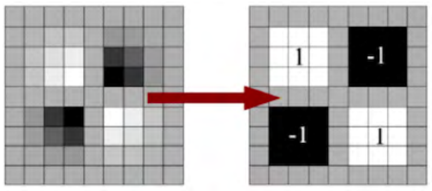
Feature Extraction
SURF approximates Laplacian of Gaussian with Box Filter to obtain scale-space; in comparison, SIFT approximates using DoG. The Box Filter is computationally agile when using the Integral Image. For selecting location and scale, the method makes use of the determinant of the Hessian matrix.

$$H(f(u,v)) = \begin{bmatrix} \frac{\partial^2 f}{\partial u^2} & \frac{\partial^2 f}{\partial u \partial v} \\ \frac{\partial^2 f}{\partial u \partial v} & \frac{\partial^2 f}{\partial v^2} \end{bmatrix}$$

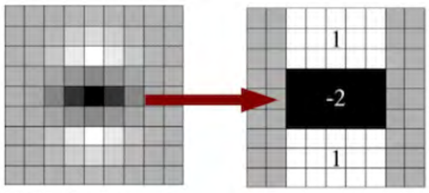
With $\mathbf{x} = (u,v)$, the Hessian matrix is defined:

$$H(\mathbf{x},\sigma) = \begin{bmatrix} L_{uu}(\mathbf{x},\sigma) & L_{uv}(\mathbf{x},\sigma) \\ L_{uv}(\mathbf{x},\sigma) & L_{vv}(\mathbf{x},\sigma) \end{bmatrix}$$

By approximating the second-order derivative with integral images, SURF reduce the computation time compared to SIFT. In Figure 3.4 a box filter with size 9x9 is used to



(a) L_{xy}



(b) L_{yy}

Figure 3.4: noe

approximate the second order-derivative in the given scale σ. This gives the determinant:

$$\det(H_{approx}) = D_{uu}D_{vv} - (wD_{uv})^2$$

To make it scale-space invariant, SURF uses the same technique as SIFT.

Feature Description
To make the descriptor invariant to rotation, SURF uses Wavelet responses in horizontal- and vertical direction forming it into a vector with a dimension of 64 bits. It is worth noticing that a SURF-descriptor takes half the memory-space compared to a SIFT-descriptor, and this is part of the reason why SURF yields faster computation time.

SURF is relatively fast and can be implemented in real-time applications, but as SIFT, the method is patented.

3.2.4 ORB

Oriented FAST and Rotation Aware BRIEF (ORB) was developed at OpenCV labs in 2011 as an efficient and viable alternative to SIFT and SURF. Both the latter are patented; thus ORB was mainly conceived as an alternative free algorithm. The developers' study is explained in their article [38]. The ORB feature builds upon the FAST keypoint descriptor [37]. Selecting the most robust features using FAST or Harris response, and finds their orientation using first-order moments. The descriptors are computed using BRIEF [4]. The algorithm is a fusion between keypoint detectors and descriptors, and the mathematical overview can be comprised in steps: The first step is a simple measure of a corner orientation, namely the intensity centroid. The moments of a patch are defined as,

$$m_{pq} = \sum_{(u,v)} u^p v^q I(u,v),$$

, with the center of mass for the patch, i.e., the centroid C.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

A vector is constructed between the centroid and the corner's center. SImple trigonometry gives us the orientation of the patch θ.

$$\theta = \text{arctan2}(m_{01}, m_{10})$$

The orientation is obtained to rotate the patch to a canonical rotation, making it rotation invariant. The second step include an improved BRIEF algorithm taking all the keypoints detected from the FAST keypoint descriptor and converts them into a binary feature vector representing objects. By using a Gaussian kernel, the binary descriptors are made insensitive to high-frequency noise.

ORB is known for being resistant to noise and is widely used in robotics because it is proven to be an efficient alternative to SIFT. It is also, in selected articles, concluded to be more efficient than SURF, but this seems to be dependent on the application and use [35]. However, Patel et al. [31] demonstrate that in low light conditions, the number of detected keypoints decreases drastically.

3.2.5 Descriptor Matching

After extracting local feature descriptors in both the images by one of the methods described in Section 3.2 one has to match the features to find corresponding keypoints for creating a disparity map. There exist various distance measures to compare the keypoints, and the choice depends on the used feature extractor. The matching methods later used are briefly explained throughout this section.

Brute-Force Matcher

Brute-Force matcher (BF) is known to be simple, and goes through all the descriptors in one image and tries to match it with all the descriptors in the second image [28]. The closest descriptor returned is the one based on some distance calculation. The distance measurement used for SIFT and SURF is usually Euclidean distance (L2-norm), due to both the methods represent their descriptors as vectors (histograms) of an oriented gradient. For binary descriptors like the ones produced by ORB, the Hamming distance should be used.

FLANN Matcher

Fast Library for Approximate Nearest Neighbors Matcher performs an efficient matching by using clustering and search in multi-dimensional spaces [28]. It was created by Muja and Lowe [29] in 2009 because there existed no efficient nearest neighbor matching in high-dimensional spaces. FLANN includes a collection of algorithms optimized for computing the nearest neighbors in large datasets. However, the creators' article [29] makes a comparison of the various algorithms and found that the overall best performance was achieved with multiple randomized k-d trees.

The classical kd-tree algorithm was proposed in 1977 by Freidman et al., [8]. Because it did not perform in higher dimensions, an optimized version is proposed and implemented in the library of OpenCV [29].

FLANN uses the same distance measurements as the BF-matcher. Usually, a FLANN based method is preferred with images with high resolution. While the Brute-Force-matcher tries every possibility and is guaranteed to find the best neighbor, the FLANN matcher finds an approximation of the nearest neighbors. FLANN finds suitable matches, but it may not be the best possible one.

3.3 Comparison

In literature, there is a limited amount of papers directly comparing correlation-based and feature-based methods. Therefore, they are comprised in two tables, where every algorithm is ranked in increasing order with the number 1 as the best. The comparisons are based on the literature read throughout this chapter.

Method	Computational Cost	Accuracy
SAD	1	4
SSD	2	5
NCC	5	3
RT	3	2
CT	4	1

Table 3.1: Correlation based methods

In Table 3.1, the correlation-based methods are compared, equivalent the feature based in Table 3.2. By examining the tables, one observes that there is a compromise between the

Methods for extracting and creating descriptors		
Method	Computational Cost	Accuracy
HCD	1	4
SIFT	4	1
SURF	3	3
ORB	2	2

Method for matching descriptors		
Method	Computational Cost	Accuracy
BF	2	1
FLANN	1	2

Table 3.2: Feature based methods

accuracy and the computational cost. Certainly, one strives for the most accurate algorithm with the less computational cost. Based on this, the best correlation-based methods, in theory, will be SAD, RT, and CT. Because there are most research and articles on SAD and CT, they are the two expected to give the best result.

For the feature-based methods, ORB scores best considering both computational cost and accuracy. It is the most recently developed algorithm and was published as an efficient alternative to SIFT and SURF. Therefore ORB is expected to yield the best results overall. For the algorithms matching descriptors, FLANN is anticipated to be preferred due to the high number of pixels and thereby many descriptors to match.

Chapter 4

Implementation

4.1 Hardware

The sensors used, the chosen setup, and the calibration is the foundation for understanding the application's potential and limitations. Understanding the key strength and weaknesses of the setup and how the real world is represented in the sensor data is, therefore, of great importance. This chapter presents the sensors used, how the application is set up, the calibration, and how the algorithms are implemented.

4.1.1 Camera

Both of the cameras used in the stereo setup is delivered by FLIR, and the camera model is called Blackfly S GigE. The selected lens was bought from Edmund Optics and is in the C Series. The most relevant specifications are written in Table 4.1, for further reading, see the suppliers' website; *Link Camera specification*, *Link Lens specification*.

Reading from Table 4.1, the sensor uses the Gigabit Ethernet or "GigE" interface, which is known to combine multiple cameras easily [1]. It is the most common interface for industrial image processing, and offer the possibility for receiving power through the GPIO pins or with Power over Ethernet. The sensor type is CMOS, and it utilizes a global

Camera specification	
Frame rate	24 FPS (min. 1 FPS)
Resolution	2448 x 2048
Pixel Size	$3.45\mu m$
Sensor	Sony IMX264, CMOS, $2/3$"
Readout Method	Global shutter
Interface	GigE
Operating Temperature	$0°C$ to $50°C$
Exposure Range	$13\mu s$ to $30s$
Dimensions (W x H x L)	$29mm$ x $29mm$ x $30mm$
Power Requirements	Power over Ethernet (PoE); or via GPIO

Lens specification	
Focal Length	$8.5mm$
Type	Fixed Focal Length Lens

Table 4.1: Camera and Lens spesicifation

shutter to capture an entire frame all at once. Since the cameras are to be deployed on a moving vessel, this enables capturing all of the motion simultaneously, resulting in no rendering of slanted lines, which should have been straight.

The sensor format is used to determine the field of view. Together with the lens specification, it was calculated to be 59.1 degrees; see Section 2.2.2. Combined with the high resolution, the camera will be able to detect and recognize objects on farther distances. In disadvantage, the number of pixels is tremendous and thus also the computational burden when iterating through the pixels in stereo processing. Moreover, the pixel size is microscopic, which gives rise to noisier images in case of little light. Considering that the image processing will have a notable higher runtime than $\frac{1}{24}$ seconds, the frame rate is acceptable even if one decides to lessen the image resolution.

The focus and "iris" of the sensor are manually adjusted on the lens, and can later be fine-tuned in software. The camera supports color images, but considering the stereo matching algorithms is intensity-based, the camera is set to capture greyscale images to maximize the frame rate. Greyscale images are the same as intensity-based images. By summing all the colored layers, $\frac{R+G+B}{3}$, a color image is transformed into greyscale.

The cameras are not waterproof, and together with the high operating temperature, it is necessary to make a waterproof and isolating case for protecting the cameras.

In consideration of stereo matching, it is important to set the manual exposure and focus of the cameras approximately equal. This will reduce noise when comparing corresponding pixels in the image processing part. Related, it is also essential to synchronize the capturing of the two cameras, so images will appear in the correct order when post-processing.

4.1.2 Synchronized Capture

Synchronized capture is when two or more cameras take pictures at ”the same time.” With the cameras provided, there are three ways to achieve this. Either trigger both cameras at the same time with code, use an external hardware trigger, or create a physical connection between the cameras linking their GPIO pins. The latter seemed to be the best choice due to less hardware, and it yields better results than code-triggering. To achieve this, a master-slave architecture is implemented, where the left camera triggers the right. When the master camera starts capturing an image, a strobe occurs, and this is used to trigger the second camera. The frame rate of the secondary camera will thus follow the master cameras frame rate. To implement this, a physical connection linking the cameras GPIO pins is used.

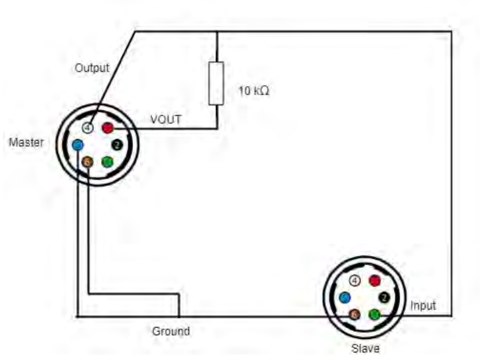


Figure 4.2: GPIO connection circuit

A sketch of the implementation can be seen in Figure 4.2. The master’s output voltage pin, 3.3 V, is connected to the secondary’s input voltage pin. Since the optoisolated output is used, the strobe signal from the master camera needs to be strengthened. This is achieved by using a pull-up resistor. A 10KΩ resistor is connected to the master camera’s non-isolated output voltage and its optoisolated output to the secondary camera’s non-isolated input. Next, the master’s ground is connected to the secondary’s ground. The optoisolated ground on master is also connected to the secondary’s ground. [6]

Figure 4.3: GPIO connection circuit

Two hirose-cables were connected to each camera, as described above. To get the length needed between the cameras’, an extension was added between the cables. A heat shrink tube was added over the connections to make it waterproof, see Figure 4.3.

Next, the cameras need to be configured. This is done by using the demo program SpinView available with Spinnaker SDK, FLIR's GenICam3 API library [6]. For the master camera, the 3.3 V line is enabled. For the secondary camera, the trigger source is set to the input voltage pin, and the Trigger mode is set On. The trigger overlap is set to Read Out, which means that the camera will start capturing as soon as the image area has been cleared.

4.1.3 Stereo setup

The fixation point was initially chosen to 100 meters to get a sense of the requirements needed for implementing stereo vision on long-distances. In lack of large enough baseline-setup, the maximum baseline achieved where 1.753 meters, giving an optimal detection on 52meters by the 1/30-rule. This was the initial setup, but because the stereo setup was going to be used in a complementary report calibrating the stereo cameras together with a LiDAR, the setup had to be changed optimized to a shorter distance. The new optimal fixation point for the stereo setup was hence set to 4 meters.

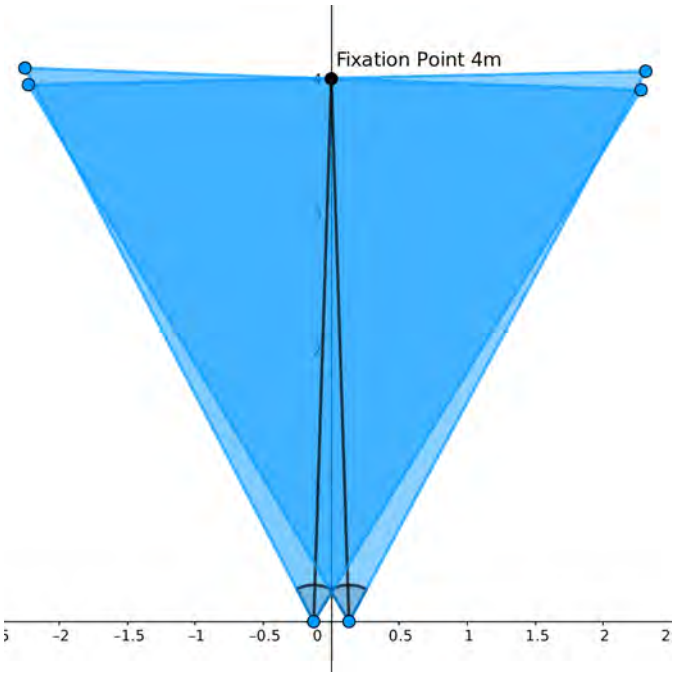


Figure 4.4: FOV stereo setup angled 1.891 degrees

Given a fixation point on 4 meters, the rule of thumb suggests a baseline on 13 cm. Because it was desired to angle the cameras for minimizing the estimated depth error, hence the baseline was chosen to 26cm. With a baseline on 26cm and a fixation point on

4m, the camera's vergence angle is calculated by Pythagoras:

$$\arctan \frac{26/2}{400} = 1.861^\circ$$

The angled system is theoretically shown in Figure 4.4, the two black lines in the middle imitate the fixation point of each camera, meeting at 4 meters. The real stereo system is

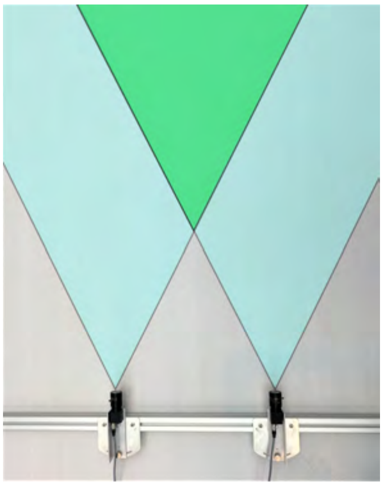


Figure 4.5: Camera setup

shown in Figure 4.5 with the angle and baseline fixed to the end resulting setup. Because the cameras have a wide FOV, the blind spot is only 22cm horizontal distance, depending on the accuracy of the camera's angle in practice.

The setup of the system is designed to easily be adjusted, using screws underneath to tighten when the desired baseline and angle are achieved.

4.2 Software

The software on the autonomous ferry Milliampere is built upon Ubuntu 16.04 LTS and the open-source robotic framework Robot Operating System (ROS). Hence, everything used is selected to be able to integrate with the already running system.

4.2.1 ROS

ROS is a meta-operating system for robots, providing device drivers and communication between multiple processes. It is known for hardware abstraction, low-level device control, and message-passing between processes. The ROS runtime graph is a peer-to-peer network of processes that can be run on multiple machines. This allows the stereo-system to be developed in a process on a single machine for later be integrated with the rest of the system by just redirecting messages.

ROS already had an implemented driver for the camera used; it can be found here: *Github camera driver*. The chosen driver is implemented in C++ and was the one with the least issues and most recent updates.

4.2.2 OpenCV

All the software implemented is based on the third-party library OpenCV (Open Source Computer Vision Library). It is a library for computer vision and machine learning applications, including both classic and state-of-the-art algorithms. The library has interfaces for MATLAB, Python, and C++, where the latter was used to implement the stereo matching and 3D reconstruction. OpenCV in C++ generally yields more functionality and more parameter options.

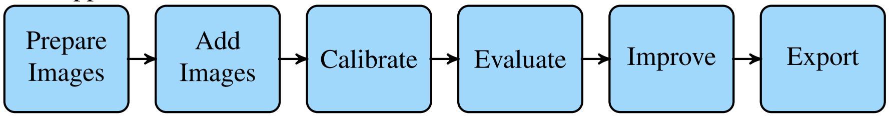
4.2.3 MATLAB

MATLAB is numerical computer software to analyze data, develop algorithms, and create models and applications. The Computer Vision Toolbox has built-in apps for both monocular camera calibration and stereo calibration. Both of the apps are used during the calibration, and they implement the calibration algorithm by Zhang's method described in Section 2.1.2. By uploading images of a calibration pattern in different orientations, the app calibrates and provides tools for evaluating and improving the calibration. An estimation of the intrinsic, extrinsic, and lens distortion parameters is outputted.

4.3 Calibration

Precise calibration is required for 3D interpretation of images and reconstruction of world models. When calibrating on longer distances, it was found to yield the best result calibrating the intrinsic and extrinsic parameters in two separate sessions. Therefore monocular camera calibration app first was used to calibrate the intrinsic parameters for each camera, subsequently using the stereo calibration app to obtain the rigid body transformation between the two cameras. This achieves better results for the intrinsic parameters of the system because the monocular calibration can be done at a shorter distance.

Both applications use the same workflow:



Further is an explanation of the stereo calibration done with the resulting parameters at the end.

4.3.1 Prepare calibration

When the calibration parameters are calculated, the accuracy of the result depends on the image resolution and the precise measurement of the calibration object. The calibration object used is a 5x7 squared checkerboard with each square measured to 10.1cm. It is printed on aluminum to make the board even and unbroken minimizing imperfections and inaccuracy. When capturing images, it is vital to keep the calibration object in focus and keep the camera parameters constant. Changes in focus or zoom will result in miscalculated focal length.

MATLAB states that the checkerboard should fill at least 20 percent of the captured image, and one should use at least 10 to 20 images of the calibration pattern at different angles and positions. This was achieved in the monocular calibration. Some of the images

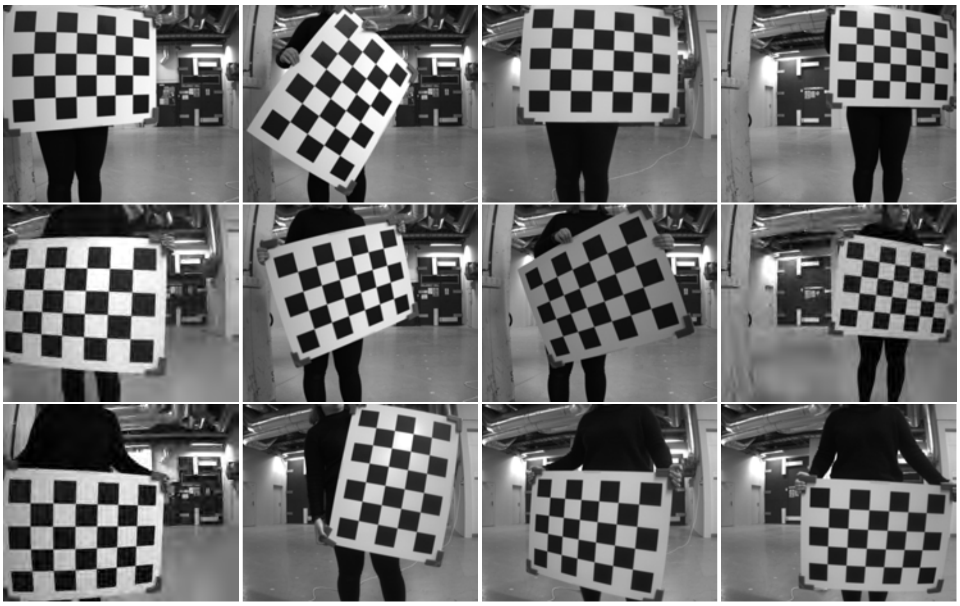


Figure 4.6: Poses of checkerboard

used in the monocular calibration are displayed in Figure 4.6. It is essential to capture images of the pattern at different positions and orientations. Additionally, the calibration board must be captured as a whole to account for the lens distortion.

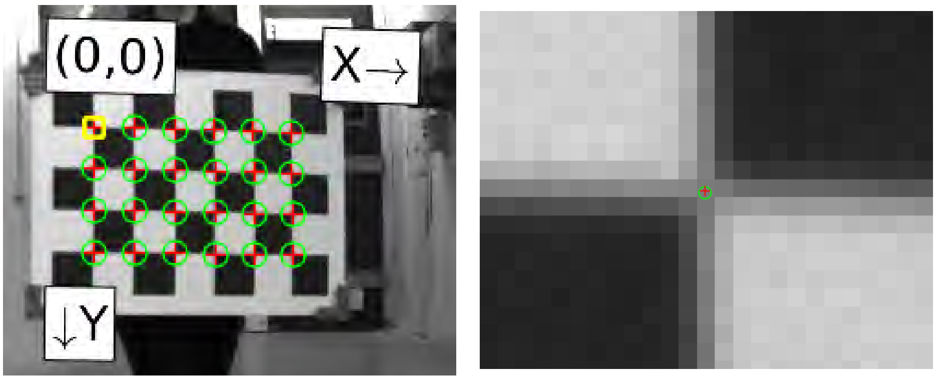
When calibrating the extrinsic parameters in the stereo calibration app, the pre-computed fixed intrinsic parameters from the monocular calibration were loaded. The calibration images now should be captured at a distance roughly equal to the operating distance [25], i.e., 4 meters away from the camera. This is not practical taking into account that the checkerboard should fill at least 20 percent of the captured image [25]. Furthermore, it is crucial to capture the stereo images at the same time, and the pattern has to be fully visible in both images. To compensate for the size of the pattern, more than 400 image-pairs were loaded into the app to cover the different poses. By calibrating several times to improve the error, the image-pairs were reduced to be 275.

4.3.2 Evaluation

There are several ways to evaluate the calibration accuracy. All the following methods explained were examined to improve the result further and re-calibrate.

Examine reprojection errors

The app displays the calibrated images with green circles around the corners detected in the image, and red cross for the reprojected 3D points. The reprojection error is measured in Euclidean distance in pixels and is the distance between the center of a green circle and the corresponding red cross. In Figure 4.7b, one can see that both the green circle and the



(a) Picture used in calibration **(b)** Zoomed into one of the corners

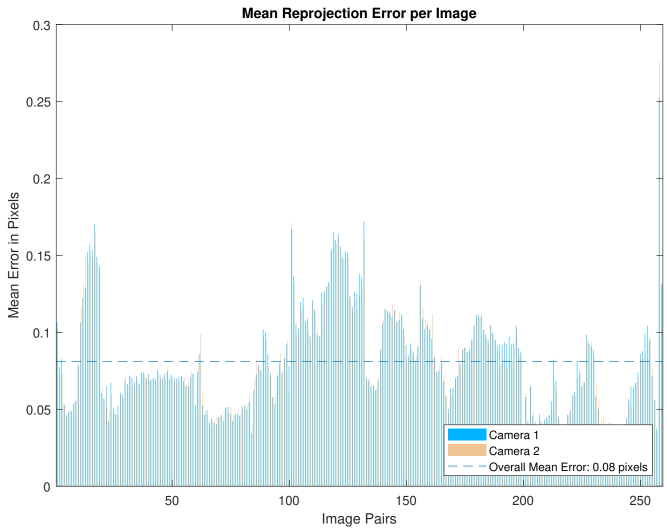
Figure 4.7: The distance between the red cross and the green circle's origin is the reprojection error

red cross is within the same pixels, meaning the reprojection error is less than a pixel. It is preferred to have a reprojection error of less than one pixel.

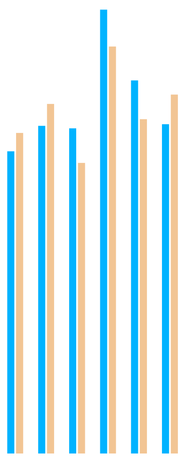
The mean reprojection error per image is plotted as a bar graph. In figure 4.8, the final result of the reprojection error is displayed. The overall mean error, shown as the horizontal stippled line, is 0.08 pixels. The highest spikes correspond to images that adversely contribute to the calibration error. In theory, some of the most significant spikes should have been removed. However, as the spikes correspond to rotated images and images captured in corners of the image plane, they need to remain to cover all variations of rotation and translation needed. The low mean reprojection error and the approximately equal error between corresponding image pairs makes this an overall satisfying result. The reprojection error alone is not a trusted measure to verify the accuracy of the calibration. Because if all the pictures where taken in the same region, this would have minimized the reprojection error, but given erroneously results when rectifying and undistorted the images after the calibration.

Examine Extrinsic Parameter Visualization

MATLAB's 3D extrinsic parameter visualization was used to discover errors in the calibration. By plotting the camera-centric view of the checkerboard's position in each image



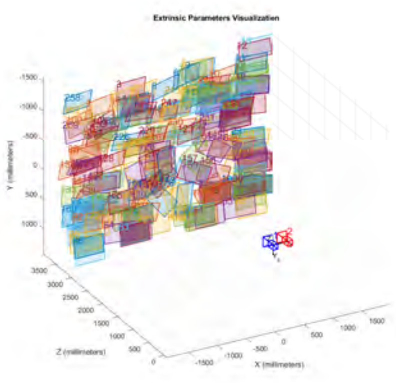
(a) Reprojection Error Stereo Calibration



(b) Zoomed

Figure 4.8: Reprojection Error Stereo Calibration

pair.



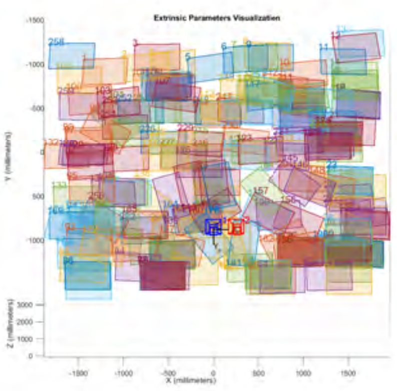
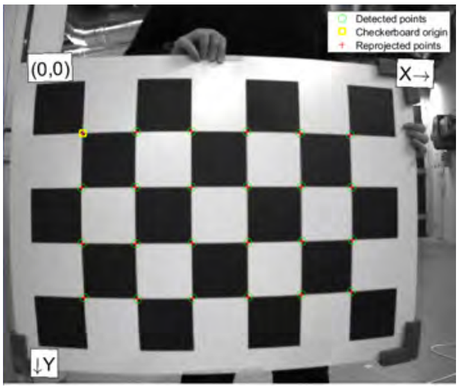


Figure 4.9: Camera Centric visualization of the checkerboard's position in the captured images

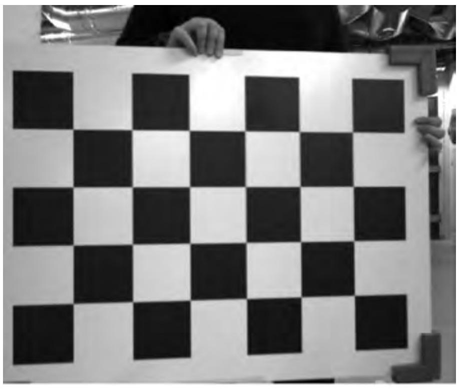
In Figure 4.9, the location of the calibration pattern is plotted in the camera's coordinate system. The cameras are displayed in dark blue and red. This is used to check if there are enough pictures for the checkerboard to cover the area and see if the relative positions match the expectations. New images were added to the calibration, and redundant ones removed to improve the calibration further.

View undistorted image

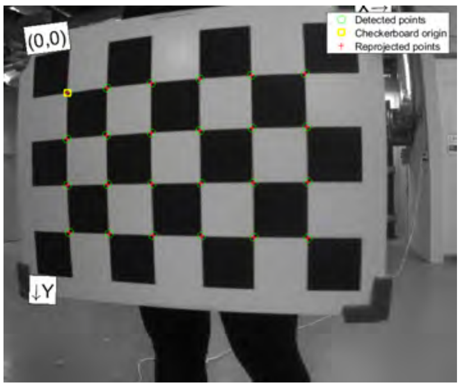
The distortion estimation might be incorrect even if the reprojection errors are low. This can appear if the checkerboard covers only some regions of the image. If the undistorted images seem wrong, e.g., the checkerboard is curved or unrealistic images, one can change the number of parameters of radial distortion. The cameras in use have barrel (positive radial) distortion; see Section 2.1.1, which usually yields the best result with only two coefficients. The result after undistorting the images from both cameras is shown in Figure



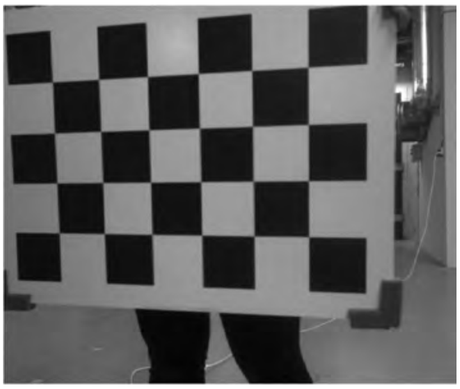
(a) Distorted left camera



(b) Undistorted left camera



(c) Distorted right camera



(d) Undistorted right camera

Figure 4.10: Before and after calibration of each camera

4.10.

Show Rectified Images

One of the primary goals of calibrating the stereo camera is to be able to row-align the image pair. Looking at the rectified images is thus a reasonable basis for evaluating the calibration. The rectified images should find the same world point in both images along the same line. This can be measured by eye. This is an important step, even if the reprojection errors are small. If the images cover too small of an area, the distortion estimation might be incorrect even though the reprojection error is small.

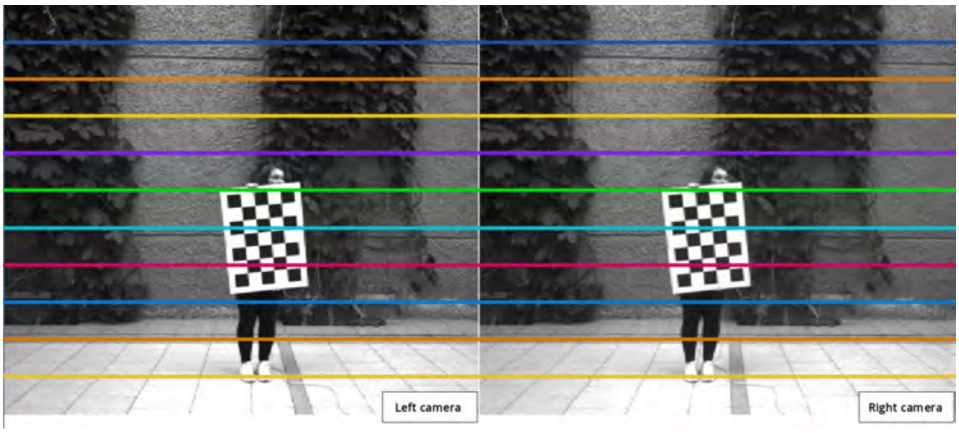


Figure 4.11: Rectified images after stereo calibration

Figure 4.11 shows the final result after calibrating and rectifying the images. As shown, the distortion is removed, and the colorful lines match points in the two views.

4.3.3 Calibration parameters

The intrinsic parameters for the left and right camera were calibrated to be:

$$K_L = \begin{bmatrix} 1229.4 \pm 0.5 & 0 & 618.4 \pm 0.3 \\ 0 & 1228.7 \pm 0.5 & 536.7 \pm 0.2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Radial distortion coefficients for the left camera:
$k_1 = -0.3891 \pm 0.0005$, $k_2 = 0.1732 \pm 0.0011$

The mean reprojection error for the left camera: 0.081

$$K_R = \begin{bmatrix} 1229.0 \pm 0.7 & 0 & 640.1 \pm 0.2 \\ 0 & 1228.4 \pm 0.7 & 533.1 \pm 0.2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Radial Distortion coefficients for the right camera:
$k_1 = -0.3875 \pm 0.0005$, $k_2 = 0.1681 \pm 0.0009$

The mean reprojection error for the right camera: 0.081

The mean reprojection error does not say much alone, but because of the high resolution of the cameras, the error is acceptable as long as it is below one pixel. In theory, the intrinsic- and radial distortion parameters for the two cameras should have been equal, but because the screw-lenses have to be set manually to the desired position, it is expected to be some differences. The parameters for the focal length are almost identical, but for the principal point on the x-axis, the values are noticeable divergent. Bear in mind that the principal point is defined to be the image position where the optical axis intersects the image plane. With this in mind, the principal point will change if the camera setting changes, e.g., changing the focus or zoom. Hence, a small distinction in the rotation of the two lenses will yield different results for the principal point's parameters.

It should be mentioned that both the skew parameters are set to 0 in MATLAB before calibrating. It is possible to allow for a non-zero skew, but with newer cameras, the skew-parameters is usually negligible. In other words, an extra free parameter to calibrate in most cases only give rise to more noise in the calibration. It was tested to calibrate with the skew parameters as well, but it did not give any noticeable enhancement to the result.

Extrinsic parameters for the stereo setup is given relative to the left camera. Hence, the rotation matrix and translation vector is given from the right camera relative to the left measured in mm:

$$\mathcal{R} = \begin{bmatrix} 0.9986 & -0.0000 & -0.0530 \\ 0.0001 & 1 & 0.0006 \\ 0.0530 & -0.0006 & 0.9986 \end{bmatrix} \pm \epsilon < 10^{-5} \tag{4.3}$$

$$\mathbf{t} = \begin{bmatrix} -260.51 \pm 0.05 & 2.24 \pm 0.07 & 6.79 \pm 0.02 \end{bmatrix} \tag{4.4}$$

For the stereo setup, the x-axis in the coordinate system is defined to be with the baseline, positive direction from the left camera to the right, y-axis pointing downwards, and z is the depth straight forward from the left camera. In the translation matrix, this means the stereo cameras are 260.51mm apart, it was measured by both ruler and a laser measure to be approximately 260mm. For the stereo setup, it was never made any effort to make the cameras be perfectly symmetrically aligned, thus it is reasonable to accept that the right camera is 2mm above and 7mm in front of the left camera considering they are both angled.

The rotation matrix converted to Euler angles, in degrees, corresponds to $[x : -0.0343, y : -3.0381, z : 0.0019]$. As expected, the change is greatest around the y-axis. In theory, it should have been $1.89 * 2$ degrees, but the angles were adjusted by hand using a protractor. Hence some error is expected. The rotation around the x- and the z-axis is negligible compared to the inaccuracy of the setup.

Overall, both the intrinsic- and extrinsic parameters are more accurate than expected-especially the extrinsic, which was calibrated using a checkerboard limited in size. In practice, the checkerboard should have been > 5.3 meters in width when calibrating the extrinsic parameters.

4.3.4 Calibration on long distance

The calibration was first done at the initial stereo setup, calibrating on 10meters for testing purposes. Since the checkerboard only measures 0.505x0.707 meters, it was hard to capture the checkerboard at all positions to yield satisfactory calibration parameters. This includes a significantly higher amount of calibration pictures, making it difficult and tedious to evaluate the calibration. Hence, when calibrating on longer distances, one should try to obtain a bigger calibration object. Considering the camera's field of view, the preferred calibration object on 100 meters should exceed 130 meters in width. The calibration object does not have to be a checkerboard, as long as the object has known measures.

4.4 Implementing stereo matching algorithms

Before implementing any of the stereo matching algorithms, it was written code to undistort, rectify, and histogram equalize the stereo image pair. The rectifying and un-distorting is a pre-request before using the correlation-based methods. The histogram equalization was applied for contrast adjustment to improve the contrast in the images by stretching out the intensity range of the image.

As a starting point, built-in algorithms were tested, but a lot of the open-source and built-in algorithms are poorly documented. This made it hard to compare due most of the algorithms used a lot of pre-filtering and post-filtering to yield better results. Some were also optimized in threads to achieve better runtime on either GPU or CPU. This in mind, the focus was adopting and writing code, which could be explained. Preferably code in which one can say what is happening in the background and with input parameters to tune. This yields a good understanding of the different algorithms used and further easier to separate which could be a better choice in a difficult scene. The algorithms are thus not optimized considering running-time, but some sense of the running time is achieved. However, most of the already inbuilt functions were tested, and usually, they seem to yield similar results even if it is hard to explain precisely why and what they use in the source-code.

The result and discussion are written in a combination of the sense of running times achieved, together with resources online. It is also worth noticing that today's computer vision programs are generally optimized for pictures around 400x400 pixels, which is today's state of the art computer vision processing. So, as expected, some of the algorithms used several minutes, up to hours to compute, and thus, the input images had to be down-scaled. This is not an optimal solution considering detecting objects on farther distances will capture only a few pixels and may ”disappear” when down-scaling. As a solution, one should, for example, crop the images around to the desired area.

As already mentioned, most of the code is implemented in C++ using the OpenCV, using MATLAB to plot some of the figures.

4.4.1 Correlation based methods

Correlation based methods directly depend on the window size chosen. In general, a smaller window will yield good precision and more details but is sensitive to noise. On the other hand, a larger window will give robustness to noise, but both precision and details are reduced, and hence the resulting disparity map will become more blurry. For a good stereo correspondence, one should aim that every pixel will find a good match in the other image, and if two pixels are adjacent, they usually will hold the same disparity. The latter is tuned by setting the maximum and minimum disparity in the scene to filter out some of the wrong matches, i.e., removing noise.

Sum of Absolute Differences and Sum of Squared Differences

SAD seems to be the most used algorithm considering the amount of code online, both directly implemented but also optimized code. However, most of the optimized code was not well-documented. Thus, both SAD and SSD were implemented using direct code with many for-loops. So the code is not optimized for running-time. After comparing, they were both tested with built-in optimized code to see the performance increase and further using a filter to get a more usable result considering triangulation. Different windows were tested to achieve the optimal matching window for the scene used for testing, see Figure 5.6. Bear in mind that variation in the window size is a compromise between the computation cost and accuracy.

Normalized Cross Correlation

The normalized cross correlation was as well implemented using both direct code and more in-built ones with more or less documentation. In total, NCC is associated with a high computationally cost, and there were not found any code which could have the potential to run in real-time.

Rank Transform

It was implemented by first applying the rank transform to the image with various window sizes and further matching the two rank images with the SAD directly implemented. A quadratic window was applied in both cases.

Census Transform

Equivalent to RT, the census transform was first applied with the census transform with different window sizes and accordingly matched with window sizes of various quadratic sizes.

4.4.2 Feature based methods

For the feature-based methods, they are all implemented with two different approaches. First, by using the four methods to both detect keypoints and make descriptors to match. The methods are designed to find points of interest, keypoints, which are robust, re-identifiable points. The number of keypoints found can be tuned in the algorithms, but even maximizing this parameter will usually yield a sparsely distributed disparity map, i.e., few pixels to be matched. This results in a sparse 3D reconstruction, which can be hard to read, hence the second approach was tried. The second approach sets all pixels as keypoints for further describing the keypoints (creating descriptors) with the four feature-based algorithms respectfully. For both the approaches, the descriptors are matched using both BF and FLANN respectfully. Based on 3.2.5, FLANN is implemented using multiple randomized k-d trees.

Because the descriptors are based on local textures, the matches do not necessarily follow any global geometrically consistency. This in mind, a threshold test is applied to remove mismatches. By applying an adjusted cross-check-test, one can remove disparities (distances) with greater or less distance than desired. The cross-check-test is applied with different ranges for filtering out descriptor-matches-lengths to remove matches that are less than the minimum or above the maximum range. This can remove some of the noise constructed in images with repetitive patterns or sophisticated structures, i.e., filtering out some of the erroneous matches.

Harris Corner Detector

HCD was implemented in both Matlab and C++ using the OpenCV library. Somehow the algorithm in C++ managed to detect some more corners, even though they are both based on the same library. The most relevant parameter to tune is the quality or the threshold R for the corners detected; see Section 3.2.1. Further, while matching, the threshold for how different a corner can be to be considered a match is tuned.

SIFT

It is implemented using the built-in SIFT class in OpenCV. As described above, the most relevant parameter to tune is first the number of keypoints to detect and, subsequently, the cross-check-test to filter out mismatches. For the matching, Euclidean distance is used as distance measurement, as described in Section 3.2.5. The other inbuilt parameters in the SIFT class were also tried tuned to yield the best result in beforehand. This includes the number of octaves (layers in each pyramid image, the number of pyramid images can not be tuned), the Gaussian filtering used, and the thresholds for an edge and for filtering out features.

SURF

Similar to SIFT, but here the threshold for the Hessian matrix must be set instead of the edge threshold. In addition, the number of pyramid images can be tuned. There is also a choice to use a descriptor of size 128, but this was not used because it makes it almost equal to the SIFT algorithm, reducing some of the improved running-time.

ORB

For the ORB algorithm, the most considered parameters to tune are the number of key-points found and the ratio while doing the cross-check-testing. The ORB-class has inbuilt functions to tune the pyramid scaling factor, the number of pyramid images, the threshold for edges. Because the ORB descriptors are binary, the Hamming distance is used for comparing matching descriptors.

Chapter 5

Experiment

An accurate disparity map achieves a better depth estimate from the triangulation. It is possible to filter the image before processing the disparity map or filter the disparity map afterward to remove noise and achieve a smoother disparity map. Finding the right filter is time-consuming, and there exist numerous filters to obtain an optimal result. Besides, every algorithm will usually perform better with different types of filters, depending on the scene. With this in mind, filters are not used to simplify the comparison between the different algorithms. Filters are another research area, and choosing the right one for the application is something that should be done after choosing the best algorithm. However, keep in mind that every disparity map can be improved by applying filters to obtain better triangulation. As an example of the result which can be obtained, the first algorithm's disparity map are filtered.

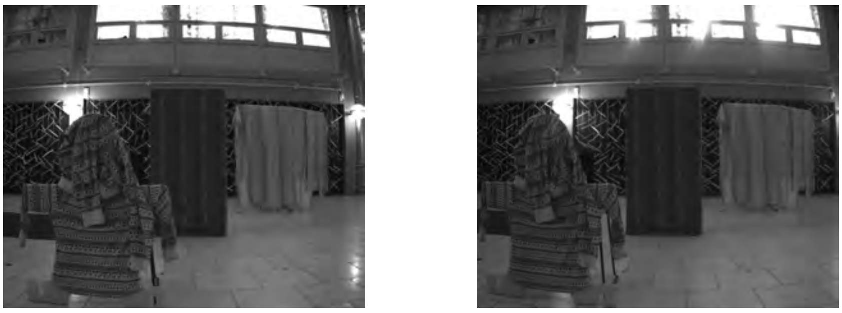


Figure 5.1: Image pair directly from the camera

The two fresh images captured directly from the cameras is shown in Figure 5.1. Before applying any of the matching algorithms, the picture pair have been undistorted and rectified, resulting in two images with 1069x1242 pixels respectfully. The two images used in the stereo matching is shown in Figure 5.2. This is the image pair after undistorting, rectifying, and histogram equalization. When the disparity maps are displayed- areas with no

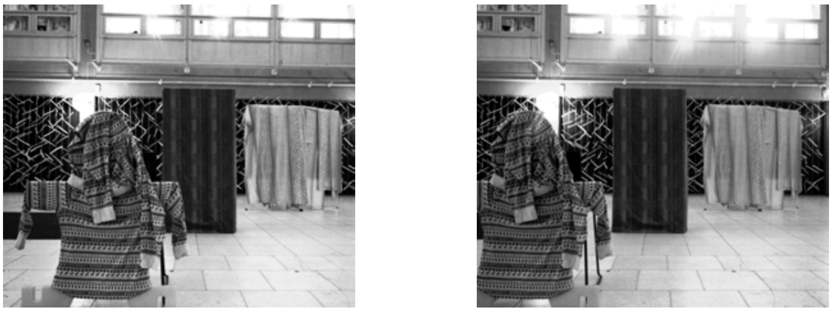


Figure 5.2: Image pair used in stereo matching

matching region, the leftmost in the left image and rightmost in the right image, is cropped.

5.1 Ground Truth

Ground Truth refers to the accuracy of the data collected. In depth-estimation, the ground truth is the real depth and tells us how accurate the depth estimation is. In the experiment of comparing disparity maps obtained from different algorithms, a fixed scene is created as the Ground Truth. The scene contains three objects with distinctive structure, located at separate distances to try to isolate them in the image processing. The arena and the three objects location is depicted in Figure 5.3.

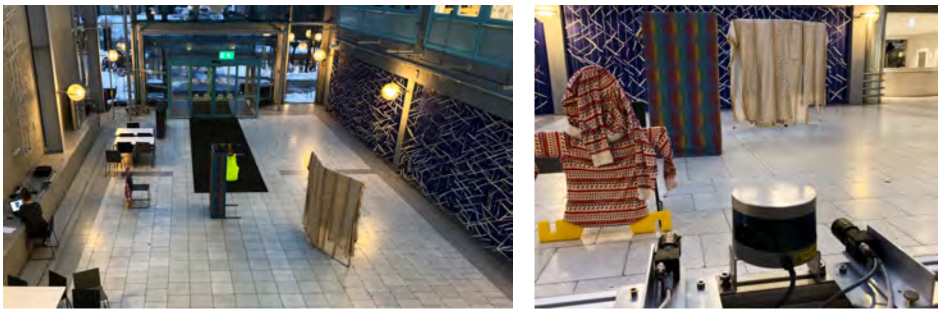


Figure 5.3: Scene used as Ground Truth

By the camera setup, the fixation point is determined to be 4 meters. The objects were placed at approximately 2, 4, and 6 meters, respectively. The distance to the background, or wall, was measured to be 9.84m. The scene is measured by a LiDAR placed in between the two cameras. The uncertainty in the measurement and the LiDARS position relative to the cameras is approximately 1 cm. The LiDAR measured the three objects to be placed at 1.85, 3.95, and 6.03 meters average.

The surface structure of the objects is chosen with the purpose of matching the corresponding pixels easier. The feature-based methods require corners and changes in the pattern to be able to detect keypoints. Likewise, the correlation-based algorithms perform poorly when matching images with no changes in intensity. The background was mainly chosen to compare how the stereo matching algorithms manage such a cluttered background. In a real-world scenario, there will be noise and inconvenient lighting conditions, and it is interesting to compare how the different methods manage to separate the objects from the background. Considering some of the methods are known to be more resistant against image noise and brightness differences.

To acquire the Ground Truth disparity map– two of the stereo images were rectified for further measuring the disparity between central pixels. The disparity was measured in MATLAB by plotting both the images in red-cyan anaglyph. Anaglyph is a stereoscopic 3D effect that fuses the brain into a perception of a 3D scene when using red-blue stereo glasses. The number of pixels measured is equal to the correct grayscale value in the Ground Truth disparity map. The Ground Truth depth map was thus refined in an image editor. Eventually, a combination of median filtering and Gaussian bilateral filtering was applied to create a notion of a more realistic disparity map. In Figure 5.4, the disparity

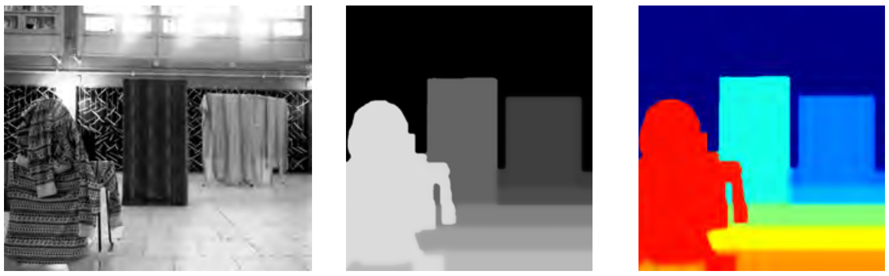


Figure 5.4: Rectified right image, and Ground Truth disparity map

map is displayed in both color and gray-scale, combined with the scene captured by the right camera. The disparity map is normalized to be in the range [0,255], where each pixel

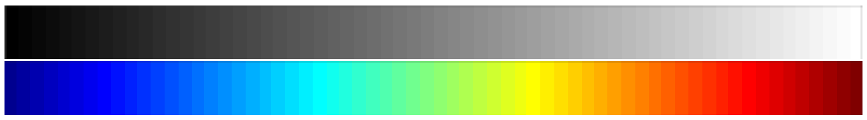


Figure 5.5: Color-map

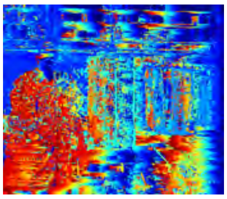
in the range corresponds to a color shown in Figure 5.5. Zero disparity corresponds to black or dark blue, and pixel-disparity of 255 is plotted in white or dark red. In the ground truth disparity map, every pixel is between [35-211], and hence, when later plotting all the disparity maps created by the algorithms, this range is used to filter out some of the noise.

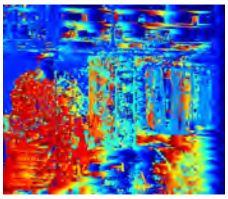
5.2 Correlation based methods

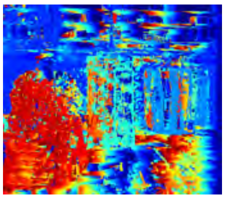
The correlation-based algorithms require a distance parameter to specify how far from the template location one wants to search for correspondences. For convenience, the algorithms search for correspondences in both directions; this is user-friendly because one does not have to consider what is the right and left image. However, in consideration of real-time applications, this should be removed to reduce the running time. The minimum disparity is set to 35 and the maximum to 211 pixels to match the scene. This will remove some noise, and additionally reduce the running time compared to searching the total width of the image.

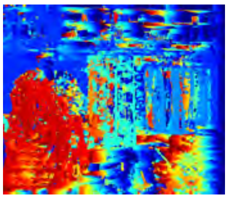
5.2.1 Sum of Absolute Differences

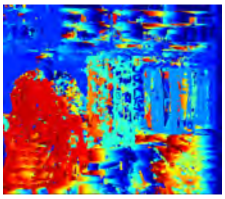
Using the correlation-based algorithm SAD on the image pair in Figure 5.2, yield different results depending on the window size. The window is square, and varying the window size

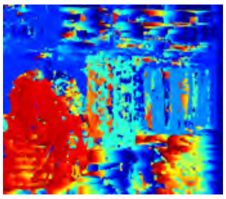

(a) $W = 7x7$

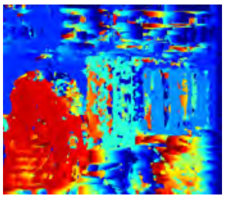

(b) $W = 9x9$

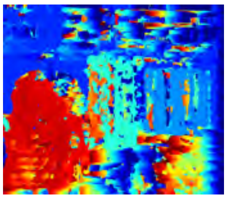

(c) $W = 11x11$

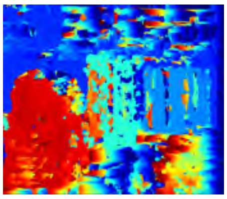

(d) $W = 13x13$

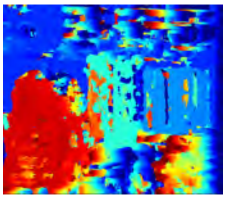

(e) $W = 15x15$

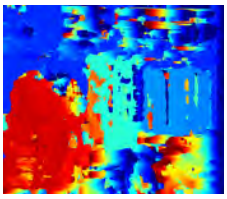

(f) $W = 17x17$


(g) $W = 19x19$


(h) $W = 21x21$


(i) $W = 23x23$


(j) $W = 27x27$


(k) $W = 31x31$

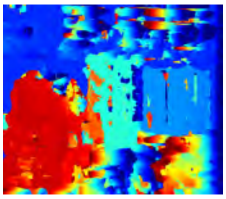

(l) $W = 35x35$

Figure 5.6: Tuning the window size in the SAD algorithm

yields the results in Figure 5.6. As expected, the disparity map gets smoother by increasing the window size, and because the objects take a lot of the image-area, greater window size is preferred. However, increasing the window size will increase the computation time. An alternative is to reduce the image resolution to lower the computation cost.

Nevertheless, the algorithm outputs an acceptable disparity map. Some of the noise still needs to be removed to be able to segment out the objects in a 3D world map, but this is achievable using filters. The code used is nowhere optimized considering computation-time, but it gives a sense of the window size needed to yield a decent disparity map.

To test how the algorithm will perform using optimized code, some of the built-in functions in OpenCV are used. Starting with scaling the image to half the size, and applying a block-window on 9x9 pixels. This will correspond to a window of size 18x18 in the original sized image. The result from the scaled image is as expected a bit noisier than the original sized, but it is more agile. The disparity map is further smoothed by a filter to be able to create a convenient 3D map. The filter used is based on Weighted Least Square– and use a fast smoother to refine the result in half-occlusions and uniform areas [26]. In Figure 5.7 the filtered disparity map is shown. The leftmost image is the scene

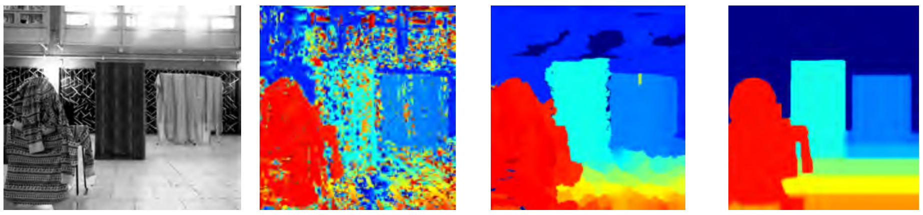
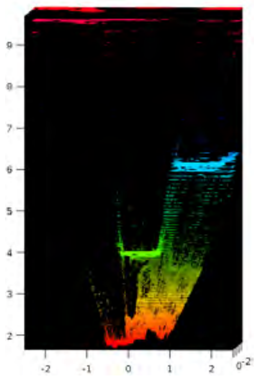


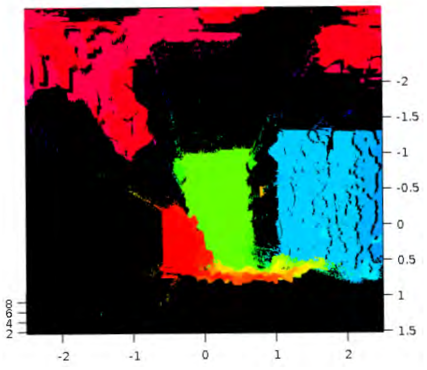
Figure 5.7: Right rectified image, matching with SAD, filtered with WLS, Ground Truth

captured from the right camera, following is the disparity map created by down-scaling the image-pair and applying SAD-matching, then the filtered disparity map is displayed with the Ground Truth to the right. The resulting filtered disparity map is quite similar to the Ground Truth disparity map. The objects seem to be in the same color, but the background has some inaccuracies. This is due to the ground truth having the background filtered out, setting the disparity to zero; this is not done when running the algorithm. For a further notion of the accuracy, the filtered disparity map is triangulated and plotted in a point cloud.

Depth in PointCloud



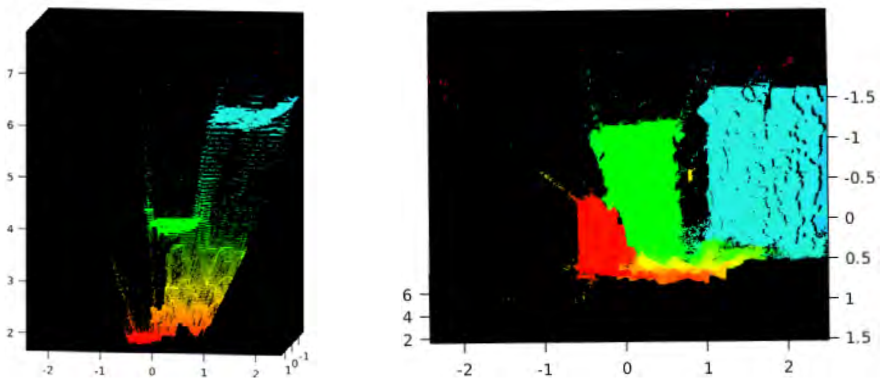
(a) Plot from above



(b) Plotted from camera view

Figure 5.8: Point Cloud, 3D map from the filtered disaprity map

The objects were by triangulation computed to average be 1.80, 3.90, and 6,04 meters respectfully. Compared to the Ground Truth that was measured to be 1.85, 3.95, and 6.03meters, it gives an overall average error of 0.03 meters— a surprisingly accurate result. The most noticeable is that the depth on the most distanced objects yields the minimum error. However, the object in front has the most uneven surface. Hence, the measurement can be inaccurate both from the LiDAR and when clustering the object. In Figure 5.8, the point-cloud from the filtered disparity map is displayed. Removing the background by



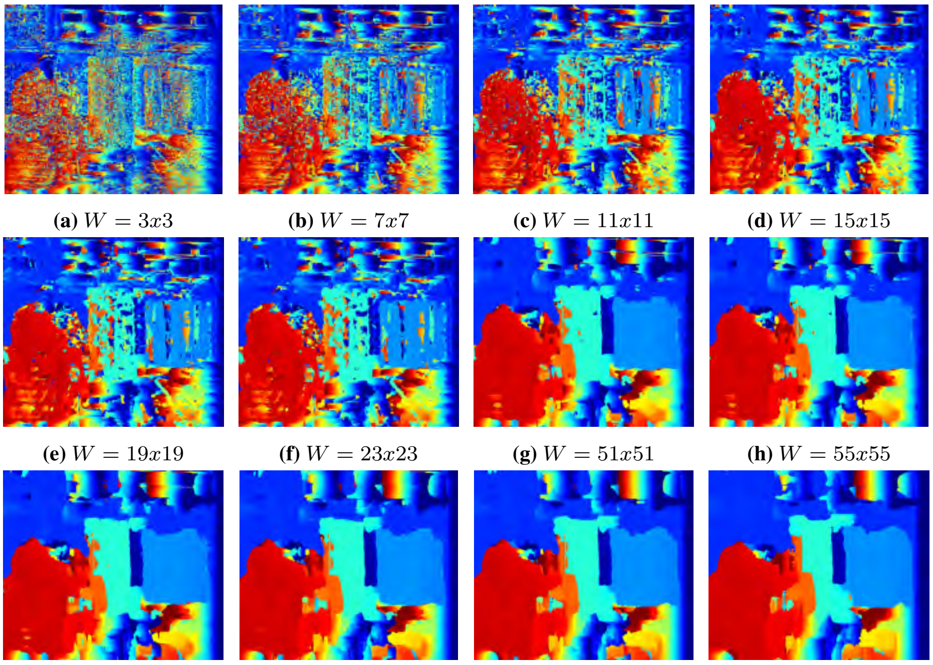
(a) Plottet from above **(b)** Plotted from camera view

Figure 5.9: Point Cloud, 3D map. Plot without the background

setting the maximum distance to 9meters yields the point-cloud in Figure 5.9. The point-cloud manages to show the three objects separated, with a negligible amount of noise. With an average distance-error on only 3 centimeters and low computation time, this is a promising algorithm.

5.2.2 Sum of Squared Differences

SSD is expected to give almost the same result as SAD-matching. Similar to SAD, SSD-matching is sensitive to outliers and since it uses squared differences in the computation, the dept discontinuities will be even harder to match. Varying the matching window yields the results in Figure 5.10. As expected, the resulting disparity map is quite similar to the one outputted from the SAD algorithm but with a negligible increase in noise. SSD is furthermore tested with some very large window-sizes, but when applied in real-time, a window-size of 101x101 is usually not feasible considering the CPU-usage. As with SAD, applying filters is preferred. The results in SSD are almost identical to SAD, and thus, the filtering and triangulation used on SAD will yield an almost identical result for the SSD.



(a) $W = 3x3$ **(b)** $W = 7x7$ **(c)** $W = 11x11$ **(d)** $W = 15x15$

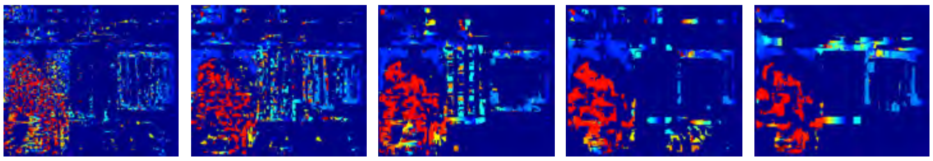
(e) $W = 19x19$ **(f)** $W = 23x23$ **(g)** $W = 51x51$ **(h)** $W = 55x55$

(i) $W = 63x63$ **(j)** $W = 73x73$ **(k)** $W = 83x83$ **(l)** $W = 101x101$

Figure 5.10: Tuning the window size in the SSD algorithm

5.2.3 Normalized Cross Correlation

NCC is expected to be robust against the different illumination conditions in the left and right captured pictures. In Section 3.1.3, it was stated to yield high computational cost, which was confirmed when no algorithms running in real-time were found. With a large window size, the algorithm runs for one hour. Reducing the running time by scaling the image pair by a factor of 0.2, it still yields a computation of several minutes. The



(a) $W = 3x3$ $(1 - 9)$ **(b)** $W = 5x5$ $(10 - 19)$ **(c)** $W = 7x7$ $(20 - 29)$ **(d)** $W = 9x9$ $(30 - 39)$ **(e)** $W = 11x11$ $(40 - 49)$

Figure 5.11: Tuning the window size in the NCC algorithm

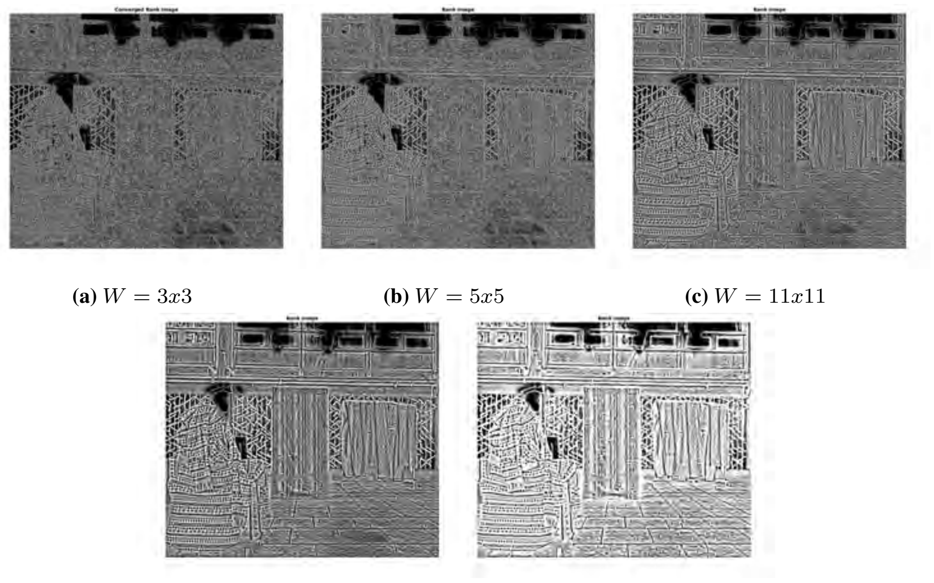
performance of the algorithm on the downscaled image pair is shown in Figure 5.11. In parenthesis, the windows corresponding to a full-sized image is written.

A window of size $7x7$ gave the best result if one were to post-filter the disparity map.

However, the algorithm is still not able to perform anywhere near real-time. The two closest objects may be possible to segment out to collect the distance, but the total disparity map is much more sparse than expected. NCC yields disappointing results overall. It managed to filter out much noise, but too much to make a disparity map for achieving a decent point cloud. However, the computational burden is too significant to be applied in a real-time system.

5.2.4 Rank Transform

Compared with NCC, RT was expected to improve performance near object boundaries and cope with brightness differences. First, the Rank Transform with different sized neighborhoods were applied to the image pair. The rank transform performed on the right image



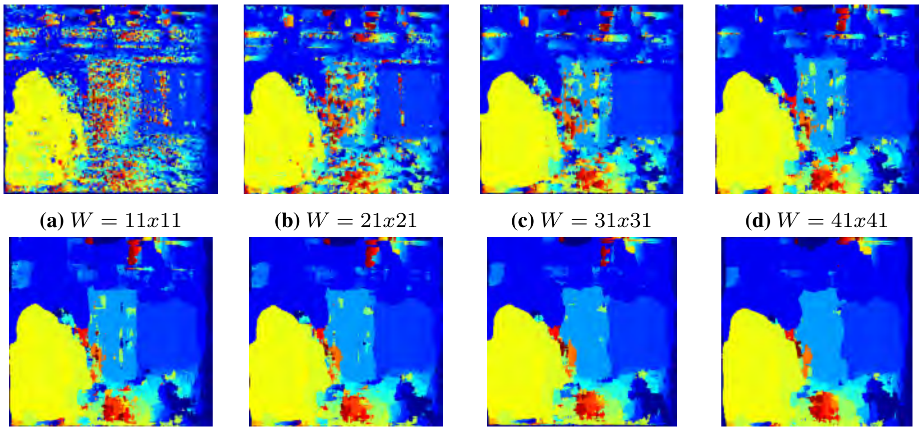
(a) $W = 3x3$ **(b)** $W = 5x5$ **(c)** $W = 11x11$

(d) $W = 17x17$ **(e)** $W = 23x23$

Figure 5.12: Tuning the size of the neighborhood in the Rank transform

is displayed in Figure 5.12. The resulting binary image gets brighter with a greater window. Subsequently, the same algorithm as used for SAD, without filter, were applied to match the image pair. The matching-window was adjusted and tested on the rank-images constructed with different window-sizes. A selection of the disparity maps created is displayed in the three Figures; 5.13, 5.14, 5.15.

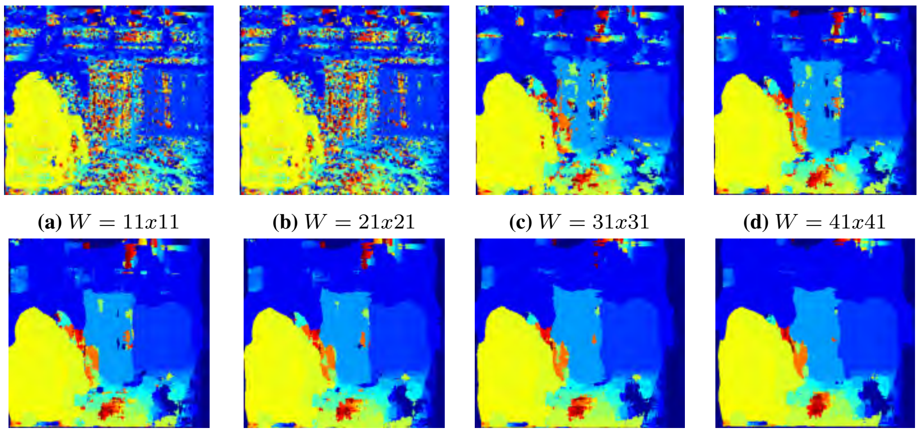
The most critical is that the colors are all wrong, meaning they will output a disparity map with different depth estimates than the SAD and SSD algorithms. The yellow color on the nearest object means a disparity of 110-130 pixels, which is nowhere near 200, which it is expected to be. All distances will therefore be almost the double of the ground truth.



(a) $W = 11x11$ **(b)** $W = 21x21$ **(c)** $W = 31x31$ **(d)** $W = 41x41$

(e) $W = 51x51$ **(f)** $W = 61x61$ **(g)** $W = 71x71$ **(h)** $W = 81x81$

Figure 5.13: Tuning Rank Transform with window 5x5, matching the images with the SAD-algorithm



(a) $W = 11x11$ **(b)** $W = 21x21$ **(c)** $W = 31x31$ **(d)** $W = 41x41$

(e) $W = 51x51$ **(f)** $W = 61x61$ **(g)** $W = 71x71$ **(h)** $W = 81x81$

Figure 5.14: Tuning Rank Transform with window 17x17, matching the images with the SAD-algorithm

Binary images are harder to match for the simple SAD approach since small patches often appear similar, making a mismatch more likely. It is reasonable to assume that a greater neighboring-window would derive a better result considering the colors, this was however not the case. Another reasonable explanation could be the scaling of the disparities, but again, the same algorithm as used in SAD is adopted. No changes are made, meaning that the algorithm struggles matching binary images.

Disregarding the colors and only examining the shapes and segmentation, RT gives valuable results. Much of the background noise is removed with a greater matching win-

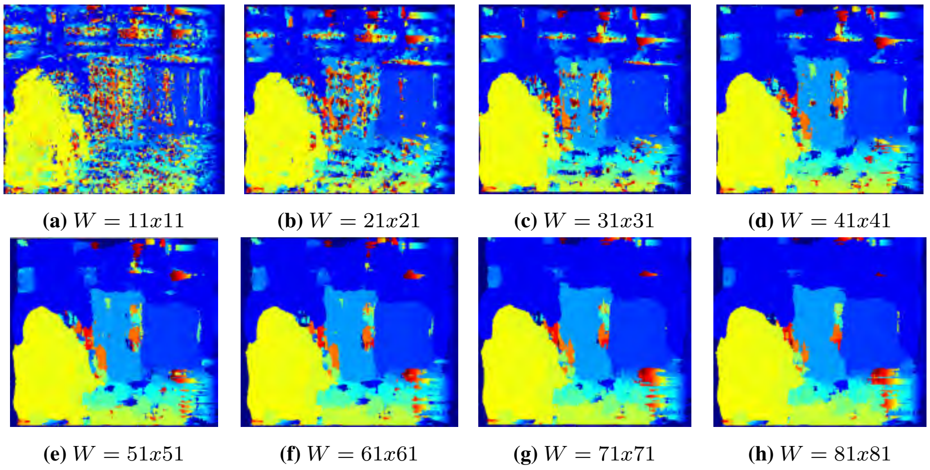

(a) $W = 11x11$ (b) $W = 21x21$ (c) $W = 31x31$ (d) $W = 41x41$ (e) $W = 51x51$ (f) $W = 61x61$ (g) $W = 71x71$ (h) $W = 81x81$

Figure 5.15: Tuning Rank Transform with window 23x23, matching the images with the SAD-algorithm

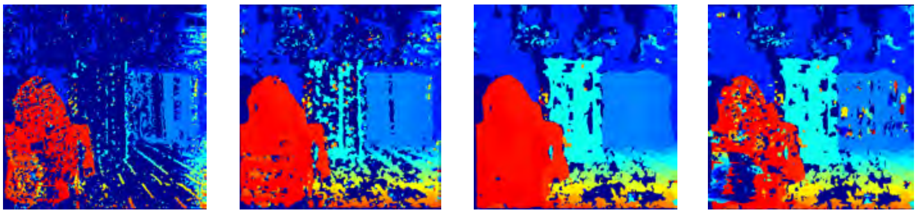
dow as shown in Figure 5.15. The object in-front, together with the one in the back, is adequately segmented, even with smaller matching windows.

It is possible to change the shape of the neighborhood or the matching-window to maybe yield some better results. If using another scene, RT appears to output a disparity map with less noise than SAD. However, considering the wrong depth estimates for the scene used, SAD seems like a more robust choice even if it creates some more noise. Also, in SAD one can use the original images while matching, and thus no need to perform a rank transform in beforehand.

5.2.5 Census Transform

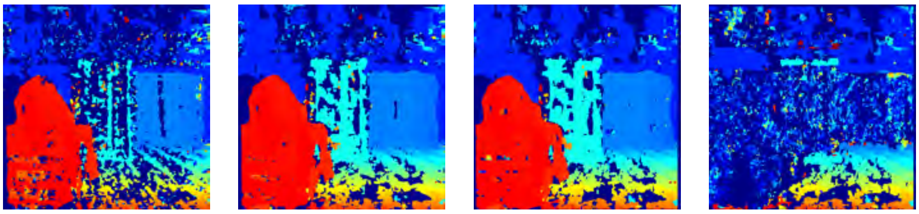
In consideration of the rank transform, CT gave a much more realistic colored depth map. The census transform was first applied with various sized neighborhoods. Subsequently, the two images were matched with the XOR operator with separate matching windows. The best results were achieved by changing the census neighborhood. In Figure 5.16, a matching window of size 9x9 is used with various census-neighboring size. Similar, in Figure 5.17, a matching window of 11x11 is used. It is noticeable how defective the disparity map gets when using too big of a census neighborhood. This is important when choosing the static size for the application. A favorable size in one scene may give an inadequate disparity map in another scene.

The algorithm yields a disparity map with less noise than SAD without any pre-processing. Given the algorithm is known to perform under challenging lighting conditions, it would be interesting to compare CT with SAD in a brighter scene outdoor. Overall, it is an interesting algorithm that could be a good choice, with the potential for real-time applications.



(a) Neighborhood sized to 5x5 **(b)** Neighborhood sized to 21x21 **(c)** Neighborhood sized to 45x45 **(d)** Neighborhood sized to 49x49

Figure 5.16: Different census windows, with block matching window of size 9x9



(a) Neighborhood sized to 17x17 **(b)** Neighborhood sized to 33x33 **(c)** Neighborhood sized to 41x41 **(d)** Neighborhood sized to 45x45

Figure 5.17: Different census windows, with block matching window of size 11x11

5.3 Feature based methods

Feature-based methods output sparse disparity maps. The algorithms first extract interesting keypoints, and subsequently creates descriptors. This is the first approach tested. The second approach sets all the pixels in each image to an interesting keypoint. The algorithms are thus used only to create descriptors, to yield a dense disparity map. The latter does not apply for HCD as the algorithm only outputs a value instead of a descriptive vector. After creating descriptors, they are matched with both FLANN and BF, and the matches used in the resulting disparity map are selected by applying a ratio check test to remove some noise.

5.3.1 Harris Corner Detector

Harris corner detector was first implemented with both FLANN based matcher and BF-matcher in Matlab. The most relevant parameters to tune is the quality or threshold R for the corners detected, and the threshold of how different two corners can be to yield a match.

The challenge with HCD was to find the right thresholds. Because there are no descriptors the algorithm only creates a score for each corner. This score is based on intensity. Thus two images of the same scene with variation in the lighting will yield two different scores on a corresponding corner. This makes it challenging to tune the parameters seeing that the image pairs can have brightness variations.

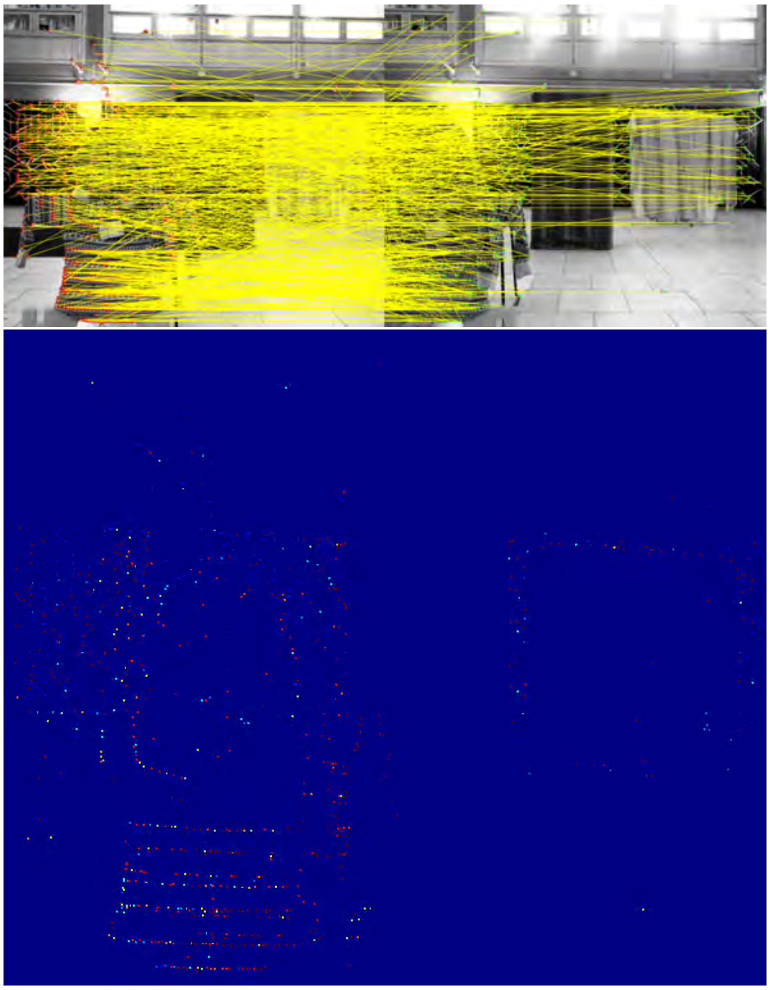


Figure 5.18: Best result using HCD. The matches and the corresponding disparity map

In Figure 5.18, the best result from HCD is shown. The dark blue area is pixels with no detected corners. To be able to see any of the matched points in the disparity map, every match is set to be 9 pixels in size, instead of the initial value of one pixel. The disparity map was achieved by increasing the threshold for a corner to one percent of the maximum value and is further set to match all the detected corners. This removes most of the noise, and it detects between 900-1000 corners in each image. Looking closely at the matches, one can see that most of the corners are matched correct, but there is still some noise. To further reduce noise a solution is to increase the threshold for a corner. But the disparity map is already too sparse to get any reasonable dept-estimates.

In comparison, the algorithm was set to detect every corner, threshold equal to zero, and further to match all the keypoints found. This yields the result in Figure 5.19. It

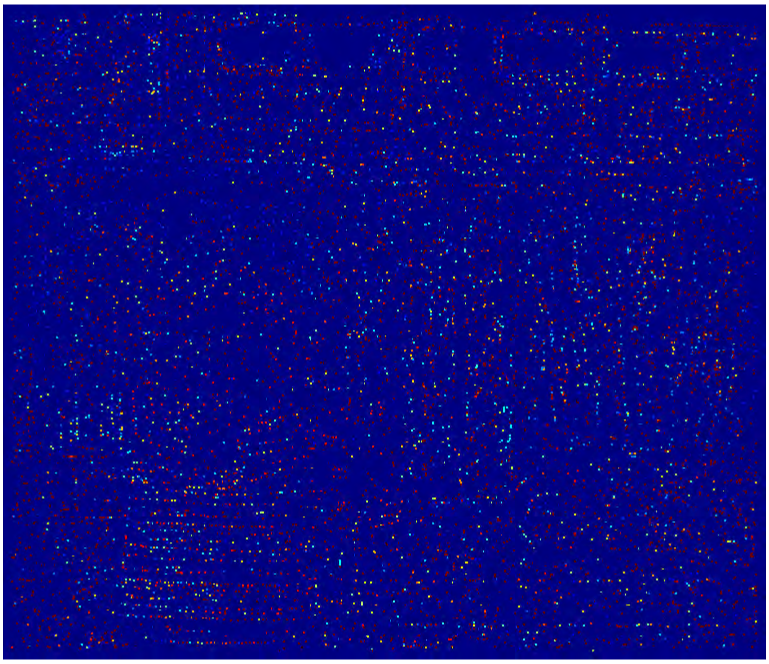
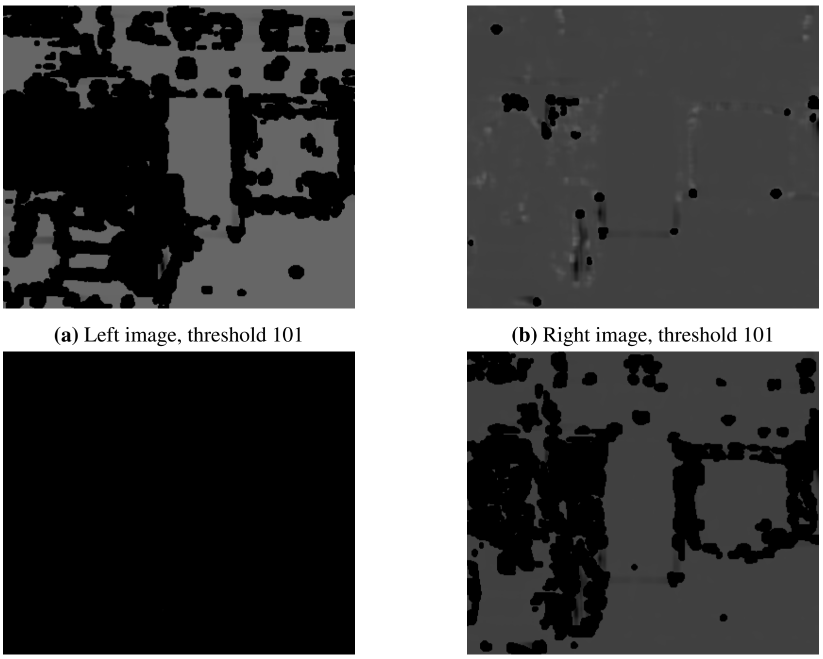


Figure 5.19: Harris Corner Detector, detecting all possible corners and edges

resulted in more points and matches in the disparity map, but way too much noise and still a too sparse disparity map to separate the scene-objects.

One has to find the best correlation between the quality of detected keypoints (corners) and the quality for matching pairs. However, the most troublesome problem of HCD was that it did not detect enough corners. When setting the threshold for being an edge or a corner to zero, the algorithm managed to detect between 9000 and 10 000 keypoints. Considering the image is $1069x1242$ pixels, HCD detects less than 0.008 percent of the pixels, giving a disparity map too sparse to yield good results. This applies independently of the matcher used. In the images displayed, a FLANN based matcher is used, but there were no noticeable changes when using BF.

To explicitly figure out why the algorithm performed the way it did, the corners detected in both the images were plotted with different threshold values R. The result seems to be that the quality or the value for the same corresponding corner has a considerable variation in the two images. A strong corner in the left image often corresponds to a weak corner in the right image. In Figure 5.20, corners detected is tried illustrated. Black areas mean there is detected corners or edges with values above the threshold, while grey means no detected corner. If one could have achieved a result like Figure 5.20a in both the images at the same threshold, one could have used different computer vision techniques, like dilation and blob detection, to filter the disparity map. However, since the threshold has to be determined in advance, it was hard to find any values giving enough edges in both the images to yield a somewhat dense or semi-dense result without too much noise.


(a) Left image, threshold 101 **(b)** Right image, threshold 101

(c) Left image, threshold 64 **(d)** Right iage, threshold 64

Figure 5.20: Detecting cornes with different thresholds. Black means corners, grey means no corners detected

HCD is one of the oldest and simplest algorithms and thus has some obstacles. To improve the resulting disparity map, a kind of normalization of the right and left image in beforehand could be a solution. This may help to reduce the problem of the necessity of a manual threshold selection. Another solution would be to apply another algorithm for creating a more descriptive descriptor than just a value for the corner. An example would be to use SIFT or BRIEF as descriptors.

5.3.2 SIFT

Using SIFT as both keypoint extractor and descriptor type yields the following sparse disparity maps. The input parameters for SIFT were first changed to get the highest number of keypoints as possible. This was done by changing the number of layers, the contrast and edge threshold, and the sigma value. In Figure 5.21, the result is shown. Many keypoints were obtained, but when matching, the result was just noise. The keypoints extracted is thus extracted from different features in the two image-views. The threshold in the parameter values is therefore too low to extract strong enough features.

To further improve the matching, better descriptors were achieved by again changing the parameters. It was hard to balance matching performance and noise. The best result gave 60 000 descriptors in each image, in contrast, to Harris's 1 000 keypoints. In Figure

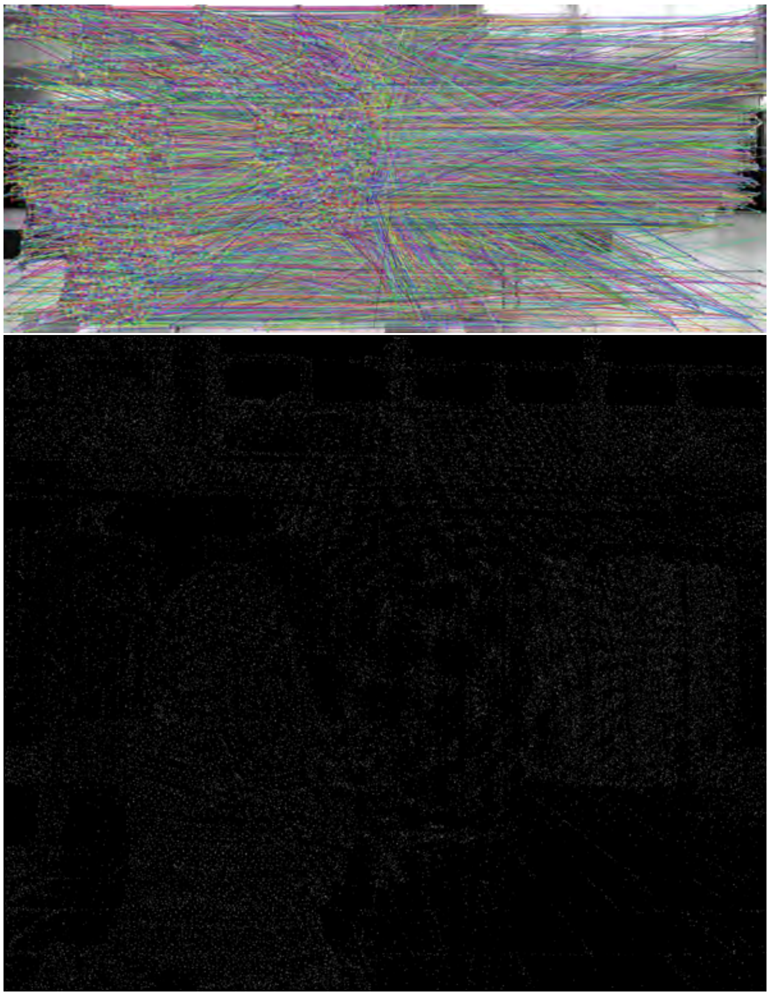


Figure 5.21: SIFT sparse, maximum detected matches achieved

5.22, one can see the best disparity map from using SIFT as both keypoint detector and descriptor. Even though the matched pixels are of reasonable results, the map is sparse, making it hard to attain reliable depth estimates. The algorithm is also not able to detect any keypoint or descriptors around the object in the middle.

The second approach sets all the pixels in each image to an interesting keypoint. SIFT is therefore used to create descriptors for every pixel in the two images. After creating the descriptors, they are all matched with FLANN. In Figure 5.23 the disparity maps are shown with increasing range for the cross-check-test. The ratio in the cross-check-test is a measure of the accepted disparity, both minimum and maximum disparity allowed.

The corresponding matches are drawn in Figure 5.24. One can see that a lot of the matches are not horizontal. This means, there is some noise in the corresponding disparity

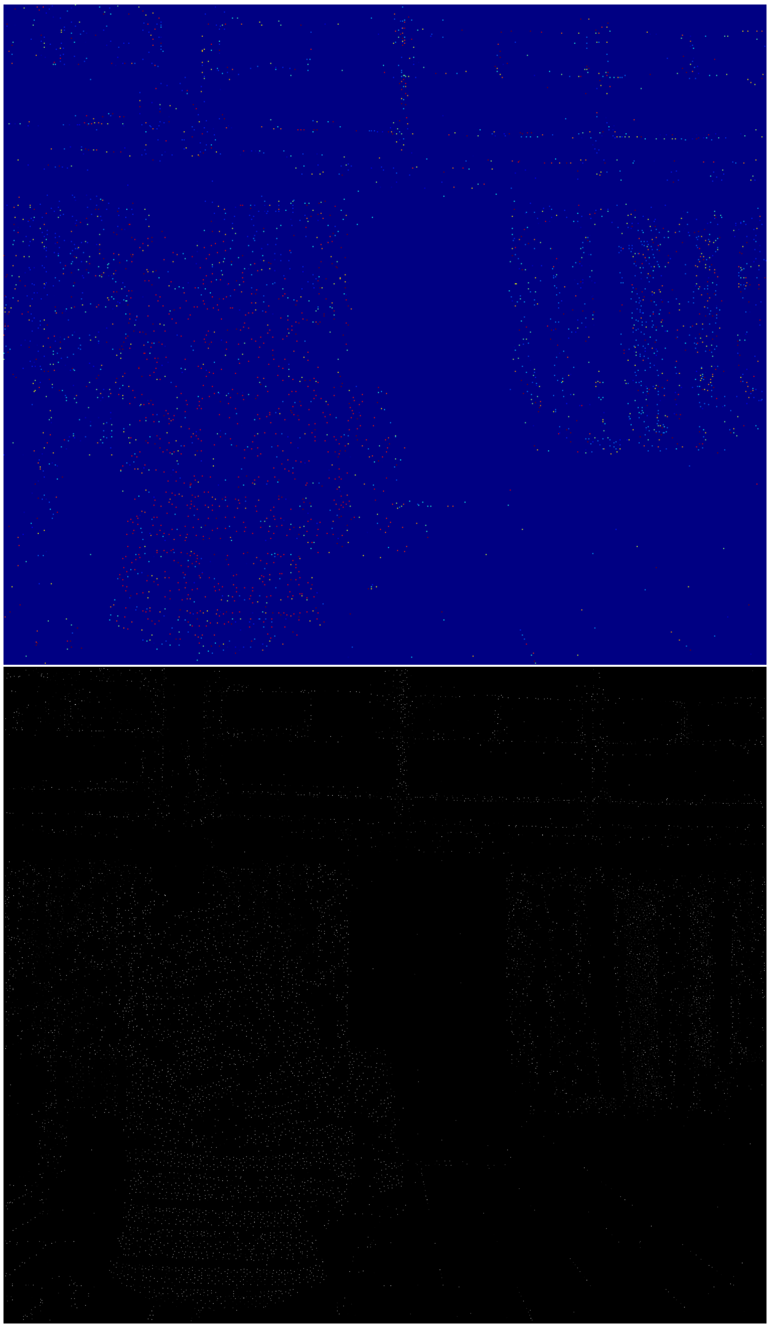
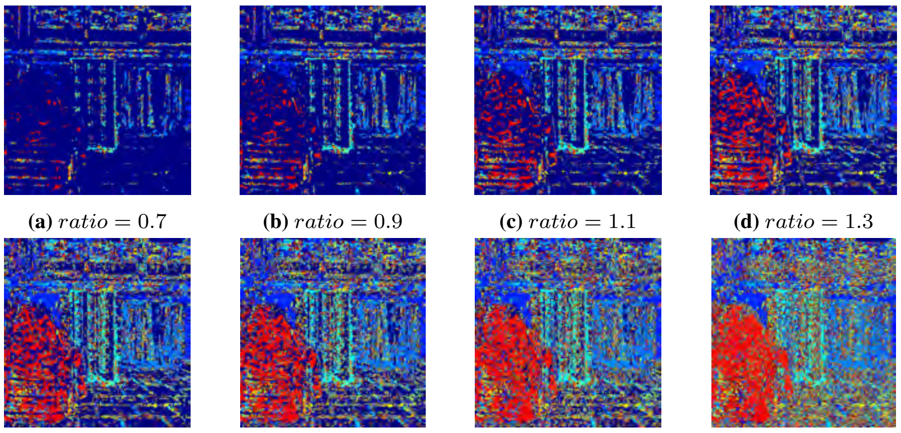


Figure 5.22: SIFT sparse, the best matches plotted in both intensity and color


(a) $ratio = 0.7$ **(b)** $ratio = 0.9$ **(c)** $ratio = 1.1$ **(d)** $ratio = 1.3$

(e) $ratio = 1.5$ **(f)** $ratio = 1.7$ **(g)** $ratio = 2.0$ **(h)** No ratio, every match

Figure 5.23: Tuning the length of disparities, i.e., descriptors to include in the map (FLANN)

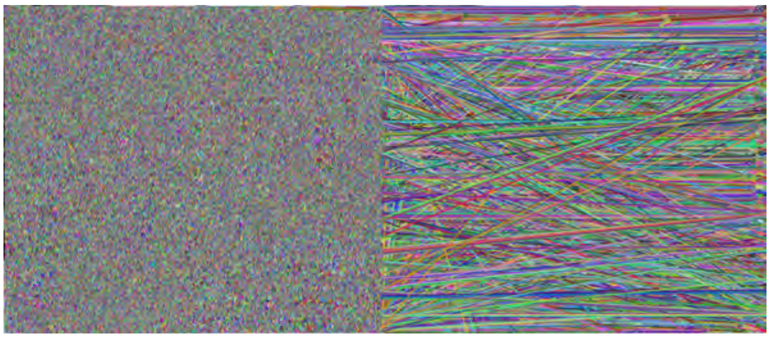
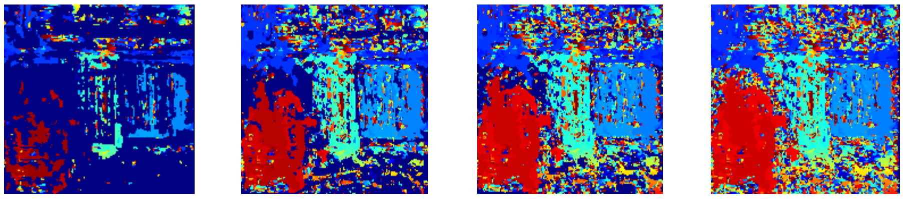


Figure 5.24: SIFT matches

map. SIFT does not take into consideration if the image pair is rectified or not. In theory, there will be no difference in matching two undistorted and rectified images or using the raw format. An idea to improve the disparity map would be to make sure all the matches are in a horizontal line. However, this may be more time-consuming than filtering the disparity map.

To compare the latter result, SIFT was also tested with a BF matcher. The terminal-window displayed the error ”Aborted (core dumped)” when using the BF matcher. The reason is that the memory load is too big while matching every single descriptor with all the other descriptors. The code runs fine when scaling the image by a value of ≤ 0.2, but even so, all the twelve CPUs running were maxed out.

SIFT is scale-invariant, and by scaling the image by the factor 0.2 it should be able to yield similar results. In Figure 5.25, some of the disparity maps achieved is displayed. BF seems to give a more dense disparity map compared to FLANN. Nevertheless, BF is tested on down-scaled images, hence fewer pixels to match, and most likely, less noise.



(a) $ratio = 2$ **(b)** $ratio = 37$ **(c)** $ratio = 4.9$ **(d)** All matches

Figure 5.25: Tuning the length of disparities, descriptors to include the SIFT algorithm (BF)

Comparing with HCD, SIFT managed to make descriptors of $\frac{1}{5}$ of all the keypoints in the scaled-down image. This makes the disparity map sufficiently dense to yield good depth-estimates. Even though BF was used on a scaled image, it seems to give the best results considering depth estimates. Some filtering is still preferable to remove some of the noise applied and to easier segment out the objects. In general, SIFT seems to yield satisfying results, even when scaling the image. Scaling the image will improve the computation time. However, it should be tested to acquire depth on minor objects in scaled-down images to find the best scaling value.

5.3.3 SURF

By using SURF as both keypoint extractor and descriptor type, it yields similar results as SIFT. Changing the hessian threshold gave the most varying results, but also tuning the number of octaves and octave-layers helped. The best values for SURF is shown in Figure 5.26. SURF struggles to find enough strong keypoints. By decreasing the threshold further, the new detected points are matched wrong and hence give rise to a noisy disparity map. Identical results outputted from both BF and FLANN.

For the dense case, SURF was not able to make descriptors for most of the keypoints giving a sparse disparity map with much noise. The two matcher's gave almost identical results. But again, while using the BF matcher, the images had to be scaled by a factor of 0.2. In Figure 5.27, the best result was achieved with a ratio of 0.03. One can see the structures of the scene. However, the matching gave rise to much noise. No reasonable depth estimates were produced.

Seeing the noisy results and sparse disparity maps, it seems that using a SURF size of only 64 bits is too small of a descriptor. The descriptors do not include enough information to match the keypoints in the challenging scene.

5.3.4 ORB

ORB uses binary descriptors of 32 bits. Hence the descriptors are half the size compared to the ones in SURF. Using ORB as both keypoint extractor and descriptor type yields the following sparse disparity maps in Figure 5.28. There are more keypoints and less noise than with both SIFT and SURF. The middle object contains no detected keypoints, but the

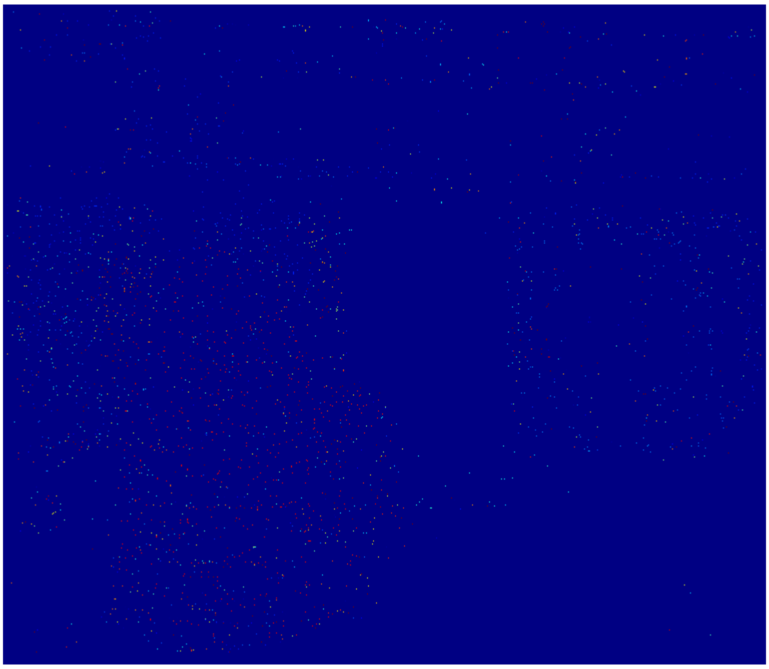
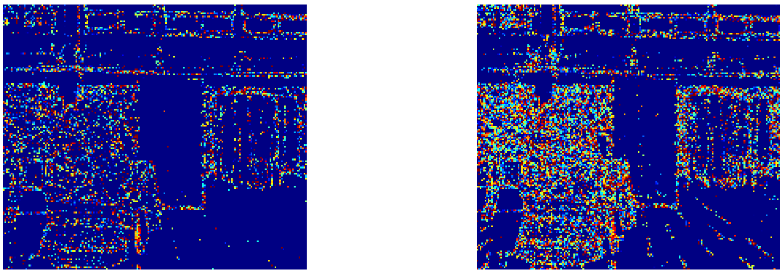


Figure 5.26: SURF sparse



(a) ratio=0.03

(b) All descriptors matched)

Figure 5.27: Tuning the length of descriptors to include in disparity map in the SIFT. The original rectified image is scaled by 0.2

other two objects may include enough to be segmented out. It needs some work, though, and makes it hard to be used in a system detecting various scenes.

The second approach testes ORB with setting every pixel as a keypoint, creating the ORB-descriptor, and further match with the Hamming distance. The more matches used in the disparity map, it yields a more dense disparity map with clearly segmented objects. In Figure 5.29, the various maps created by FLANN using an increasing range of the matches is presented. Compared to the other featured-based descriptors, ORB constructs less wrong-matches. As shown in Figure 5.30, most of the matches are along the horizontal

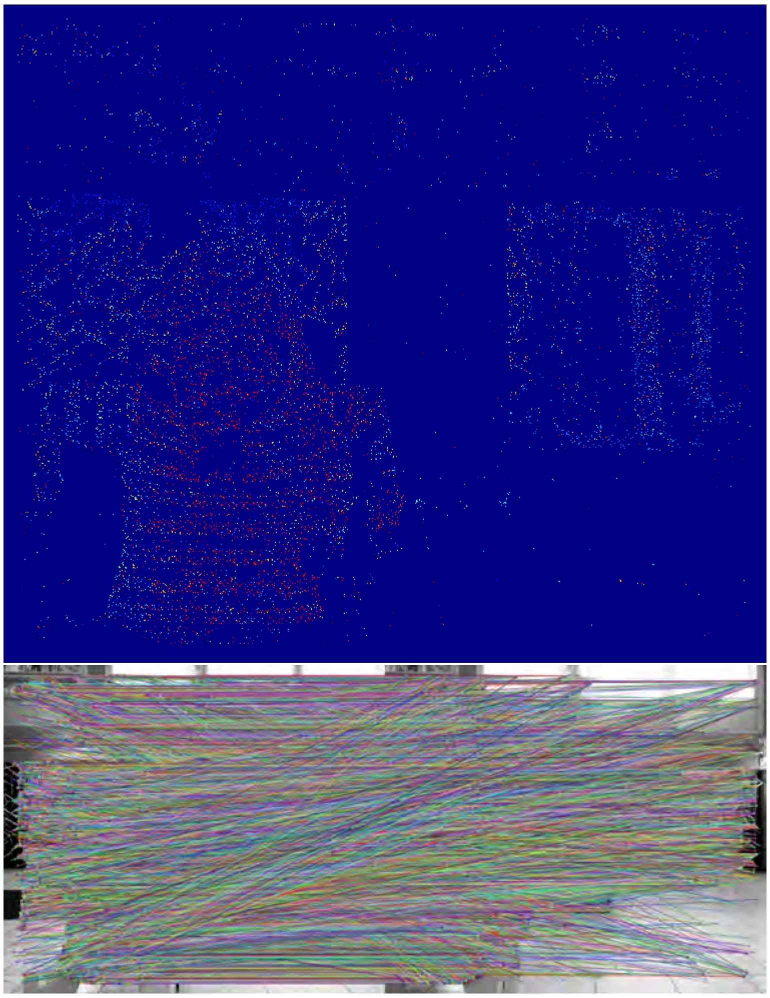
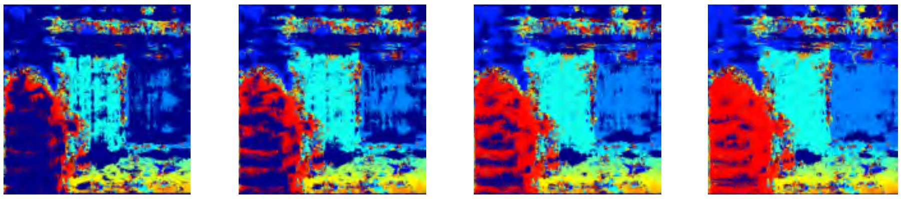


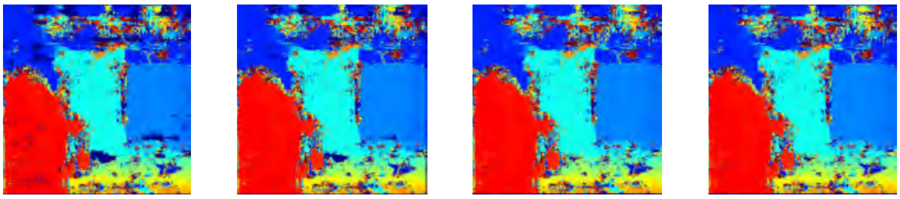
Figure 5.28: ORB, sparse matches

line creating clearly displayed objects. Still, there are some wrong matches, noise, but a significantly less amount than with SIFT and SURF. By segmenting out the background, most of the noise applied will be removed, and hence give an adequate map to query the depth.

As with the two latter descriptors, the BF matcher is not able to perform on descriptors from the original image. Hence, the images are scaled by a factor of 0.2. With the BF matcher, the output gives an odd disparity map. The result is shown in Figure 5.31. The colors are offset by value, meaning the descriptors are not matched correctly. Compared to SIFT, the BF matcher on ORB gave an unacceptable map. The downscaled image was further tested with FLANN matcher to understand the reason for the unexpected result. As seen in Figure 5.32, FLANN yields almost identical results when applied to the images



(a) $ratio = 0.7$ **(b)** $ratio = 1.0$ **(c)** $ratio = 1.2$ **(d)** $ratio = 1.4$



(e) $ratio = 1.6$ **(f)** $ratio = 1.8$ **(g)** $ratio = 2.0$ **(h)** $All feature matches$

Figure 5.29: Tuning the length of descriptors to include in disparity map in the ORB algorithm (FLANN)

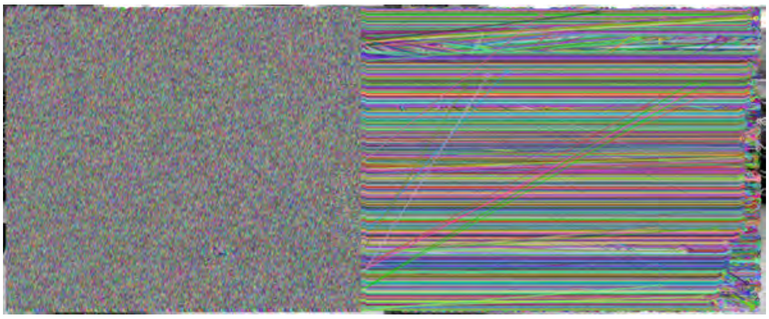
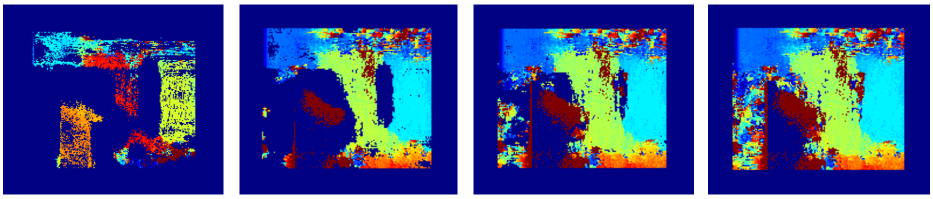


Figure 5.30: ORB matches



(a) $ratio = 0.0.5$ **(b)** $ratio = 1.0$ **(c)** $ratio = 1.5$ **(d)** All matches

Figure 5.31: Tuning the length of descriptors to include in disparity map in the ORB algorithm (BF) The original rectified image is scaled by 0.2

scaled by a factor of 0.2.

This means that ORB certainly has a threshold of how low resolution of the image can be to provide valuable descriptors. The most noticeable is that the matched keypoints are

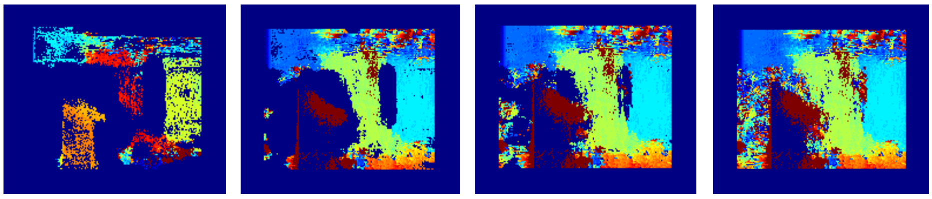

(a) $ratio = 0.0.5$ **(b)** $ratio = 1.0$ **(c)** $ratio = 1.5$ **(d)** All matches

Figure 5.32: Tuning the length of descriptors to include in disparity map in the ORB algorithm (BF) The original rectified image is scaled by 0.2

almost all on a horizontal line, displayed in Figure 5.33. So it indeed has potential when scaling down the image. On the other hand, a scaled-down image will have more trouble detecting smaller objects or objects on longer distances. The colors may be a bit offset due

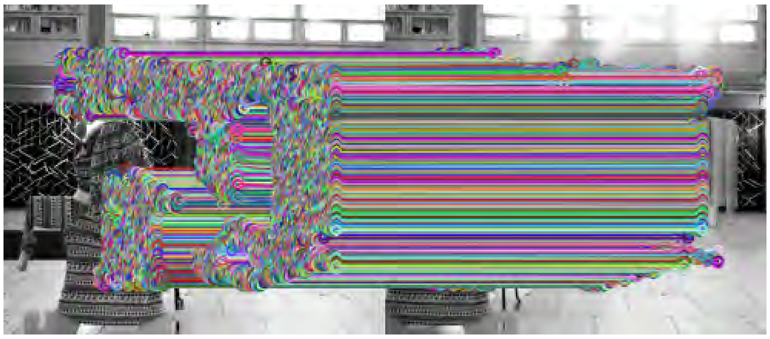


Figure 5.33: Flann scaled matches

to the scaling. Nevertheless, it did not seem to give such a result with SIFT. Considering the running time, FLANN uses two-thirds of the time compared to BF.

In general, ORB is a promising descriptor for achieving disparity maps. Preferably a filter is to be applied in the post-processing, but even without this it gives reasonable results.

Chapter 6

Discussion

The resulting disparity maps and the dept-estimate achieved are all results dependent on the stereo setup, the calibration, and the chosen algorithm. The best depth-error achieved was 3 cm. The error is an overall exceptional satisfying result, considering the report was written with no pre-knowledge of stereo setup and calibration. However, there is always potential for improvement, and this chapter gives a discussion of the different results achieved. Disclaimer; the running time is mentioned in the algorithms with the most potential, but when writing code, no effort was made to optimize the computation cost. The code used focus on understanding, and is written for testing purposes. The code itself can be optimized, and most likely, there exist faster libraries and functions.

6.1 Stereo Setup

The errors achieved is dependent on the stereo setup. The setup is the foundation for the application and thus will influence the results. It is hard to say how much the setup affected the correspondence problem because the algorithms were only tested with one setup using a fixation point of 4 meters. However, the stereo setup used gave some reasonable disparity maps, meaning the algorithms managed to solve the correspondence problem. Therefore, it seems that the angling of the cameras helped to minimize the depth-error. This in mind, it is reasonable to expect a more significant baseline and increased angling, with the same fixation point, will decrease the error. The baseline was set to be 26 centimeters, and widening it will increase the theoretically precision of depth. However, widening the baseline and angling the cameras will also make the search problem more challenging. A compromise is needed between the hardness of the search problem and the theoretically precision. Hence, testing the scene with the same algorithms using a broader baseline and increasing the angling will give a further understanding of how the setup affects the dept-estimates.

Overall, the chosen setup was acceptable and conceived consistent results. It gave disparity maps, and depth estimates acceptable for understanding how a 3D world map differs with various algorithms.

6.2 Calibration

Calibrating the parameters for the cameras was a vital part of this project to be able to reconstruct the scene by creating disparity maps. Dividing the calibration into two separate parts seems to give the most valuable parameters when calibrating on longer distances. The intrinsic parameters calibrated on the preferred distance, resulting in accurate parameters. The extrinsic parameters were calibrated on the operating distance, 4 meters, and thus needed 275 image pairs to cover the image plane. Using this amount of images is time-consuming, and not recommended as it increases the likelihood of errors occuring.

A reason for wrongly matched pixels in the correlation-based methods is the calibration parameters. However, the rectification and distortion parameters seem acceptable as most of the algorithms found suitable matches on the horizontal line. The calibration is most likely the origin of some of the noise in the correlation-based disparity maps. Even though some noise is to be expected, it can be reduced with a flawless calibration. Overall, the calibration yield better results than expected, given the narrow calibration board used.

Improving the calibration should be attempted. First, setting the physical lens parameters more equal, focus and iris, to easier compare the intrinsic parameters. This will equalize the intensity of illumination being fed into the lens, and therefore making the intensity images more balanced. Nevertheless, there will always be some illumination variations between two pictures captured at different positions. Additionally, a substantial greater calibration board is preferred to lessen the number of images needed in the extrinsic calibration. Instead of using an enormous checkerboard, a static object with known measures is most likely a better and more realistic alternative.

Kalibr seems to be a reliable alternative to the MATLAB's calibration apps, which should be tested, seeing if it achieve the same results.

6.3 Stereo matching algorithms

Knowing something about the scene is the key to yield the best depth estimates. The knowledge is achieved after testing the algorithms with a bunch of parameter values in Chapter 5. The searching range in the correlation-based methods was predefined, and with the feature-based methods, the matches were filtered out with a cross-check-test. This yields a smoother result, removing some of the noise. A predefined scene also makes it easier to apply filters segmenting out the objects. In general, use everything known in beforehand when choosing the parameters and filters.

Considering the running time, the size of the image is of considerable importance. Every pixel in the two images is used for matching in both the stereo matching methods. The image used contains 1 327 698 pixels, giving rise to a time-consuming matching. To shrink the number of pixels, one could easily crop some parts of the images. In general, there is no need to match every pixel. Like in the scene used, one could easily have cropped a large portion of the pixels at the top of the image. An alternative approach would be to match or create descriptors for every second pixel. This reduces the running time and creates a semi-dense disparity map. The semi-dense map can effortlessly be made dense by applying smoothing, or for example, using simple clustering algorithms like KNN or DBSCAN. A similar approach would be to down-scale the image. Nevertheless, both the ese proposals may struggle while used in the detection of objects depicted with fewer pixels.

Insight into some pre-information about the scene to be detected, the third option is to down-scale only parts of the scene. By smoothing and reducing the resolution on only near distances, one can achieve better matching for several objects with different sizes. Further, it is also possible to up-scale the size of objects captured at longer distances. This ensures that one matching window will yield better results on several objects captured on varying distances and with disparate sizes. Similarly, it lets the featured based methods to detect objects faster using fewer layers and octaves. Information about the scene ahead of the stereo matching could, for example, be given through sensor fusion with a RADAR.

In general, it would be interesting to test the algorithms at longer distances with additional objects. However, since the camera setup was used in another project at the same time, there was not enough time. Nevertheless, the checkerboard board used would not have achieved an adequate calibration on farther distances. Furthermore, the algorithms should be tested on several scenes with several diverse objects, backgrounds, and illumination conditions. Objects on greater distances will eventually, dependent on the setup, vanish in the background. Furthermore, the use of pre-filtering and post-filtering should further be examined with the chosen algorithm.

Following, the algorithms tested are briefly discussed with some comments on how they may be improved.

6.3.1 Correlation based methods

In common for the algorithms, adjusting the windows' size yields the most prominent variation. Therefore, it is necessary to know the size of the objects in beforehand to yield the best results. With this in mind, the most valuable objects to detect and measure distance to has to be predefined. Dependent on the surfaces of the preferred objects, one can estimate a window size expected to yield the best results. Choosing a window too big will remove much of the noise, but also remove the smaller objects. Choosing a window size too narrow will create a lot of noise, removing large neutral objects.

An alternative would be to look into windows with different shapes. A rectangular window, a short and wide one, could be preferable when detecting boats. While writing code, there is possible to use a predefined window-shape of any architecture. There also exist articles on adaptive the window in real-time [34].

Most of the algorithms implemented to search for correspondences to match in both the direction on the horizontal line. Considering the running time, this should be avoided. If the calibration is satisfactory and the images rectified, there is no need to search in both the directions making the burden double. Additionally, it can also apply extra noise.

The correlation-based method creates dense disparity maps. The resulting disparity maps should be further refined to yield an more accurate map for estimating the depth. For the dense case, there exists an excessive amount of filters to be applied, both before creating the disparity map and post-filtering.

Sum of Absolute Differences

SAD gave one of the best results overall. It is maybe the fastest matcher, easiest to understand and implement, while also yielding satisfactory results. There is much noise, but one can easily spot the objects in the resulting disparity map, regardless of the chosen window-size. By applying a WLS-filter, the noise was shown to be easily removable, creating a smooth disparity map to achieve the depth.

For the semi-optimized code used to create the disparity map for the point cloud, the running time for the matching part only is measured by *GetTickCount()* to be average 0.0071 seconds. The filtering used 0.025 seconds. The filter may be replaced with a better one considering the running time. The total running time measured included command line parser and creating four disparity maps (no imshow), yields 0.050 seconds.The computational cost can be minimized by removing print sentences and the extra for-loops for testing-purposes. However, the algorithm is feasible in real-time applications, of course, depending on the real-time requirements for the system.

A problem with SAD, as with the other correlation-based algorithms, is detecting smaller objects. Smaller objects require smaller window-sizes, which give rise to noise. Noisy disparity maps are harder to filter, and the smaller objects tend to disappear throughout the noise or in the filtering. A solution is to increase the resolution around objects on farther distances or similar approaches, as discussed above. As well, the algorithm should be tested in variate illumination conditions. The algorithm may struggle with brighter or darker scenes, creating an excessive amount of noise.

Overall, SAD is one of the methods with the most articles and code online. It is widely used, and there exists optimized code for use in real-time applications. However, most of the codes used in real-time are applicable with images of sizes up to 400x400 pixels. The image pair should, therefore, be cropped or smoothed to decrease the running time. A larger window also increases the computation time.

Sum of Squared Differences

SSD seems to yield identical results as SAD, maybe some more noise, but nothing noticeable in the images presented. In general, SAD and SSD achieve similar results, but SAD is preferred. SAD is computationally faster and less sensitive to outliers. Hence, in this report, SAD is preferred before SSD. SSD can yield better results in some applications due to it being differentiable, e.g., in optical flow.

Normalized Cross Correlation

NCC was first tested on the full-scale image. The algorithm was interrupted after one hour, as it did not manage to complete the computation. Code with less computation cost was found, but still, it did not give any acceptable results considering the resulting disparity map and the computational time.

NCC may be favored in a brighter and noisier scene. As stated in the theory chapter, it is notorious for high complexity and thereby rarely meets speed requirements in real-time systems.

Rank Transform

RT, as expected, yields proper segmentation of the objects. It is known to improve performance near object boundaries, but again, no noticeable improvements occurred compared to the tree latter methods. The distance measures are all wrong, and the only logical explanation is that binary images are harder to match for the simple SAD algorithm. Mismatches are more likely to happen, resulting in wrong depth estimates.

It could be tested in another scene to see if it yields the same unacceptable result. Overall, it seems the algorithm has more drawbacks than advantages compared to using only SAD.

Census Transform

CT provides reasonable results, with less noise than SAD. Looking at the disparity maps from both filtered SAD and CT, it looks to be in the same range. Calculating the average depth on the three objects from CT's disparity map, happened to give the exact same results as with SAD. Namely an overall average error on 3 cm. Two algorithms producing the same depth-estimates suggest that it may be the calibration or the stereo setup, causing the errors. However, considering the running time, SAD outperforms CT. In total, CT takes around 0.9-1 seconds to run. The code used has some potential to minimize the computational cost, but nothing close to SAD.

The algorithm is frequently used in local stereo vision due its performance under challenging lighting conditions. It also seems to be robust against noise. Regardless, there are several articles on improving the CT to be more robust against noise in stereo matching, e.g., Lee et al. propose a star-census transform [19]. It can be improved either by using filters or an improved version like the star-census transform. The binary matcher yields one of the best disparity maps and should be further examined in various scenes. It is conceivable that CT will be more robust than SAD considering various illumination conditions. Therefore, CT is considered to be the algorithm with the most potential using in a real-world scenario, implemented on the autonomous ferry prototype ”MilliAmpere”. However, the algorithm generates noise when the census neighborhood is too large. Thus one has to be tentative when choosing the aggregation window.

6.3.2 Feature based methods

Feature-based methods are generally more robust against illumination conditions, environmental conditions, and change of contrast. Traditionally, the correlation-based methods have been preferred in stereo vision due to computation cost when using rectified images. However, from the algorithms tested, there is potential for the feature-based methods. They may, traditionally, be more time consuming because of the extra step for extracting features. Nevertheless, their computational cost is decreasing with more ongoing research.

The main restriction of feature-based methods is that they do not allow to generate dense disparity maps. However, in this report, it was resolved by setting all pixels to an interesting keypoint. As a result of the approach, a simpler algorithm may have been preferred. An alternative method can be to use the BRIEF descriptor directly with rectified images. Using rectified images and smaller angles on the cameras, will most likely not need any rotation-invariant descriptors. Thus, a BRIEF descriptor seems to be a good alternative, minimizing the computational cost. Similarly, one can set the feature-based methods not to extract rotation; this is not tested throughout this project.

All of the images were captured using grayscale because the algorithms chosen are intensity-based. Grayscale matching also reduces complexity. But there exist descriptors operating on colored images like Opponent-SIFT and the Colored SIFT (CSIFT). Especially since the algorithms are going to be implemented outside in a marine environment, one may predict some of the expected colors, and thus the colors can be used to improve the contrasts. An example is Zhao et a. [50] which showed how image matching could be improved by using color and exposure information with SIFT. Another example is van de Sande et al., concluding that object recognition from the use of color benefit 30 percent in performance improvement over using intensity only [46].

The feature-based methods tested, except HCD, is said to be invariant to rotation and scale. Hence, in theory, there is no need to use rectified images or the total image in size. A fresh down-scaled image should be tested to reduce computation time.

Harris Corner Detector

HCD struggled to detect corners to match in the images. Mainly, it struggles to detect any corner in the middle of the objects or on the floor, meaning it needs sharp corners to be able to detect and match them. HCD will, like the correlation-based methods, be dependent on the size of the image, and also smaller objects are complicated to detect and match. Overall, the descriptor seems to be the biggest problem as it only selects a value for each corner. A value was shown to be hard to match when the images have some illumination variations. Normalization of the right and left images can be applied to improve matching and the resulting disparity map. Nevertheless, it is not robust enough to use the same threshold values for various scenes, seeing that a more robust descriptor should be used. A classic combination is a Harris detector + SIFT descriptor, but since the algorithm detected few interesting keypoints, it seems not to be the best approach in this report.

SIFT

SIFT gave noisy disparity maps, but nothing that cannot be fixed with applying filters. Chiefly SIFT achieves acceptable results considering the colors in the disparity maps. Setting all pixels in the original image to an interesting keypoint and following creating descriptors takes 0.92 seconds for SIFT. For a down-scaled image, it takes 0.035 seconds. The total run-time with the down-scaled image using the BF matcher is 28 seconds compared to FLANN using 0.97 seconds. On the original sized image-pair, the total run-time with FLANN is 42 seconds. There are some improvements than can be applied to minimize the computational cost, but so far, SIFT with BF is far from real-time applicable. SIFT with FLANN has potential.

The disparity map was less noisy with down-scaled images using BF. However, it may be that the method would have removed smaller objects if present in the scene used. Calculating the depth in the images matched with BF yield some small errors. This is because the image-pair was first multiplied with 0.2, and the resulting disparity map multiplied with 5 to get the real-world disparity.

Another alternative to SIFT would be DAISY [45]. The DAISY feature descriptor has been shown to outperform SIFT and SURF in wide-baseline stereo matching and is more computationally efficient than SIFT [32].

SURF

SURF yields noisy results, which was unexpected. In articles and projects online, SURF seems to perform results similar to SIFT but faster. With the scene used, no parameters were found to yield satisfactory disparity maps. Reading the specification, about the restrictions, and further tuning the parameters several times, there was still not achieved any satisfactory results. The result, when setting all pixels to interesting keypoints, gave a sparse set of descriptors. Testing SURF with a descriptor of size 128 may yield an improved disparity map. Nevertheless, other scenes should be tested to see if it yields the same noisy and sparse results. If this is the case, SURF features may not be robust enough in certain scenes. Improvements could be to combine different algorithms. Li, Aomei, et al. propose a fast matching algorithm based on the combination of FAST feature points and SURF descriptors, resulting in an improved matching compared to SIFT [20].

ORB

ORB yields one of the best disparity maps without applying any filtering. Together with CT's result, it seems like binary descriptors provide less noisy disparity maps. Even when using less than half of the descriptors created, ORB with the FLANN matcher manages to create an acceptable disparity map. There is less noise compared to the other algorithms testes, maybe except CT, and depth estimates are usable without any filtering. The average depth provided on the object is identical to SAD and CT.

A point of interest is the unexpected result when scaling the image by a factor of 0.2. This yields an inaccurate disparity map, implying that ORB has a threshold for the images to be matched. Regardless, it is beneficial that the matches seem to be mostly on the horizontal line. The ORB descriptor is robust and easy to match in the scene used. However, ORB, as well as the other algorithms, should be tested in various scenes to consider the possibility that the scene used is a preferred scene for ORB.

Considering the running time, there is much potential for improvement. ORB with the FLANN matcher on the full-scaled image-pair used 20 minutes (setting all keypoints to be made descriptor of). To create the descriptors, ORB uses 2 seconds. ORB is proven to be an efficient alternative to SIFT, so the slow running time was an unexpected result. Nevertheless, there is potential for optimization, better parameters and code may exist.

Descriptor matching

For the two feature matchers, the result was as stated in theory. FLANN was the best and fastest matcher when using images with many descriptors, and BF yields the best result when the images had fewer descriptors to match. The most noticeable was that BF needed the image pair to be scaled with a factor of 0.2. The maximum number of features it managed to match was around 200 000 (but it yields a running time of 8 minutes). Overall, FLANN gave satisfactory results. It computed much faster compared to BF and hence should be preferred in real-time systems with a large number of descriptors to match.

Chapter 7

Conclusion and Future work

In total, the algorithms which were stated to yield the best results managed to calculate the best depth-estimates; SAD, CT, and ORB all three resulted in an average error on 3 centimeters. An insignificant small error, much better than what was thought possible in advance.

This thesis gives an understanding of the stereo setup, the calibration, and the algorithms. In complete, some algorithms have been outsourced; SSD, NCC, RT, HCD, and SURF. Ending up with some algorithms with potential; SAD, CT, SIFT, and ORB. Overall, CT seems to yield the best result considering accuracy, computation cost, and robustness against variations in illumination. Moreover, new additions that may be beneficial to look into were discussed; DAISY, BRIEF, Opponent-SIFT, and CSIFT.

By using a bigger calibration board and testing with distinct stereo setups, even better knowledge is to be achieved. Additionally, several scenes are to be preferred tested, and with optimizing the algorithms considering run-time and filters, stereo vision indeed has potential on longer distances.

The average depth-error was only 3 centimeters, and the system achieved seems to have much potential on longer distances. However, the range of topics covered gives several subjects to look more into. To be able to implement this on the autonomous all-electric passenger ferry, several improvements need to be conducted.

- Find calibration objects which are suitable for calibrating at longer distances, replacing the common checkerboard
- Test how much the cameras can be angled before the correspondence problem is intractable with stereo matching
- Test in scenes with varying illumination conditions and different sized objects.
- What is the best solution to reduce the resolution of the images without removing the smaller objects
- Figure out the threshold for down-scaling images with feature based methods

- Look into descriptors using color images.
- Test DAISY and BRIEF for extracting descriptors
- Test histogram-based metrics instead of Euclidean
- Test the FLANN descriptor matcher with other index methods and parameters
- RANSAC can be used to remove outliers in the disparity maps
- Filters. What kind of filtering yields the best result.
- The best computation cost to be achieved for the different algorithms
- In which scene do the algorithm struggle the most. The robustness of the various algorithms

Bibliography

[1] B. AG. Gigabit Ethernet and GigE Vision. `http://www.shortcourses.com/stereo/stereo3-14.html`, not announced.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *European Conference on Computer Vision*, 2006.

[3] A. Buades and G. Facciolo. Reliable multiscale and multiwindow stereo matching. *SIAM Journal on Imaging Sciences*, 8:888–915, 2015.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. *European Conference on Computer Vision (ECCV), Heraklion, Crete*, 2010.

[5] D. P. Curtin. Understanding the Baseline. `http://www.shortcourses.com/stereo/stereo3-14.html`, 2011.

[6] FLIR. *Configuring Synchronized Capture with Multiple Cameras*, 2017.

[7] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[8] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic time. *ACM Trans. Math. Software*, 3:209–226, 1976.

[9] C. G. and M. Stephens. A combined corner and edge detector. *Alvey vision conference*, 15:10–5244, 1988.

[10] T. V. Haavardsholm. A handbook in Visual SLAM. Compendium in TTK21 Introduction to Visual Simultaneous Localization and Mapping - VSLAM at NTNU, 2019.

[11] D. Hafner, O. Demetz, and J. Weickert. Why is the census transform good for robust optic flow computation? *Scale Space and Variational Methods in Computer Vision. SSVM 2013. Lecture Notes in Computer Science*, 7893, 2013.

[12] R. A. Hamzah and H. Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016, 2016.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision, 2nd edition*. Cambridge University Press, 2004.

[14] J. Jiao, R. Wang, W. Wang, S. Dong, Z. Wang, , and W. Gao. Local stereo matching with improved matching cost and disparity refinement. *IEEE MultiMedia*, 21(4):16 – 27, 2014.

[15] J. Joglekar and S. Gedam. Area based stereo image matching technique using hausdorff distance and texture analysis. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-3/W22:109–114, 2013.

[16] J. Joglekar and S. S. Gedam. Area based image matching methods – a survey. *International Journal of Emerging Technology and Advanced Engineering*, 2(1):130–136, 2012.

[17] K. Jyothi. A robust and efficient pre processing techniques for stereo images. *International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 89–92, 2017.

[18] D. Kumaria and K. Kaur. A survey on stereo matching techniques for 3D vision in image processing. *IJEM*, 4:40–49, 2016.

[19] J. Lee, D. Jun, C. Eem, and H. Hong. Improved census transform for noise robust stereo matching. *Optical Engineering*, 55(6), 2016.

[20] A. Lia, W. Jianga, W. Yuana, D. Daia, S. Zhanga, and Z. Wei. An improved fast+surf fast matching algorithm. *Procedia Computer Science*, 107:306–312, 2017.

[21] A. Liszewski. Elon Musk Was Right: Cheap Cameras Could Replace Lidar on Self-Driving Cars, Researchers Find. `https://gizmodo.com/elon-musk-was-right-cheap-cameras-could-replace-lidar-1834266742`, 2019.

[22] Y. Liu and J. Aggarwal. *Local and Global Stereo Methods*, pages 297–308. 12 2005.

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[24] N. A. Manap, S. F. Hussin, A. M. Darsono, and M. M. Ibrahim. Performance analysis on stereo matching algorithms based on local and global methods for 3d images application. *JTEC*, 10:23–28, 2018.

[25] K. McCabe. How to Set Up a Stereo Machine Vision Solution. `https://www.qualitymag.com/articles/93543-how-to-set-up-a-stereo-machine-vision-solution`, 2016.

[26] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do. Fast global image smoothing based on weighted least squares. *Image Processing, IEEE Transactions on*, 23(12):5638–5653, 2014.

[27] B. Mohamad, S. Yaakob, R. Raof, A. Nazren, and M. Nasrudin. Template matching using sum of squared difference and normalized cross correlation. *Journal of Sensors*, pages 100–104, 2015.

[28] A. Mordvintsev and A. K. Revision. Feature Matching. `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html`, 2013.

[29] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP*, 1, 2009.

[30] T. U. of British Columbia. SIFT: Scale Invariant Feature Transform. `https://uilo.ubc.ca/sift-scale-invariant-feature-transform`, 2004.

[31] A. Patel, D. R. Kasat, S. Jain, and V. Thakare. Performance analysis of various feature detector and descriptor for real-time video based face tracking. *International Journal of Computer Applications*, 93, 2014.

[32] X. Peng, A. Bouzerdoum, and S. L. Phung. Efficient cost aggregation for feature-vector-based wide-baseline stereo matching. *EURASIP Journal on Image and Video Processing*, page 24, 2018.

[33] S. Praveen. Efficient depth estimation using sparse stereo-vision with other perception techniques. *Advanced Image and Video Coding*, 2019.

[34] M. Pérez-Patricio and A. Aguilar-González. Fpga implementation of an efficient similarity-basedadaptive window algorithm for real-time stereo matching. *Journal of Real-Time Image Processing*, 16(2):271–287, 2019.

[35] M. Ramakrishna and S. S. S. Is orb efficient over surf for object recognition? *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 3(8), 2014.

[36] R. Ranftl, S. Gehrig, T. Pock, and H. Bischof. Pushing the limits of stereo using variational stereo estimation. *IEEE Intelligent Vehicles Symposium*, pages 401–407, 2012.

[37] E. Rosten and T. Drummond. Machine learning for high speed corner detection. *European Conference on Computer Vision*, 1:430–443, 2006.

[38] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. *ICCV*, 11(1):2, 2011.

[39] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[40] J. M. Røe. Continuous Calibration of 3D Vision Systems. `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/260911`, 2013.

[41] Y. Sadakuni, R. Kusakari, K. Onda, and Y. Kuroda. Construction of highly accurate depth estimation dataset using high density 3d lidar and stereo camera. *2019 IEEE/SICE International Symposium on System Integration (SII)*, 2019.

[42] Y. Shen. Efficient normalized cross correlation calculation method for stereo vision based robot navigation. *Frontiers of Computer Science in China*, 5:227–235, 2011.

[43] B. Templeton. Cameras or Lasers? `https://www.templetons.com/brad/robocars/cameras-lasers.html`, 2013.

[44] R. H. Thaher and Z. K. Hussein. Stereo vision distance estimation employing sad with canny edge detector. *International Journal of Computer Applications*, 107(3), 2014.

[45] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Trans. Pattern Anal*, 32:815–830, 2010.

[46] T. G. van de Sande, Koen EA and C. G. Snoek. Color descriptors for object category recognition. *Conference on Colour in Graphics, Imaging, and Vision*, 2008, 2008.

[47] Y. Wang, W. L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[48] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. *Lecture Notes in Computer Science*, 801, 2005.

[49] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22:1330 – 1334, 2000.

[50] Y. Zhao, Y. Zhai, E. Dubois, and S. Wang. Image matching algorithm based on sift using color and exposure information. *Journal of Systems Engineering and Electronics*, 27, 2016.