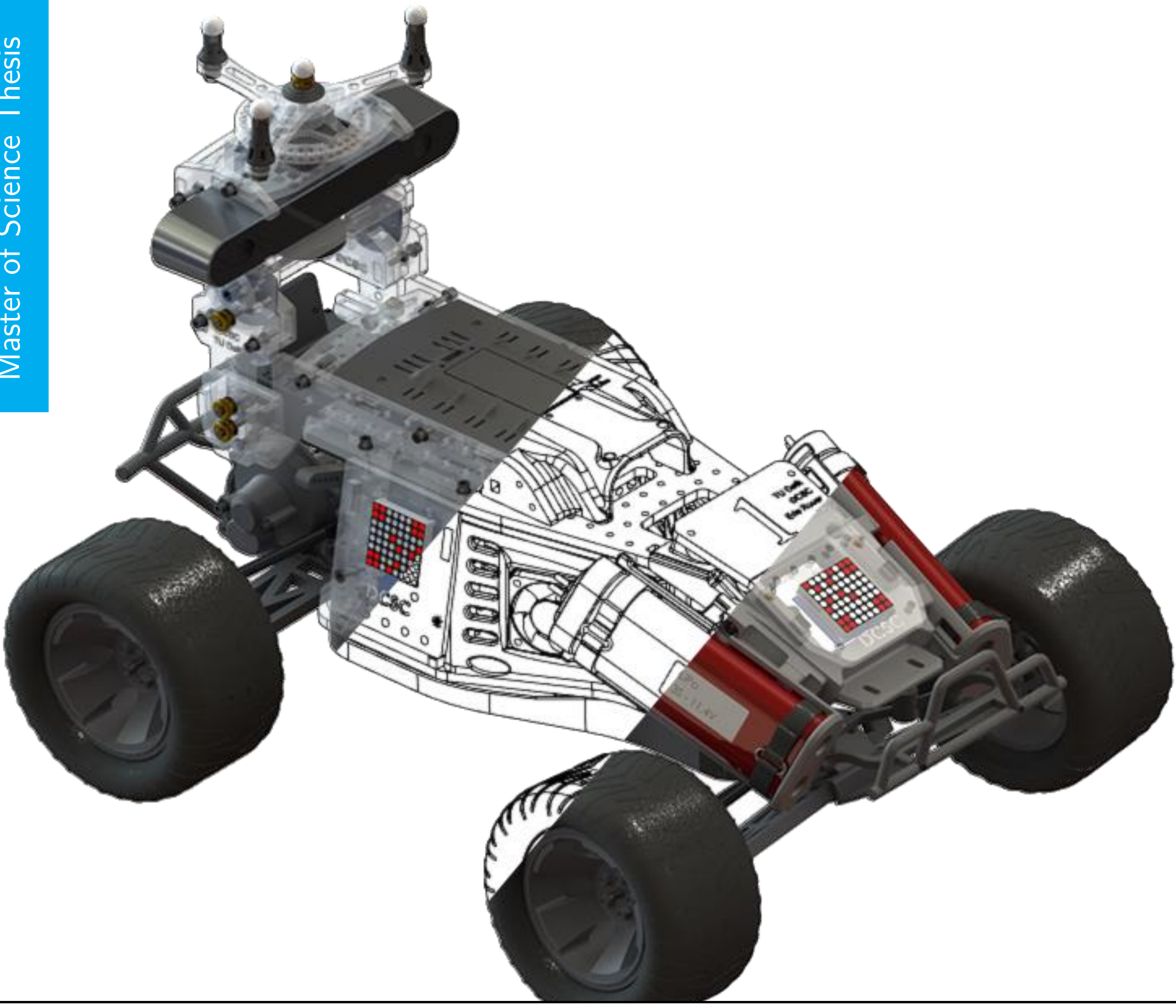
Development of a Platform for Stereo Visual Odometry based Platooning

S.J. van der Marel

Master of Science Thesis



Delft University of Technology

Delft Center for Systems and Control

Development of a Platform for Stereo Visual Odometry based Platooning

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

S.J. van der Marel

June 10, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Development of a Platform for Stereo Visual Odometry based Platooning

by

S.J. van der Marel

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: June 10, 2021

Supervisor(s): ______________________

Reader(s): ______________________

Abstract

As autonomous driving is a popular and ever growing field of research, real world experiments provide a required manner of testing. In this thesis a driving research platform is developed, with a focus on platooning using visual messaging. These visual messages are conveyed using LED matrices. This thesis proposes two methods of LED matrix detection using YOLOv2, one using a sliding window, and one using the entire image. Furthermore two ways of distance estimation are proposed, one using the centers of the estimation bounding boxes and one using the used camera proprietary toolbox depth map. Results from an online experiment show best results from the depth map based depth estimation. The LED matrix detection using a sliding window gave generally dependable results in different environments, at the cost of being computationally demanding. The detection using the entire image provided less consistent results, but was significantly less computationally demanding. In a second offline experiment using a preannotated validation dataset as groundtruth all LED matrices were detected for all detectors. The SqueezeNet based YOLOv2 detector using a sliding window had the best results between tested detectors, with the highest intersection over union between detection and groundtruth.

Contents

Chapter 1

Introduction

Autonomous driving vehicles is a popular field of research since the fifties of the last century, and in the last decade real world testing and commercial applications have become increasingly feasible.

One of the most important arguments to switch to autonomous vehicles is that for over ninety percent of traffic accidents human error contributed to the cause, and for fifty-seven precent it was the sole cause[1]. Despite being a promising technology, autonomous driving vehicles have at the moment of writing not yet reached a level of autonomy where these vehicles are allowed to drive without any human supervision.

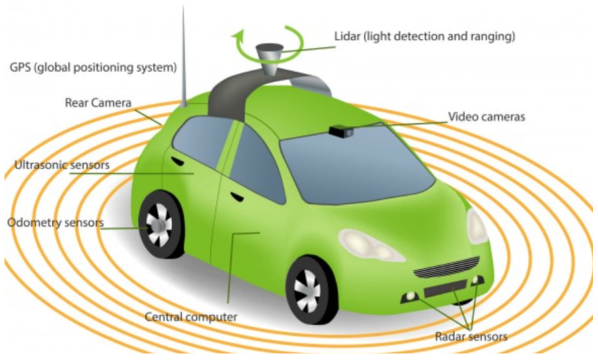


Figure 1-1: Example of sensors implemented for autonomous driving[1].

One of the focal points of this thesis is the development of a platform for research on autonomous driving. A fundamental ability for autonomous vehicles and robots is autonomous navigation. Herein the platform creates an environmental map of its surroundings using sensors, either onboard or external. Using this understanding of the environment such a platform is able then to plan a path and to subsequently execute this plan.

[1]Courtesy of https://www.engineering.com/IOT/ArticleID/18285/How-Sensors-Empower-Autonomous-Driving.aspx

Platooning

Modern intelligent vehicles have to be able to perform a growing number of tasks. Included in these requirements should be the ability to drive within a platoon. Platooning is grouping in such a way that each car is automated to follow the car in front of it. This can have advantages, such as reduced congestion and fuel consumption[2]. Platooning is possible without having direct communication between vehicles, but communication allow cars to predict each other's behaviour more accurately and therefore enhances the earlier mentioned advantages.

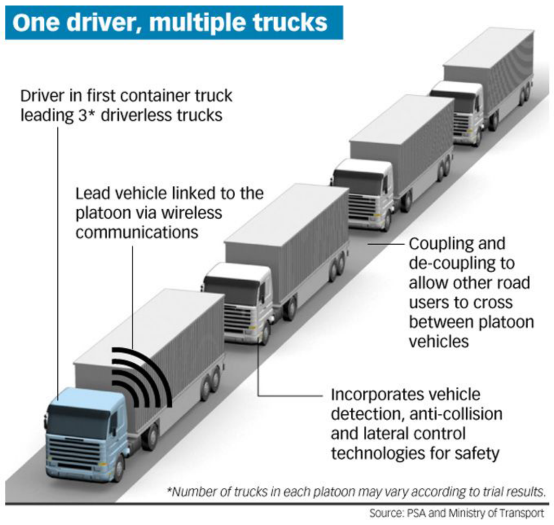

One driver, multiple trucks
Driver in first container truck leading 3* driverless trucks
Lead vehicle linked to the platoon via wireless communications
Coupling and de-coupling to allow other road users to cross between platoon vehicles
Incorporates vehicle detection, anti-collision and lateral control technologies for safety
*Number of trucks in each platoon may vary according to trial results.
Source: PSA and Ministry of Transport

Figure 1-2: Visualization of platooning.

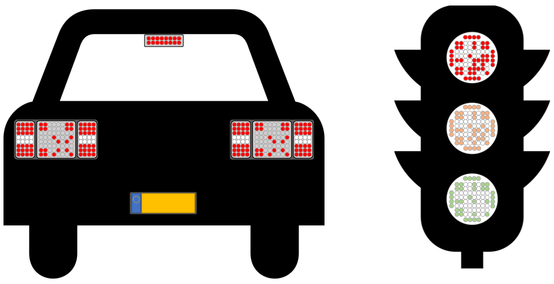


Figure 1-3: Visualization of real life applications of visual markers.

Visual Communication

In order to increase the reliability of these vehicles it can be important to have multiple methods to communicate in-between vehicles and between vehicles and their environment.

As mentioned, semi-autonomous vehicles have a wide variety of sensors, including cameras. This thesis explores the use of these cameras in combination with visual markers to communicate messages and depth estimation. In real world applications this commucation could be run parallel to different types of communications, such as cellular, creating a certain level of redundancy. If one of the communication systems would fail, for example the telecommunications provider experiencing a system failure, this redundancy would allow a different communication system to keep the vehicle completely functional.

This thesis will only explore a laboratory set-up, using 8×8 LED matrices as visual markers, but real world applications could be integrated into car tail lights, traffic signs and lights and the signs of emergency services. For example, traffic lights could communicate the time left until a green light, which allows the car motor's automatic start/stop function to accurately determine whether a full stop would increase fuel economy.

1-1 Overview

Autonomous driving car control algorithms can be divided into three main components:

1. **Recognition**
   Using a (stereo) camera system to create an overview of the surroundings and identify objects, including other cars.
2. **Path planning**
   With the gained knowledge about the surroundings and identified objects, plan an optimal path avoiding obstacles and following the target vehicle.
3. **Velocity Control**
   The velocity and steering angles of the vehicle will be determined using the determined desired path and known properties of the other vehicles. Other cars should be avoided to prevent collisions, also in case of unpredicted behavior, and in the case of platooning a vehicle should properly follow these other cars.

The scope of this study is focussed on the first point, the recognition of the environment using camera imaging.

1-2 Research Goal

This thesis will be focused on development of two subjects based on the previously mentioned parts of the study.

1. *Platform development for platooning*
   With one of the objectives of thesis being the development of a platform suitable for research on platooning, the thesis focuses on the following research question:
   - 𝔸. What is a suited design of the vehicle for platooning purposes while satisfying the design criteria of being easy to reproduce, having a balanced weight and flexible camera placement, while minimizing cost?
2. *Vision based communication*
   In order to develop an algorithm for visual communication, this thesis focuses on the following research questions:
   - 𝔹. Using currently available vision toolboxes, is it viable to consistently localize a LED matrix in a real time scenario?
   - ℂ. Having found a LED matrix, is it viable to consistently estimate the depth of this LED matrix from the camera in a real time scenario?

Chapter 2

Vision

In order to have an understanding of environment recognition of autonomous vehicles, this chapter first discusses general (computer) vision and imaging. First the basics of camera anatomy will be discussed. After this subject general optics will be introduced and be extended to stereo vision.

2-1 Camera Anatomy

A camera can be described as a combination of an optical system and a (digital) image sensor. This optical system is used to focus an image on the sensor by using a combination of lenses, apertures, mirrors and shutters.

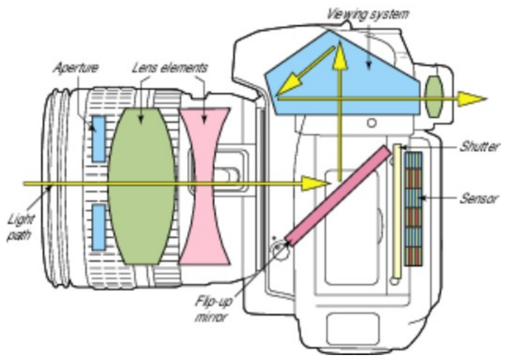


Figure 2-1: Cut through of a modern digital camera.

This section will discuss all relevant parts of the camera, before discussing the typical applications.

2-1-1 Lenses

Lenses are part of the optical system to either focus or disperse light beams. Lenses are commonly made out of glass or translucent plastics and can be used individually or combined in an optical system. Depending on properties of the lens images may be distorted, as shown in fig. 2-2, which can, to a certain extent, be rectified numerically.

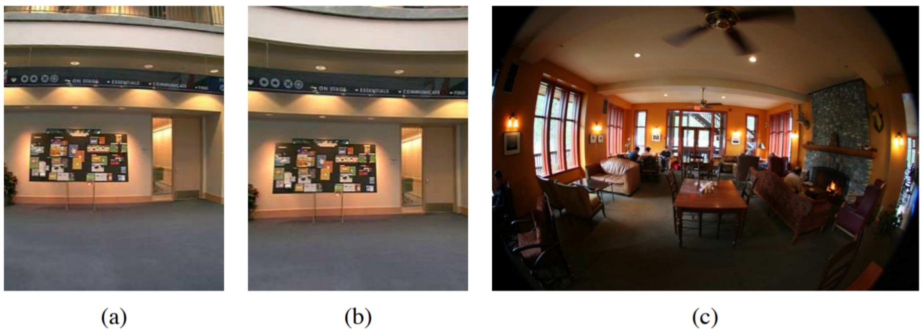


Figure 2-2: Three examples of lens distortion: barrel (a), pincushion (b), and fisheye (c) distortion[3].

2-1-2 Color Filter Array

In order to create accurate images it is important to be able to accurately distinguish different colors. Photosensors however do have little capabilities to do so, with a exception of monochrome sensors which only detect one color. To overcome these limitations color filters can be applied on the sensor, these filters are then placed in a certain configuration to allow optimal color recognition, such a filter is often refered to as a Color Filter Array (CFA). One of the most commonly used is the Bayer CFA, in which the green color is twice as much present as red and blue, therefore often refered to as *RGGB* filter, this and two other types are shown in fig. 2-3.

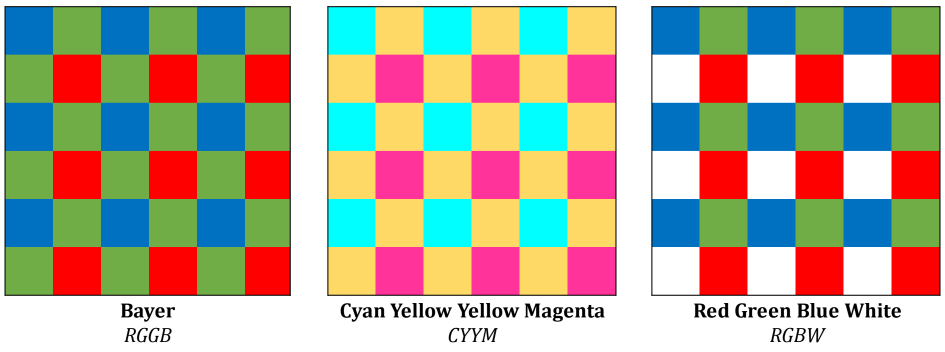


Figure 2-3: Examples of types of CFAs.

2-1-3 Sensor Chip

The sensor chip is basically an array of photosensitive picture (pix) elements, where the term *pixels* stems from. By exciting these arrays based on the source object one can create an image of said object. The two most common sensor chip types are Charge-Coupled Device (CCD) and Complementary Metal-Oxide-Semiconductor (CMOS), which are discussed below.

Charge-Coupled Device based Sensor Chips

One of the most common camera sensor chips are CCD type sensors, visualized in fig. 2-4, these sensor only use one Analog to Digital (A/D) converter for the entire chip. These sensors are built up of two types of active regions, a photosensitive region, built up from a capacitor array, and a transmisson region, made from shift registers.

When exposing the sensor to an image, light falling on the photosensitive capacitor arrays of the CCD causes charge to accumulate in each of the affected capacitors depending on light intensity. After the sensor is exposed the accumulated charge is shifted to its neighbour. In 2D images this is done columnwise and these lines are shifted to the A/D converter, after which a new image can be taken.

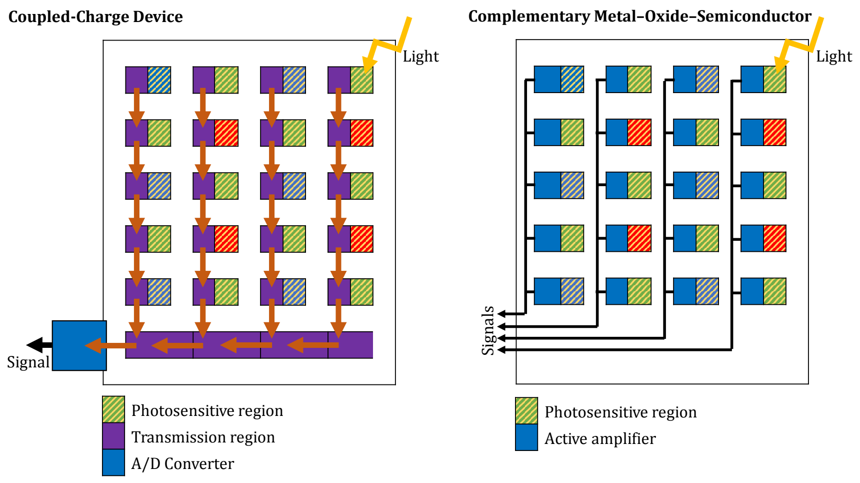


Figure 2-4: Charge-Coupled Device (CCD) and Complementary MetalOxideSemiconductor (CMOS) architecture.

Complementary MetalOxideSemiconductor based Sensor Chips

The CMOS sensor, also known as an active-pixel image sensor, uses an active amplifier per pixel. These amplifiers take up space on the sensor surface leading to a reduced photosensitivity compared to CCD sensors, which may lead to problems capturing images in low light situations. When capturing images this is typically done row-wise at a certain refreshment rate, which in high speed applications may lead to a type of distortion often called the *rolling shutter* effect, this can however be solved by using a global shutter as shown in fig. 2-5. CMOS sensors are generally less expensive compared to CCD sensors, have higher possible frame rates and use less power. CMOS cameras are in general more noisy because the production differences of the pixel amplifiers.

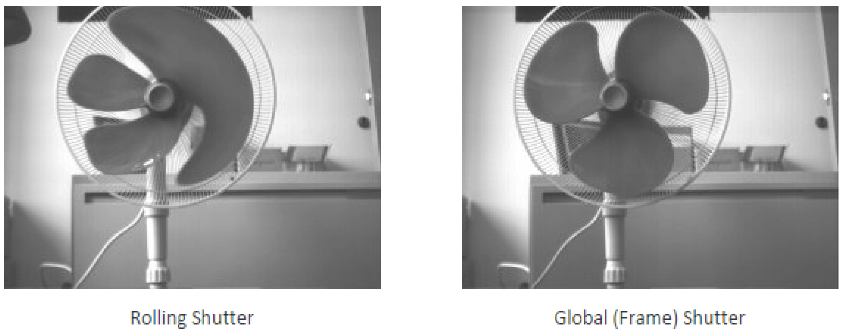


Figure 2-5: Effects of the use of rolling and global shutter CMOS[1].

2-1-4 Trade-off Resolution versus Field of View

Generally there is a trade-off between the Field of View (FoV) and the resolution of a camera. When taking a picture with a camera the image is spread over the entire FoV of its lens. Having a wider FoV means having a lower resolution (in pixels per meter), losing detail in the image of the observed object.

When selecting a system certain design criteria have to be determined concerning the demands on the FoV and minimum resolution.

[1]Source of image: *http://www.cropcopter.co/factors-impacting-uav-sensor-payloads/*

2-2 Optics

2-2-1 Camera Properties

For a camera system, such as the one in fig. 2-6, using a thin lens approximation wherein the thickness is small compared to the curvature radii. The magnification M can be expressed as shown in eq. (2-1). The *thin lense law* is shown in eq. (2-2).

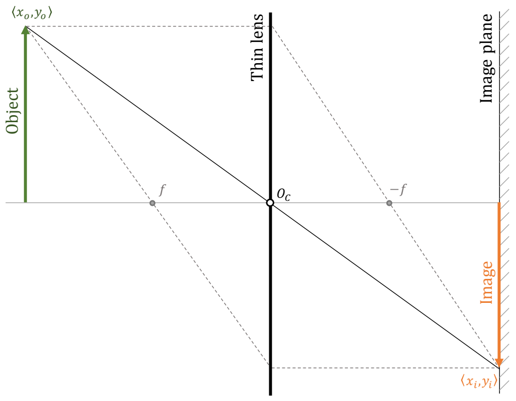


Figure 2-6: Thin lense model visualization.

$$M = \frac{y_i}{y_o} = -\frac{x_i}{x_o} \tag{2-1}$$

$$\frac{1}{f} = \frac{1}{x_i} + \frac{1}{x_o} \tag{2-2}$$

Circle of Confusion

If a camera is not focussed correctly at the object, a circle of confusion appears; an area wherein the ray of light shines on the image sensor. A smaller aperture leads to a smaller circle of confusion, at the cost of darker images. The Depth of Field is the range around the focal point wherein objects are 'acceptably' sharp. For example an object can be acceptably sharp if the circle of confusion is smaller than sensor resolution/pixel size. The image shown in fig. 2-7 gives an example of a non-ideal focussed lens, with the resulting circle of confusion.

In stereo imaging applications the focal point is often set at infinity, which allows objects to be seen sufficiently sharp at finite distances, provided these objects are a certain distance away from the camera.

For a camera focussed at infinity, in other words $z_i = f$, the diameter of the circle of confusion is given by eq. (2-3), wherein d is the aperture diameter.

$$c = \frac{fd}{x_o} \tag{2-3}$$

The Hyperfocal distance H is the distance at which objects are viewed sufficiently sharp, with a maximum allowed circle of confusion c. This c is often based on the pixel size.

$$H = \frac{fd}{c} = \frac{f^2}{Nc} \tag{2-4}$$

For example, the hyperfocal distance of the used ZED camera is around 500mm.

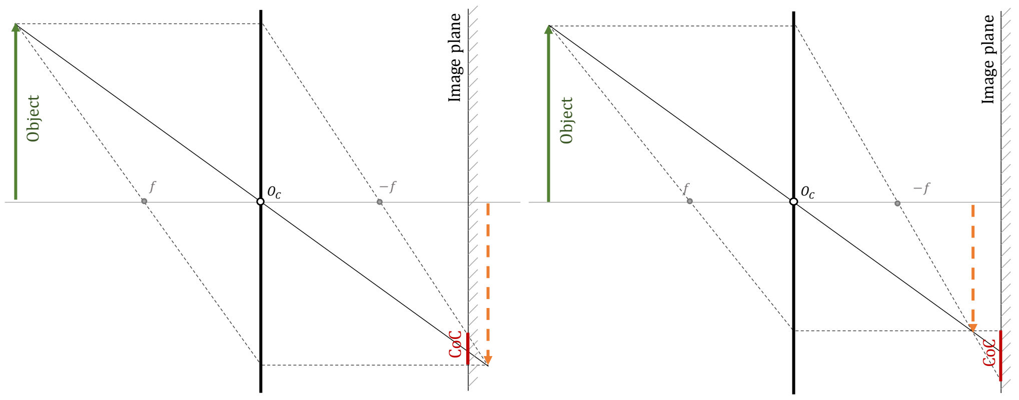


Figure 2-7: Example of a non-ideal focussed lens.

Pinhole Model

A popular way to model a camera is by having a infinitesimal hole in a plate as camera origin or optical center, wherein all detected beams coincide. Afterwards these rays are projected on a image plane at focal distance, as is visualised in fig. 2-8. This result is a simple geometric model, which is also suitable for normal cameras in certain boundary conditions, lens distortions are for example neglected. As the image is rotated by 180° a choice is often made to use a *virtual image plane* instead of the true image plane in models for the sake of simplicity. The size of projections relate as shown in eqs. (2-5) and (2-6).

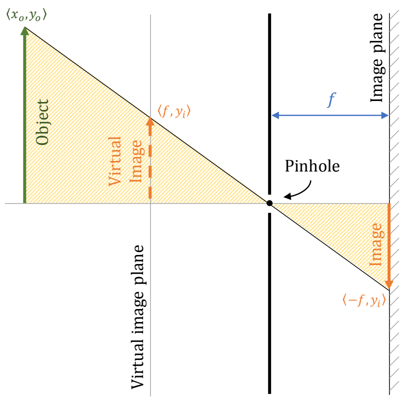


Figure 2-8: World, camera and image coordinate systems, and respective transformations, based on image in [4, p. 11].

$$\text{Image plane: } \frac{y_o}{x_o} = -\frac{y_i}{f} \tag{2-5}$$

$$\text{Virtual image plane: } \frac{y_o}{x_o} = \frac{y_i}{f} \tag{2-6}$$

2-2-2 Projective Geometry

Using the pinhole model it is possible to start modelling the image ray behaviour from real world objects to the sensor plane. Using projective geometry, allowing for rotation, translation, scaling, shear and projection as shown in fig. 2-9, it is possible to completely model this behaviour using a combination of intrinsic and extrinsic transformations, which are explained in this subsection.

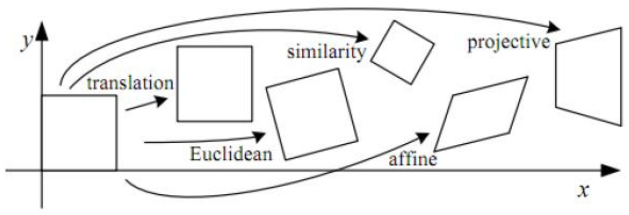


Figure 2-9: Types of transformations[2]

Intrinsic transformations are the transformations from camera to pixel coordinates, extrinsic transformations are those from world coordinate system and the camera coordinate system.

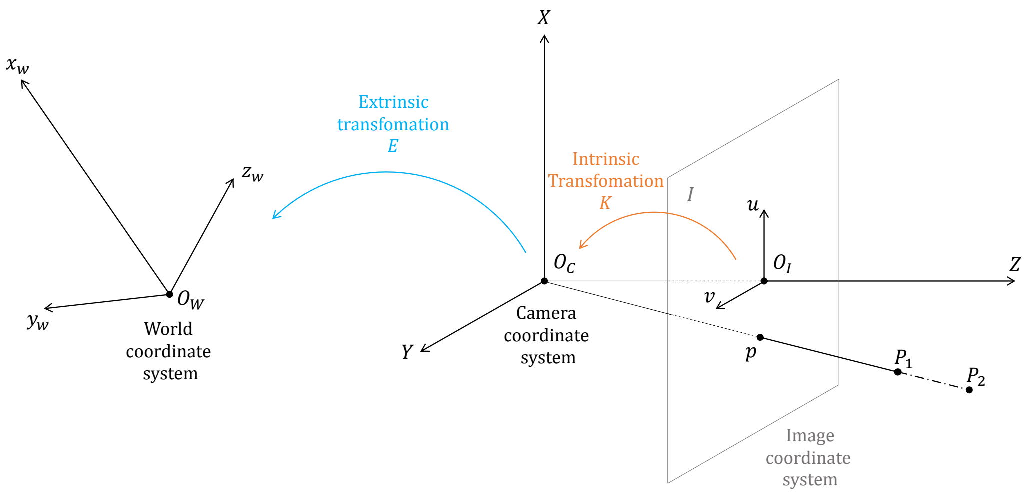


Figure 2-10: World, camera and image coordinate systems, and respective transformations, based on image in [4, p. 11].

Expressing many eucleudian transformations from one coordinate system to another can be described as one matrix multiplication:

$$x_B = R_n\left(R_{n-1}\left(R_{n-2}\left(\ldots\left(R_1 x_A + t_1\right)\ldots\right) + t_{n-2}\right) + t_{n-1}\right) + t_n \rightarrow \tilde{x}_B = E\tilde{x}_A \tag{2-7}$$

[2]https://publiclab.org/notes/anishshah101/06-02-2014/gsoc-update-leaflet-draw-and-non-affine-transformations

Herein $\tilde{x}_B, \tilde{x}_A$ and E are partioned as follows:

$$\tilde{x}_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}, \; E = \begin{bmatrix} R & t \\ 0_3^T & 1 \end{bmatrix}, \quad x, t, 0_3 \in \mathbb{R}^3, E \in \mathbb{R}^{4\times 4}$$

In this particular context the upper transformations are useful to capture any number of real world transformations in a single 4×4-matrix.

Extrinsic Transformation

In the case all real world transformations are gathered in one matrix, this matrix E is called the *Extrinsic transformation*, as shown in eq. (2-8).

$$\tilde{x}_{\text{camera}} = E\tilde{x}_{\text{world}} \tag{2-8}$$

Intrinsic Transformation

Using pinhole model to represent intrinsic transformation, the transformation between pixel coordinates and camera coordinates, leads to the transformation shown in eq. (2-9)[5, p. 47].

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f & \rho & c_x \\ 0 & a \cdot f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2-9}$$

Herein s is a scaling, ρ skewness, f focal distance, c_x,c_y projection of optical center, a aspect ratio. The matrix K is called the intrinsic matrix.

Homography Transformation

Combining a combined extrinsic transformation matrix E and the intrinsic transformation K, the following transformation can be denoted, where H is called the homography matrix[5, p. 46].

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & \rho & c_x \\ 0 & a \cdot f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x_{\text{image}} = KEx_{\text{world}} = Hx_{\text{world}} \tag{2-10}$$

2-3 Stereo Vision

Using one camera, it is possible to gather a large range of information, however one camera set ups are generally quite limited for depth information. As shown in fig. 2-11 using a single camera can be limiting as a closer smaller object, such as N_1, appears at the (virtual) image plane the same as a farther away larger object N_2.

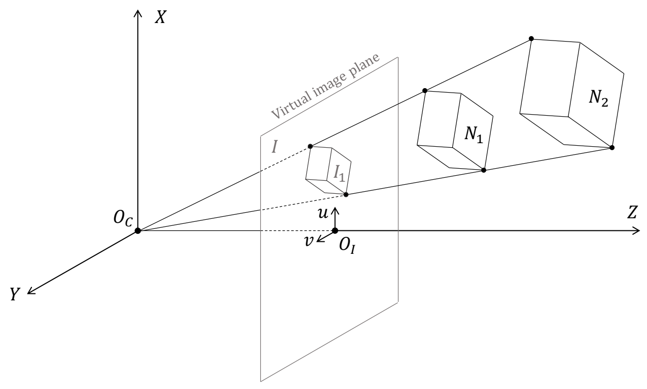


Figure 2-11: Three dimensional objects seen with a single camera system. No difference can be seen between close small objects and larger further away objects in certain configurations.

In order to overcome these limitations a stereo camera setup can be extended by adding a second camera, as shown in fig. 2-12, seperated by a certain distance or *baseline*, denoted by b. Where the original camera, now in this case the left camera, can not detect a difference between N_1 and N_2, the second camera can. By analysing the differences between the images created by both camera's information can be gathered about depth in these images.

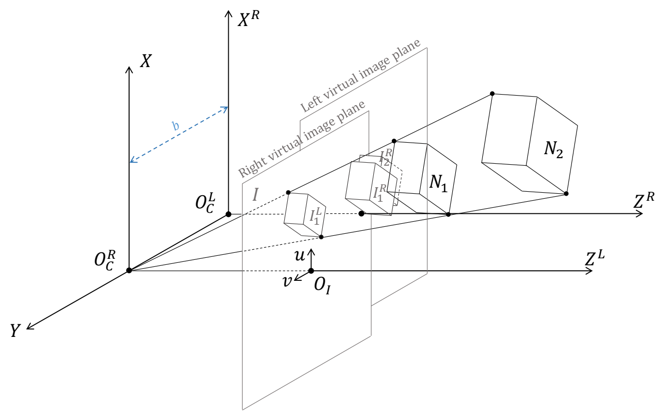


Figure 2-12: Three dimensional objects seen with a stereo camera system. Due to the different viewing angles objects can be differentiated, that would not be visible on a single camera.

This section will discuss the background in among other things depth recognition and the matching of both images, by first introducing concepts such as disparity and epipolar geometry, before continuing to multiple methods to match images.

2-3-1 Disparity

In order to determine the depth of a point the *disparity* of such a point needs to be determined. For a given stereo image camera pair, as shown in fig. 2-13, looking at point of interest p, we observe that both cameras can clearly detect this point. The detected coordinates, however, are different for the left and right camera.

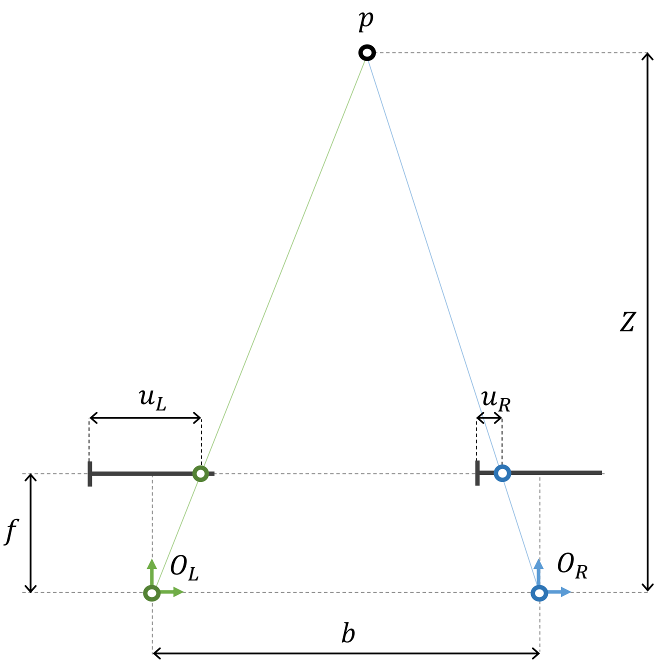


Figure 2-13: A 2D visualisation of dispartiy for a parallel stereo camera system.

This difference in pixel coordinates of similar features in a set of stereo images is called the disparity. For a point which is closer to the camera pair this disparity is larger than for points further away from the camera pair. Using this property the disparity can be used in stereo image sets to determine depth of points and features. The disparity, Δu, as shown in fig. 2-13 relates to the depth of a point as shown in eq. (2-11), the rest of the parameters are defined as shown in fig. 2-13.

$$Z = f\frac{b}{u_L - u_R} = f\frac{b}{\Delta u} \tag{2-11}$$

Points need to be matched between images in order to determine the disparity and depth of a point, which will be discussed in section 2-3-3.

2-3-2 Epipolar Geometry

Shown in fig. 2-14 is a set of stereo images. We consider a case wherein both of the stereo images are still described using the pinhole model, and the image is projected on a virtual image planes in front of the optical centers, O_L and O_R. When both camera's are viewing a point of interest such as point P, we have the projections of that point, $\bar{x}_L$ and $\bar{x}_R$, on the images planes. In the left camera there is no visible difference between point Q and P, because for this camera the points lay in the same ray, while the right camera detects a clear difference. The projection of one optical center on the virtual image plane of the the other camera is called an *epipole*. As this works both ways the four points O_L, O_R, e_L and e_R all lie on the same line, shown in the orange line. The line from point $\bar{x}$ to e is called the *epipolar line*, literature sometimes refers to the epipolar plane, which is spanned by the points P, O_L, O_R.

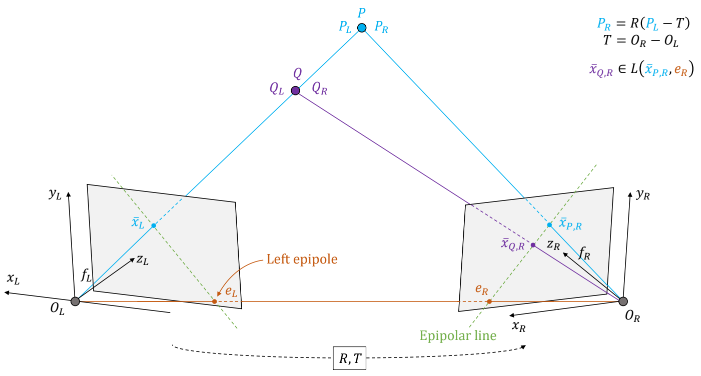


Figure 2-14: Epipolar lines for non-parallel stereo camera system, based on [4, p. 11].

Often *rectification* is applied, a transformation of both (left and right) images such that the transformed stereo system has horizontal epipolar lines, for a given stereo pair, with known intrinsic and extrinsic parameters. After rectification the image points $\bar{x}_L$ and $\bar{x}_R$ always lie on the same epipolar line, a property called the epipolar constraint. This property is quite useful as instead of matching over two axis, there is only one search direction, over the epipolar line, reducing the computation time by an order of magnitude.

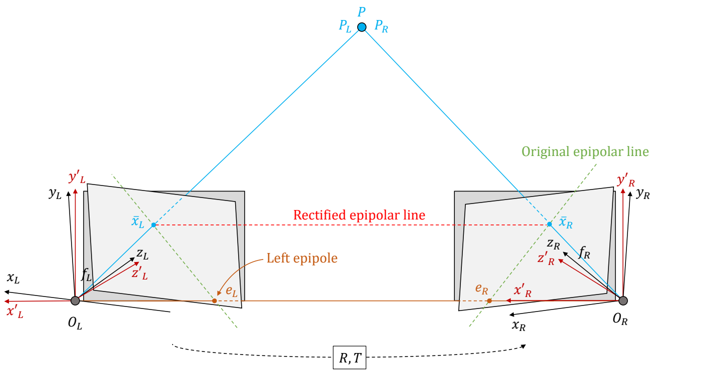


Figure 2-15: Epipolar lines for non-parallel stereo camera system after rectification, based on [4, p. 11].

2-3-3 Stereo Matching

The disparity can not be directly measured, instead it has to be estimated based on a number of possible algorithms which attempt to match points or image features between images. These algorithms can be divided in two categories: dense and sparse stereo matching algorithms, of which the basics will be discussed in this subsection.

Dense Stereo Matching

In dense stereo matching it is attempted to match all pixels, yielding a disparity map of the entire picture based on correlation. As each applicable pixel is matched, this method is computational heavy and prone to noise. In fig. 2-16 an example disparity map is shown which was post-filtered to reduce the noise.

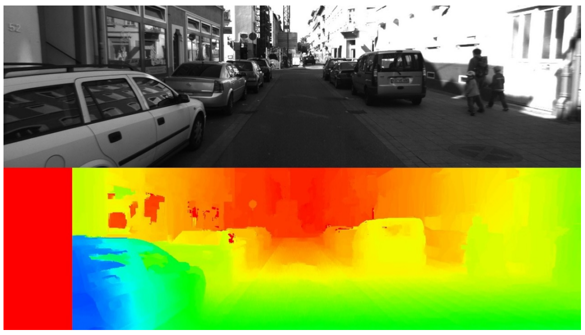


Figure 2-16: Disparity map with post-filtering on KITTI Dataset [6], [7]. In the image there is a sift from blue to red, blueish objects are closer by/ have a higher disparity and reddish objects are further away, thus have a lower disparity. The red rectangle in the left of the image denotes a region with an undefined disparity as that part of the stereo image set does not overlap and therefore can not be matched.

The algorithm can be pictured as shown in fig. 2-17.

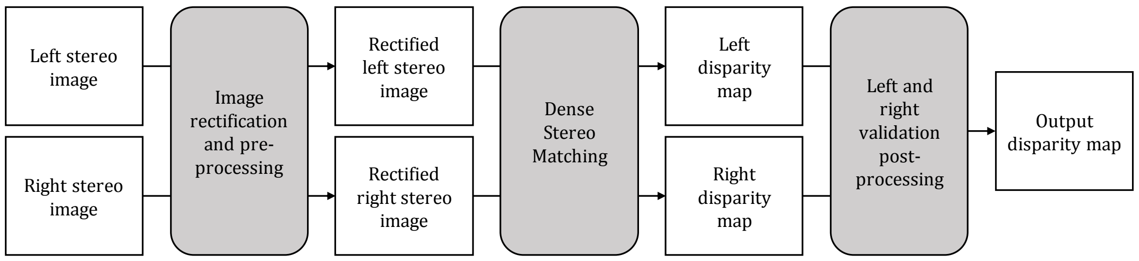

Left stereo image
Right stereo image
Image rectification and pre-processing
Rectified left stereo image
Rectified right stereo image
Dense Stereo Matching
Left disparity map
Right disparity map
Left and right validation post-processing
Output disparity map

Figure 2-17: Dense stereo matching algorithm.

Sparse Stereo Matching

In sparse stereo matching keypoints are sought and matched. This is done using feature descriptors for each keypoint and these are matched between images in feature space. In general this is a faster and possibly very accurate method to match points in both images.

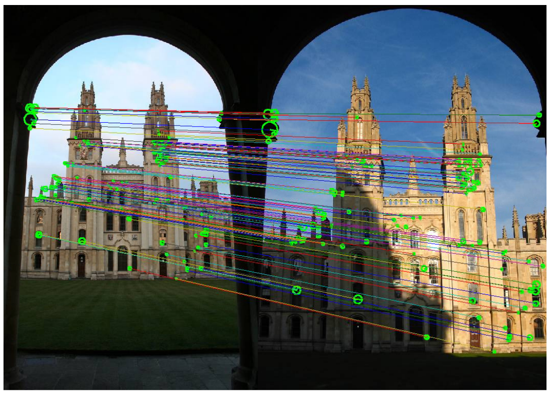


Figure 2-18: Matched features using stereo camera system[8].

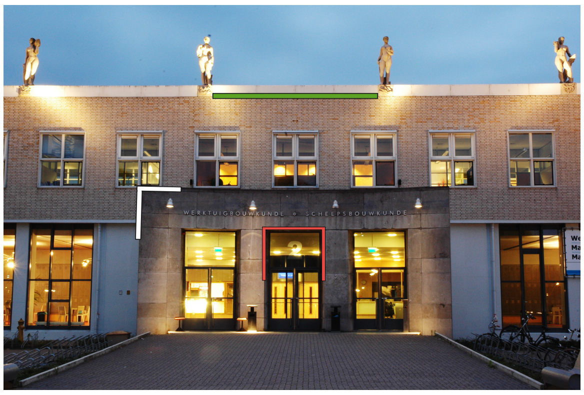


Figure 2-19: Examples of recognizable features, such as edges (green), corners (white), and structures (red).

In sparse matching algorithms features are sought over the epipolar lines in rectified stereo sets, as if the (ideal) rectifying process only allows for horizontal displacements of the search window in the resulting stereo sets. As shown in fig. 2-20 corners can be, within some boundary conditions, uniquely matched between images, horizontal lines however can be translated along with the search window and often can not be matched uniquely.

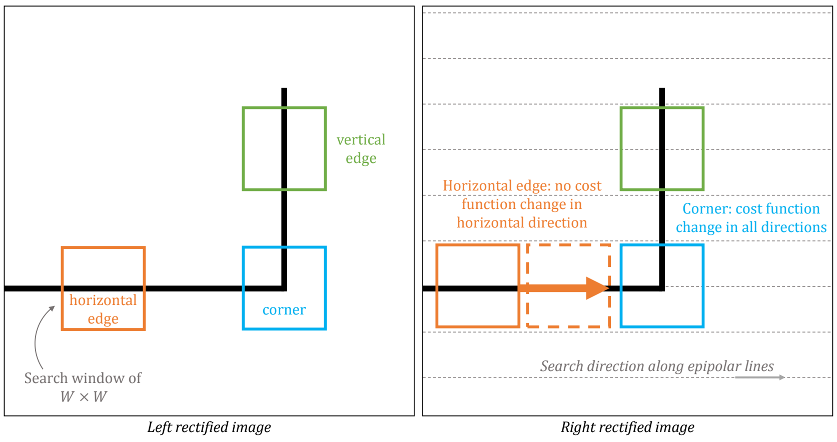

vertical edge
horizontal edge
corner
Search window of $W \times W$
Left rectified image
Horizontal edge: no cost function change in horizontal direction
Corner: cost function change in all directions
Search direction along epipolar lines
Right rectified image

Figure 2-20: Example of feature matching between two images in a rectified set.

There are many challenges in feature matching, either making it more difficult to differentiate between features and/or making it harder to accurately recognize features, such as:

- **Reflective Surfaces**
  False feature matches can result from reflective surfaces, as the difference between the true image and mirror image is for most algorithms difficult to differentiate.
- **Repetitive Features**
  When encountering repetitive features it can be difficult to differentiate between multiple instances in feature matching algorithms.
- **Significant Different Angles**
  In stereo image pairs with large angles features will lose their similarity between images and will therefore be paired less accurately.

2-4 Environment Representation

Where in the previous chapter creating a set of 3D points was discussed, this section will discuss how to construct environmental representations based on the gathered points.

One could for example take the entire point cloud as a low level representation, this is however computationally demanding. Higher level representations, such as object based, $2^1/_2$D grid map-based or stixel based representations, may be a more computationally efficient solution.

2-4-1 Point Cloud

When all gathered points are shown in a single three dimensional coordinate system, the result is called a *Point cloud.* Point clouds are used in creating 3D meshes and can be used to match a detected 3D structure to a model, in which case it is called *Point set registration.*

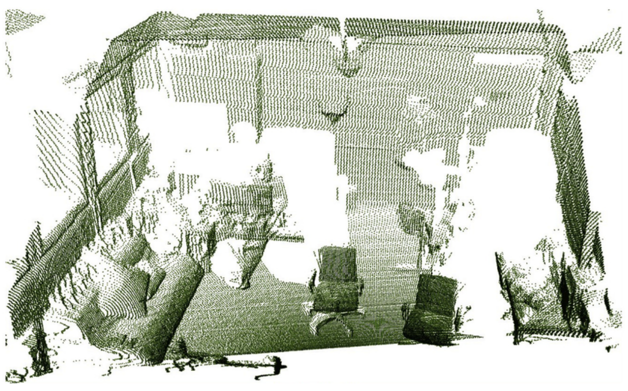


Figure 2-21: Example of a point cloud[3].

2-4-2 Ground Plane Removal

One of the first steps in creating a point cloud is to estimate and remove the ground plane. In doing so a number of assumptions are generally made. Depending on the environment it is often sufficient to estimate flat ground, for example in a laboratory set-up. However, outside laboratory environments the ground planes are often non-flat. On for example bridges and or sloped roads assumptions about the curvature include assuming constant pitch.

[3]Courtesy of https://www.researchgate.net/figure/Example-of-a-point-cloud-representation-of-a-room-visualized-using-the-PCL-library-16_fig3_303863276

Chapter 3

System Description

This chapter discusses the physical platform, the design aim and requirements. A render of the final design is shown in fig. 3-1, with the mayor sensors, actuators and other parts labeled.

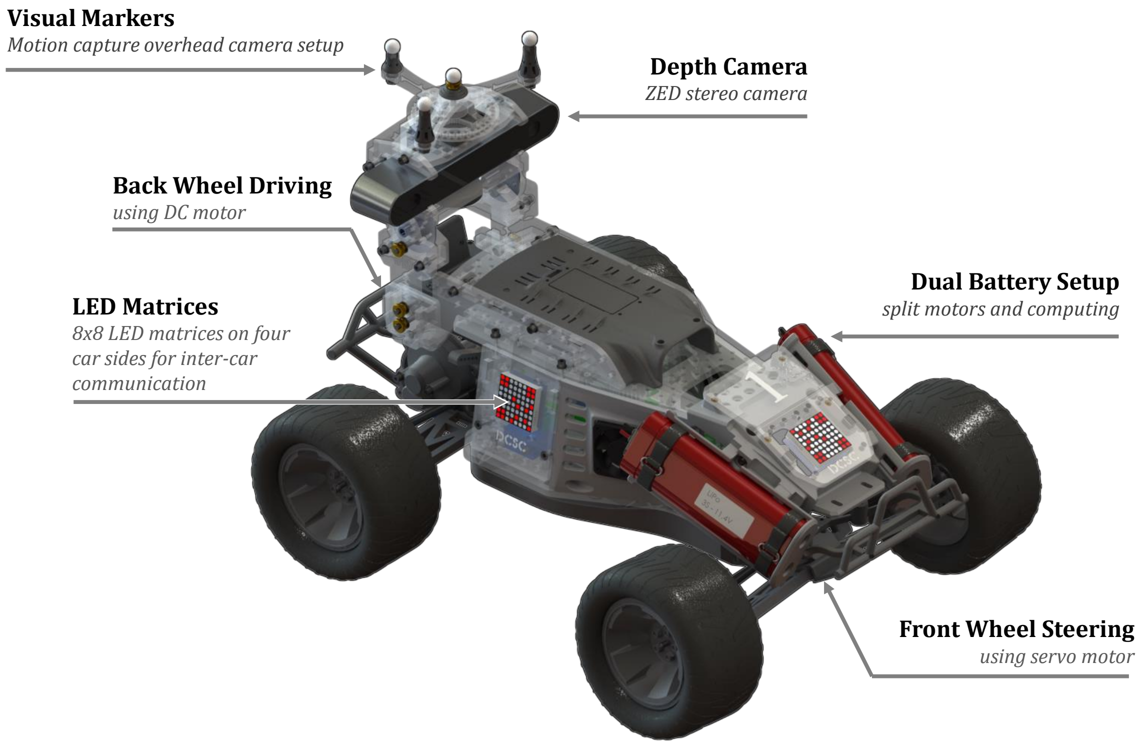


Figure 3-1: Render with sensors, actuators and other visible mayor parts annotated.

3-1 Design Aim

This section discusses first comparable existing opensource projects, after which the design requirements for the platform developed in this thesis will be discussed.

3-1-1 Overview of Comparable Systems

A number of comparable platforms are developed previously, widely ranging in focus, capabilities and cost range. Listed below are some of the most common know opensource 1/16, 1/10 and 1/5 model vehicles.

- **Donkey Car**
  Donkey Car is a 1/16 platform designed for low cost DIY construction aimed towards research on self driving cars. The authors provide the 'donkey library' on their website[1], which can be run on most RC car compatible systems, but recommend the Donkey2 design, a vehicle build from around 250 USD in parts using a shopping list provided on the previously mentioned website.
- **BARC**
  Berkeley Autonomous Race Car (BARC)[9] is a fairly low cost 1/10 development platform for autonomous driving. The system is equiped with among other sensors sensors, the ELP USB3 camera module, an IMU and multiple line sensors (to research lane changing). There are optionally more expensive sensors supported, such the RP LiDar 360.
- **MIT RACECAR**
  The Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot, or RACECAR[10], is an 1/10 scale platform developed by MIT, housing state of the art hardware. The RACECAR is equipped with, among other sensors, a Hokuyo UST 10LX 2D LiDar, ZED stereo camera and an IMU. Furthermore the system is equiped with a VESC[2] and a NVIDIA Jetson TX1 System on Module (SoM).
- **F1/10**
  The F1/10 is a one to ten scale model project founded as a 'Cyber-Physical Platform' (CPS)[11], in order to allow researchers to perform real world experiments and testing. The vehicle is equipped with, among other sensors, a Hokuyo UST 10LX 2D LiDar, a stereo camera, a FLIR flea camera and an IMU. Furthermore the system is equipped with a Foebox VESC X and a NVIDIA Jetson TX2 SoM. A system overview is shown in fig. 3-2.
- **AutoRally**
  The rugged AutoRally platform is a platform focused on aggressive autonomous driving and high speed maneuvers[12]. The system is equiped with a Intel Skylake i7 quad core processor and a NVIDIA GTX-750ti GPU and a number of sensors. These sensors include a GPS sensor, IMU and two FLIR Flea3 FL3-U3-13E4C-C cameras. A photo of the AutoRally without the protective cover is shown in fig. 3-3.

In table 3-1 an overview is shown of these systems.

[1]https://www.donkeycar.com/

[2]Opensource smart Electronic Speed Controller (ESC), the Vedder ESC, or VESC, project website `http://vedder.se/2015/01/vesc-open-source-esc/`

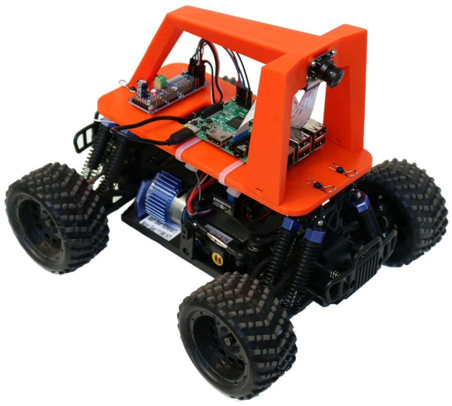


Figure 3-2: Photo of the Donkey Car, courtesy of the project website.

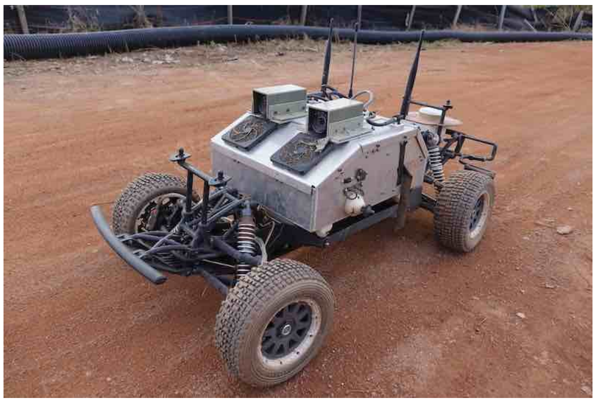


Figure 3-3: System overview of the AutoRally platform[12].

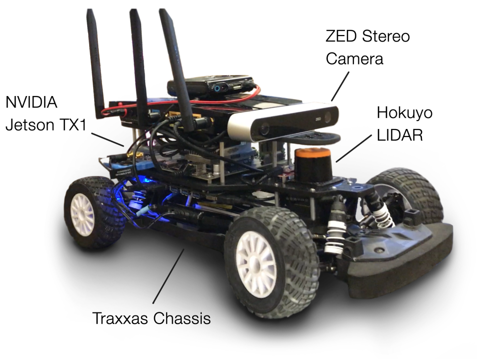


Figure 3-4: System overview of the MIT RACECAR[10].

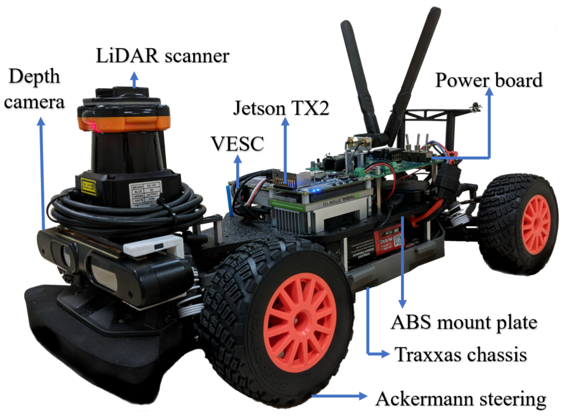


Figure 3-5: System overview of the F1/10 platform[11].

Platform	Size	Cost in [USD]	Construction time in [h]	Weight in [kg]	Computation Module
Donkey Car	1/16	250	2	2	Raspberry Pi
BARC	1/10	500	3	3.2	Odroid XU4
MIT RACECAR	1/10	3383	10	4.5	NVIDIA Jetson TX1
F1/10	1/10	3628	3	4.5	NVIDIA Jetson TX2
AutoRally	1/5	14000	not specified	21	Mini-ITX Intel Skylake i7

Table 3-1: Comparison between existing open source scaled autonomous platforms and the research *aim* of this vehicle[12]. The build costs (and time) of each car does not include 3D printed parts as costs vary heavily between manufacturers and can often be done for free in house at research facilities.

3-1-2 Design Requirements

This section discusses the design requirements for the vehicle. An overview of the system is shown in fig. 3-7, herein the mayor mechanical and electrical parts are labelled. The developed system was designed with a number of main design criteria:

1. **Scalable fleet**
   As the vehicle is developed for platooning one of the design criteria is the scalability of the size of the fleet, in other words, it should be easy to create extra vehicles when required for an experimental set-up.
2. **Flexible camera placement**
   The camera position and pitch should be adjustable for each car. For the current experimental set-up in a laboratory environment no automatic camera pitch adjustment is required, this could be different in later iterations wherein the car drives on slopes and hills.
3. **Balanced weight**
   In order to keep the propulsion and steering properties as intended, the center of gravity should lie around the middle of the vehicle.
4. **Minimizing cost**
   One of the design criteria is to keep the cost per vehicle low, apart from the systems that can not be avoided - such as the ZED camera, NVIDIA Jetson TX2 and other essential parts.

3-2 Design Considerations

The previous section listed the design aim and requirements. This section discusses considerations about satisfying these requirements, some of which are visualized in fig. 3-6. This thesis aims to to develop a platform in the 'mid range' compared to the in section 3-1-1 mentioned vehicles, with capabilities and costs between the BARC and the MIT RACECAR and F1/10. The thesis aims at a cost around €1800.00 per platform, while having a powerful stereo camera and processing capabilities.

- **Camera Location**
  The camera is required to be placed at the end of the car, in order to be able to focus on objects in front of the vehicle, because of the hyperfocal distance of the camera.
- **Battery Location**
  Moving the camera backwards shifts the center of gravity towards the back, introducing difficulties while steering. The batteries are places in the front of the car in order to counteract this effect. This moves the center of gravity, according to the CAD software, to the point indicated in fig. 3-6.
- **Scalable Low Cost Design**
  The system is constructed mainly from laser cut acrylic and 3D printed parts, which

generally low cost compared to alternatives. In section 3-6 there will be more information available about this.

- **Modular Design**
  In most designs there is a trade-off between modularity and integration of parts. In this design there is a large focus on modularity, which comes at the cost of size. The system is build up in different sub assemblies, as shown in fig. 3-7, which can be constructed apart from each other, with the exception of the LED matrix connection as these are each connected to a previous one. Combining these sub assemblies is done with a few screws, allowing for replacement of a broken part when necessary.

- **Camera Placement**
  The system has two camera systems, the main ZED stereo camera and one integrated in the ErleBrain 3. The stereo camera is mounted in manner which allows manual vertical and pitch adjustments, as shown in fig. 3-6. The integrated ErleBrain 3 camera is fixed pointing behind the vehicle, with possible uses in later research.

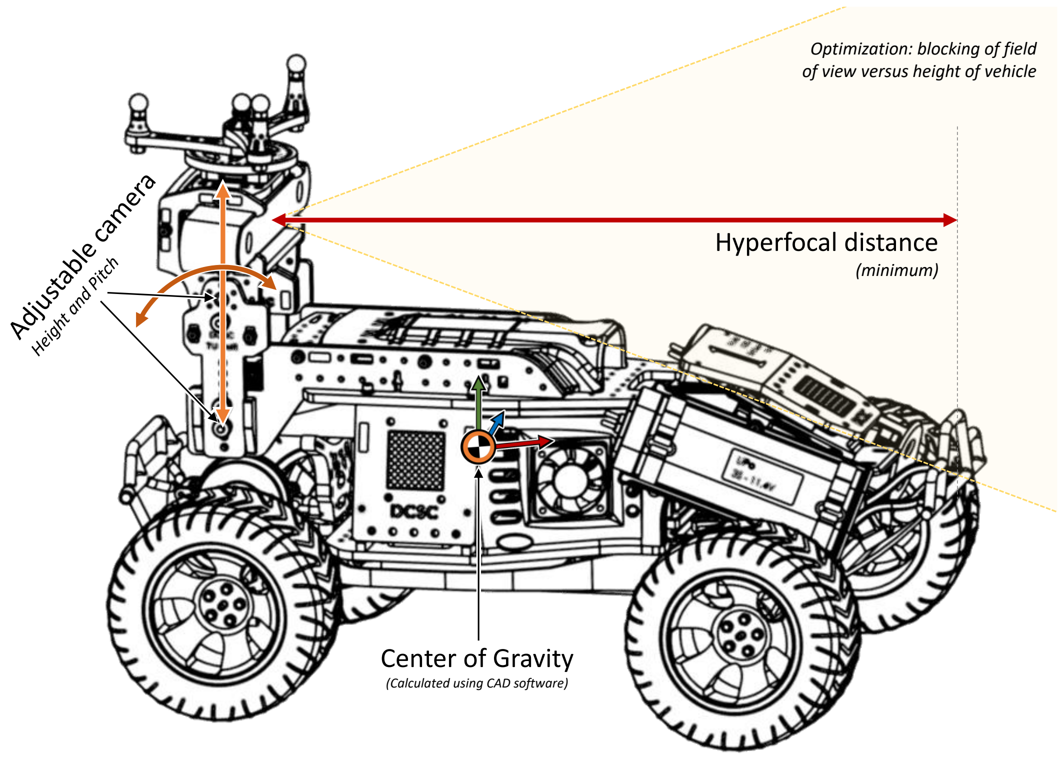

Optimization: blocking of field of view versus height of vehicle
Adjustable camera
Height and Pitch
Hyperfocal distance
(minimum)
Center of Gravity
(Calculated using CAD software)
DCSC

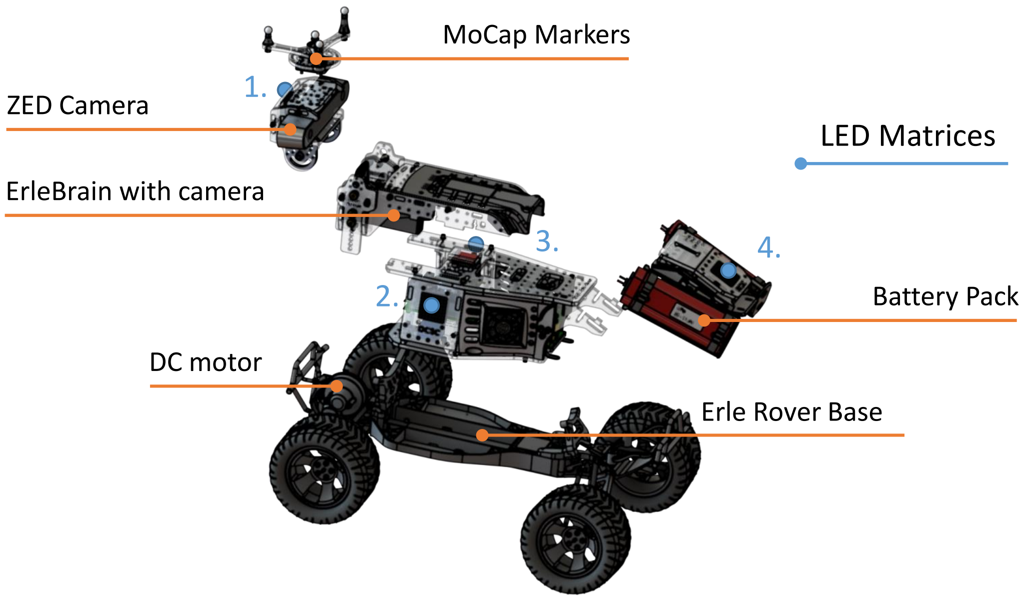
Figure 3-6: Visualized design criteria.

3-3 Overview of Components

This section describes the mayor components used in the vehicle. A summarized list of the mayor components and their purposes is shown below:

- **ZED Camera**
  The ZED stereo camera used for robot vision and depth estimation.
- **NVIDIA Jetson TX2**
  The NVIDIA Jetson TX2 is a high performance SoM used for camera processing. The Jetson is mounted on a ConnectTech Orbitty Carrier board.
- **ErleBrain 3**
  The ErleBrain 3 is a Raspberry Pi based computer which controls the vehicles motion.
- **Batteries**
  The system uses two batteries. One battery mainly for the motion and ErleBrain, one battery mainly for the NVIDIA Jetson and camera.
- **LED Matrices**
  Four LED matrices are mounted on the front, left, right and backside of the vehicle in order to convey visual messages. These LED matrices each have a dedicated driver and are controlled by an Arduino.
- **MoCap Markers**
  In order for the motion camera system to estimate the location of the vehicle, this vehicle needs to be equipped with visual markers. These markers need to have a unique configuration per used vehicle in order to differentiate between these.
- **Propulsion and Steering Motors**
  A DC motor is used as a propulsion motor with a matching RC car Electronic Speed Controller (ESC). For steering a servo motor is used. Both the servo and ESC are controlled by the ErleBrain.

An overview of the system is shown in fig. 3-7, and these components are connected as shown in fig. 3-8. The following subsections expand on the mayor components discussed above.



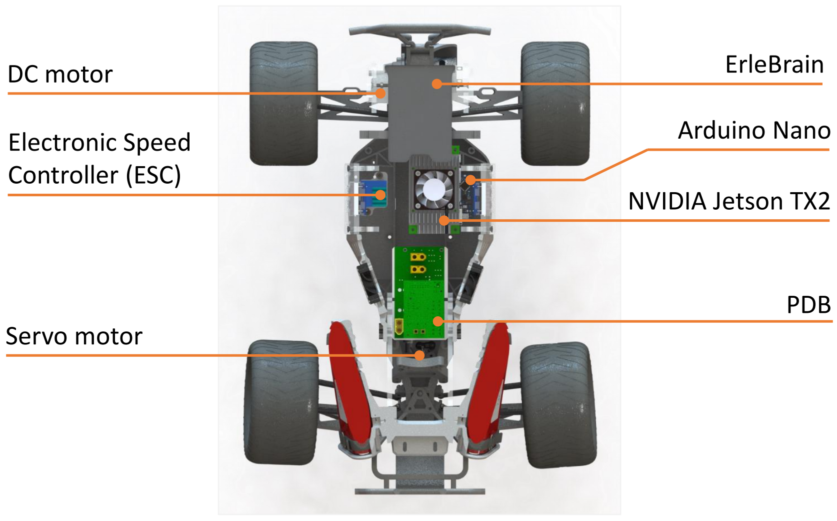


Figure 3-7: System overview; exploded view with mayor parts annotated *(top)*, and cutthrough with large part of electrical system visible *(bottom)*.

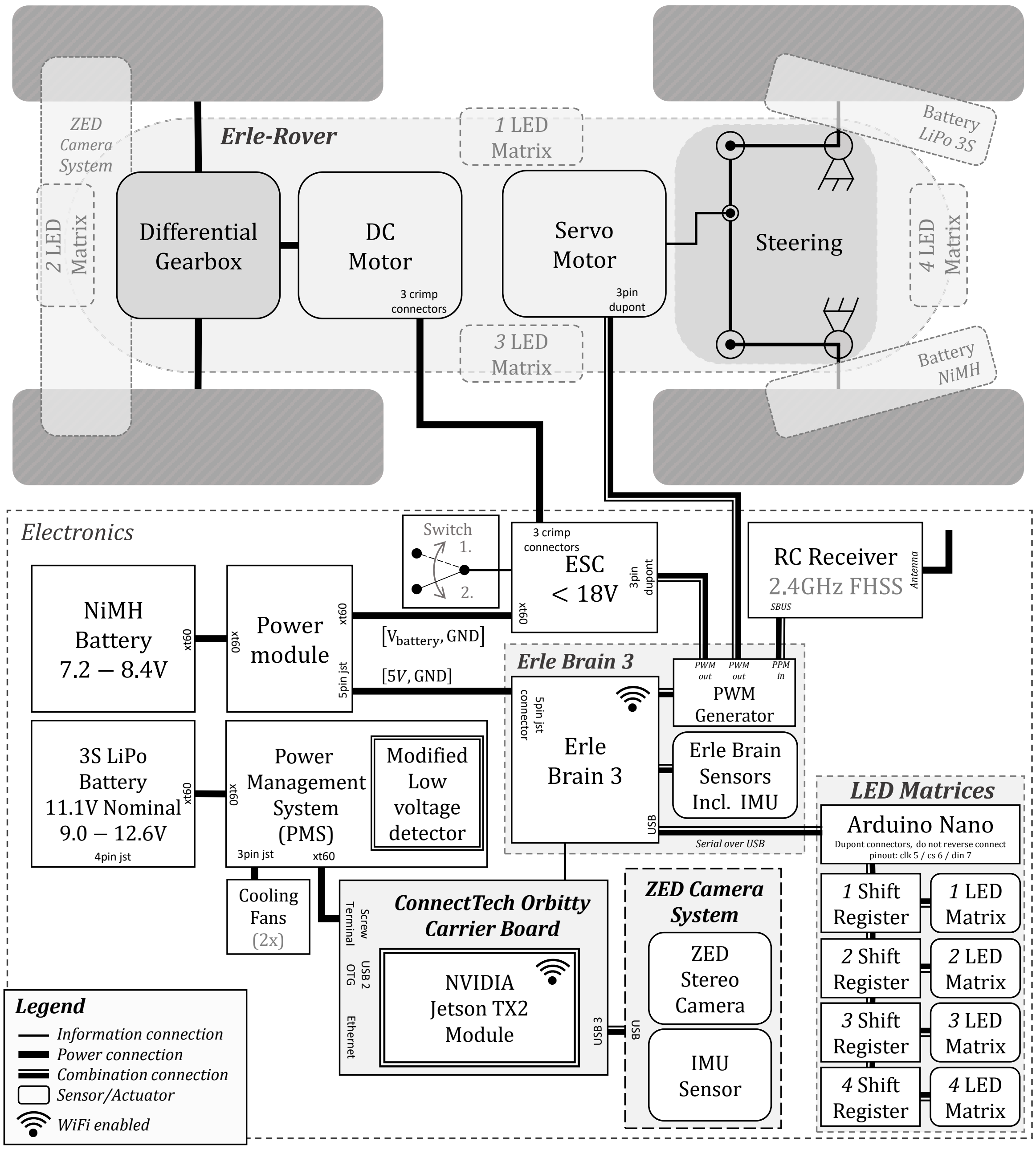


Figure 3-8: Overview of the complete electrical system.

3-3-1 ZED Camera

The ZED camera, shown in fig. 3-9, is a dual 4 MP camera set up with specifications as shown in table 3-2. The camera has a wide viewing angle and relatively low resolution, a trade off discussed in section 2-1-4. For a driving vehicle this suits the purpose of having a wide overview of the surrounding, at the cost of losing some details. These details, often required to observe further away objects, often might not be as important as having a complete overview. The ZED camera Software Development Kit (SDK) includes an extensive scala of algorithms which include similar imaging processes as described. Due to the proprietary nature of the toolbox it is unknown which exact methods/algorithms the toolbox uses.

Video		
Capture Mode	FPS	Resolution
2.2K	15	4416×1242
1080p	30	3840×1080
720p	60	2560×720
WVGA	100	1344×376

Depth		
Depth range	0.5-20	m
Stereo baseline	120	mm

Figure 3-9: The ZED camera.

Lens	
Aperture	f/2.0
Field of View	90°(H) × 60°(V) × 110°(D) max

Sensor	
Resolution	4M pixels per sensor with 2 μm pixels

Table 3-2: The manufacturer specifications of the ZED camera.

3-3-2 Erle Brain 3

The Erle Brain 3 is the third generation of a Linux based autopilot for, among other systems, the Erle Rover, with official support for ROS. This system combines a Raspberry Pi embedded computer with a daughter board containing power electronics, a PPM receiver, a PWM generator and multiple sensors. These sensors include a 5MP camera, an IMU, temperature and pressure sensors. The Erle Brains PWM generator is used to relay the transmitter commands to the actuators (ESC/motor combination and servo motor). These commands are either directly relayed via the transmitters or generated in the Erle Brain depending on a physical switch.

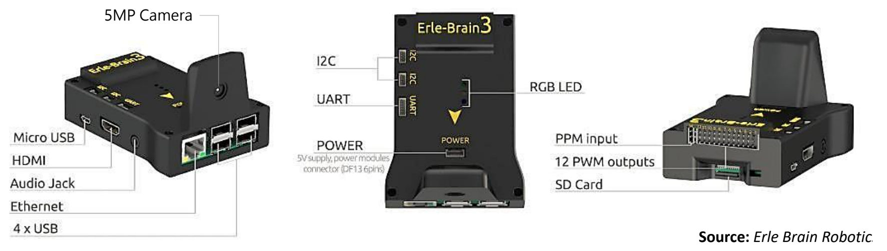


Figure 3-10: The Erlebrain 3 embedded computer.[3]

[3]Image courtesy of manufacturer, `https://erlerobotics.com/`, accessed on January 2020, website now unavailable due to bankruptcy of Erle Robotics.

3-3-3 NVIDIA Jetson TX2

The NVIDIA Jetson TX2 module is a embedded System on Module (SoM) used in the system in order to process the camera images of the ZED stereo camera system.

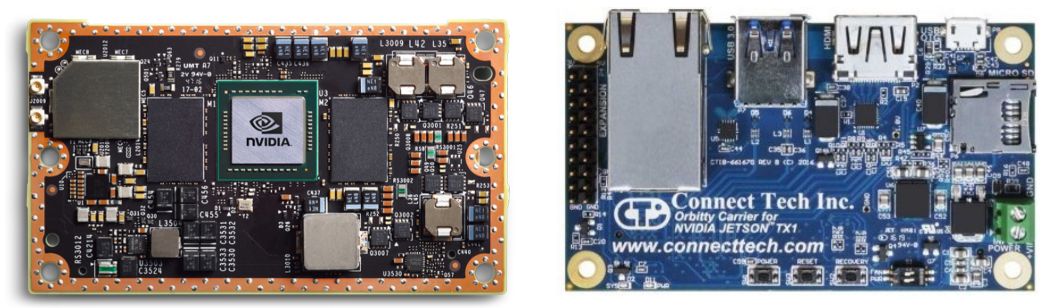


Figure 3-11: Jetson TX2 module *(Left)*, ConnectTech Orbitty carrier board*(Right)*[4].

The TX2 module is used in combination with the ConnectTech Orbitty carrier board. The carrier board is connected as shown in fig. 3-8.

Table 3-3: Technical specifications of the NVIDIA TX2 module

NVIDIA Jetson TX2 module	
GPU	256-core NVIDIA Pascal GPU architecture with NVIDIA CUDA cores
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU
	Quad-Core ARM© Cortex©-A57 MPCore
Memory	8GB 128-bit LPDDR4 Memory 1866 MHx - 59.7 GB/s
Storage	32GB eMMC 5.1
Power	7.5W / 15W
WiFi	Yes
ConnectTech Orbitty carrier board	
Size	87 mm $\times$ 50 mm
Connections	GbE, HDMI, USB 3.0, USB 2.0, 2x 3.3 V UART, MicroSD, I2C, 4x GPIO
Voltage	9 V to 14 V DC Nominal, 19 V Peak
Weight	41 g (Carrier only), 144 g (Carrier + Jetson TX2)

3-3-4 Batteries

The system uses two batteries. One 1600 mAh NiMH battery, with six cells, for the standard Erle Robotics Erle Rover, with a nominal voltage of 7.2 V. This battery powers the ErleBrain, the servo motor for steering, the DC motor for forward motion, and the 2.4 GHz RC receiver for operation via a standard RC car remote control. The second battery, a 3800 mAh 3S LiPo battery powers the NVIDIA Jetson, ZED camera and LED matrices.

[4]Both images courtesy of manufacturers, https://developer.nvidia.com/embedded/jetson-tx2, http://connecttech.com/product/orbitty-carrier-for-nvidia-jetson-tx2-tx1/

3-3-5 Power Distribution Board

The Power Distribution Board (PDB) is designed to distribute power in the vehicle and to automatically turn off when the battery voltage is below a predefined threshold. This is achieved using a off the shelf low battery alarm, which triggers a latching relay in the configuration shown in the image. Pressing the momentary on-off button resets the system, in order to turn on the setup. The six pole latching changeover switch cuts all power, allowing for safe storage or charging the battery in place. A schematic of the main power distribution board is shown in fig. 3-12, wherein the low voltage module, shown in fig. 3-14, is included as the symbol denoted as *'undervoltage circuit'*. The power to the motors and ErleBrain is in the standard configuration on a separate battery, but may be connected to the PDB as well. The figures shown in figs. 3-13 and 3-15 show the design in the CAD software.

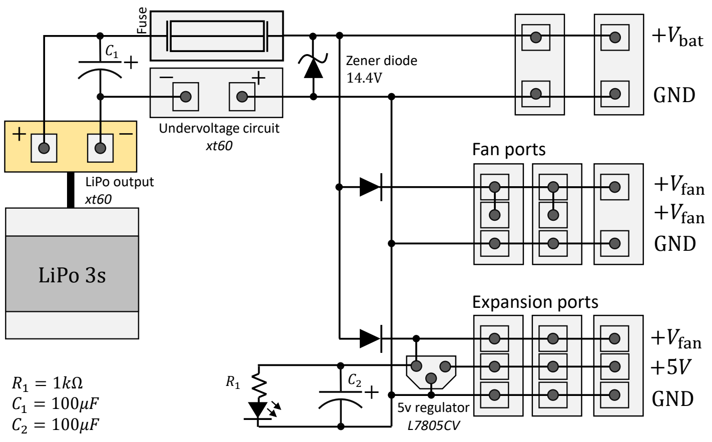


Figure 3-12: Power distribution board: Power Module.

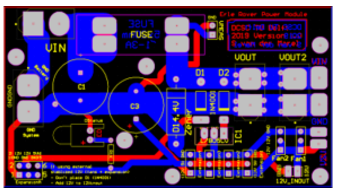


Figure 3-13: Power distribution board: Power module.

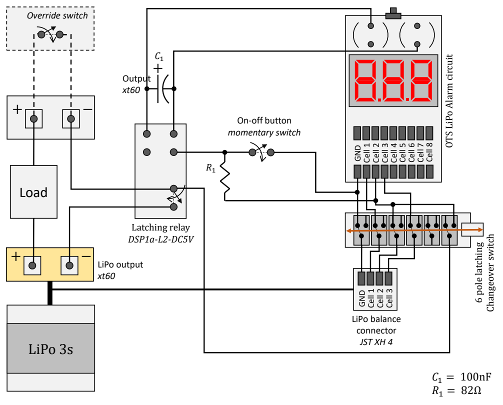


Figure 3-14: Power distribution board: Low voltage module.

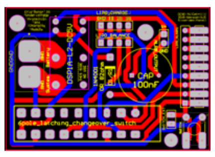


Figure 3-15: Power distribution board: Low voltage module.

The production of the printed circuit boards was done by a production company[5], to which the 'Gerber files' and drill hole coordinates were provided, both of which were exported in the CAD software. The soldering of the components was done in house.

[5]SeeedStudio, a Chinese based electronics company.

3-4 LED Matrices

In order to implement the LED matrix, two set ups are created, one to implement in the car and one for data generation and software testing purposes. In both set-ups a number of LED matrices will be controlled using an Arduino Nano and dedicated drivers connected to the arduino via a SPI connection. The connections are shown in fig. 3-16.

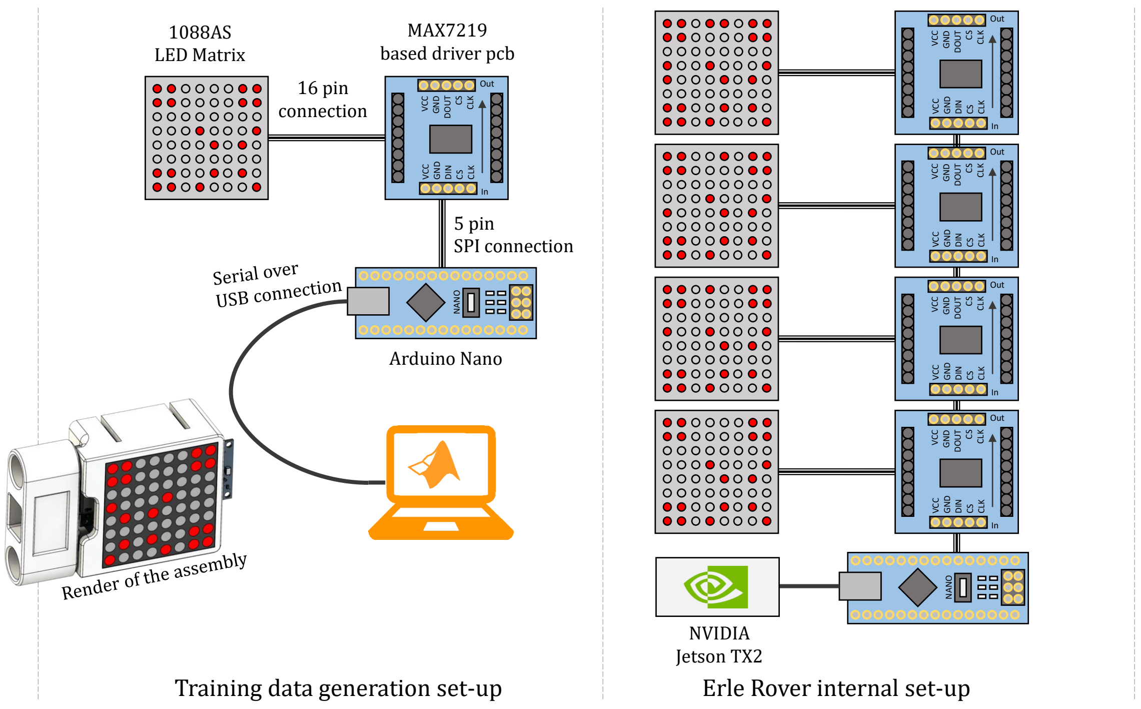


Figure 3-16: Connections for the LED matrix subsystem.

3-5 Visual Messages

Before the recognition of the LED matrices is discussed the structure of the messages will be discussed in this section. First common QR codes will be discussed before we move on to the used messages.

3-5-1 Two Dimensional Barcodes

Two dimensional barcodes, and the better known *'Quick Response (QR) Codes'*, refer to two dimensional patterns which encode data. Typical applications of these codes include (commercial and industrial) labelling and a growing number of applications in augmented reality.

Standard QR codes include some elements to denote position, alignment, spacing and formatting and version information. The image shown in fig. 3-17, denotes the template for a QR codes [13], [14], with the mayor elements highlighted.

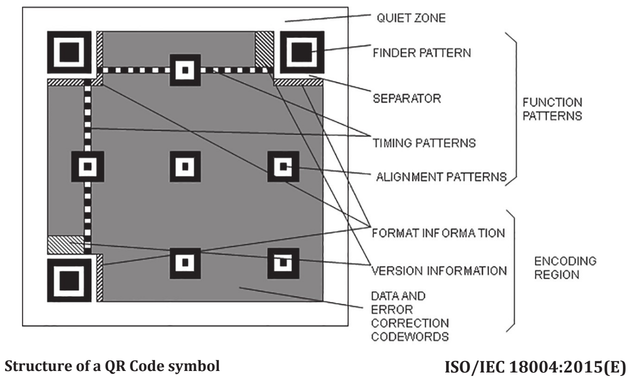


Figure 3-17: ISO/IEC 18004:2015

3-5-2 Implementing Two Dimensional Barcodes

Standard type QR codes start at a resolution of 21 × 21-elements (version 1), where the used set-up has a maximum of 8×8 resolution, therefore no standard QR-code can be implemented, so a custom algorithm has to be created, for example as shown in fig. 3-18, with a simple predetermined set of expressions.

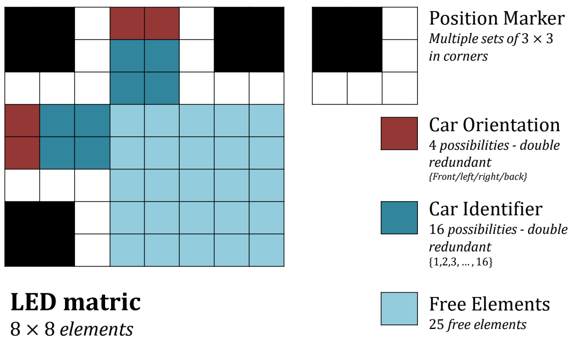


Figure 3-18: Example QR code based algorithm

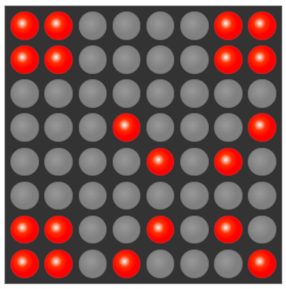


Figure 3-19: Example QR code resulting from algorithm shown in fig. 3-18

An example of a LED matrix expression is shown in fig. 3-19, this expression will be used as main expression throughout the thesis.

3-6 Production and Construction of Custom Parts

This subsection discusses the production techniques and assembly of the vehicle. Most of the vehicle is constructed from lasercut acrylic parts and 3D printed parts, the paragraphs below expand on these techniques.

Lasercut Parts

The modifications on the Erle rover are largely made of lasercut acrylic. The choice for lasercutting as production process is based on the quick production speed, low costs and relative high tolerances. A drawback of lasercutting is that all operations have to be two-dimensional.

The choice for acrylic, also known Poly(methyl methacrylate) (PMMA), is because of its thermal and non toxic properties it is suited to be lasercut, which is the main production process of the frame, while being a relative light weight material and relative inexpensive.

Joints between parts are generally as shown in fig. 3-20. The screw absorbs the for in the direction perpendicular to the part screwed on the base part, as shown in red, while the slot/nook construction absorbs forces in plane with this part, as shown in green. While assembling one should be carefully to not overtighten the screws, as a primary failure mode of this design is cracking of the t-slot between the nut and the head of the bolt.

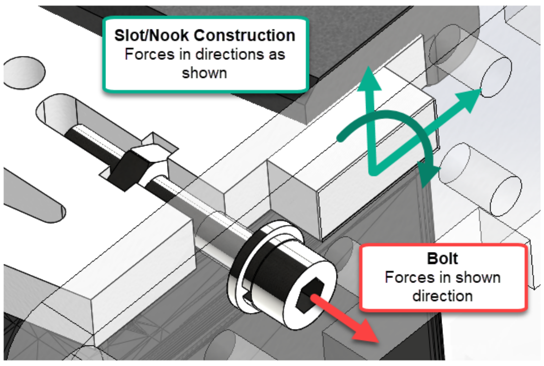


Figure 3-20: The used method to how the vehicle is assembled. The forces are absorbed in shown directions.

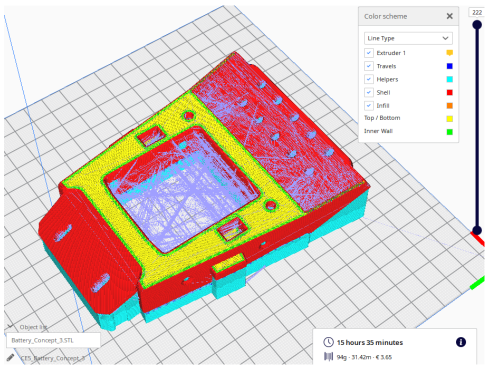


Figure 3-21: Example of interface of the used 'Cura' - CAM software for one of the parts in the front sub assembly.

3D Printed Part

3D printed parts are generally low cost compared to other methods to create similar part. A disadvantage of 3D printing is that it is, generally, a time consuming production method, printing parts takes generally more time than part creation with other production methods, and part post processing is time consuming for most prints. Post processing includes removing print supports, removing bed adhesion constructions and refining the surface finish.

3-7 Cost Overview

In table 3-4 the part costs, excluding shipping, are shown. The total indicative cost, are within the earlier discussed price range. As the Erle Robotics Erle Rover is no longer available, a suited replacement is shown here, a LRP S10 Twister 2 2WD monster truck in combination with a Pixhawk PX4. At this moment the back camera from the ErleBrain 3 is not implemented, and not added to the parts list, however there are numerous available if a necessity should arise.

Catergory	**Component**	**Amount**	**Cost** per Component
Off the Shelf	StereoLabs ZED Camera	1	$349.00
	Orbitty carrier board	1	$174.00
	NVIDIA Jetson TX2	1	€440.00
	Computer fan, 12 V, 40×40 mm	2	€5.00
	MoCap Markers	4	€5.00
	LiPo 3s 3400 mAh battery	1	€60.00
	LiPo alarm	1	€5.00
	LED matrix + Driver	4	€2.50
	Arduino Nano	1	€30.00
	~~Erle Robotics Erle Rover~~	~~1~~	
	LRP S10 Twister 2 Monster Truck 2WD	1	€150.00
	Pixhawk PX4	1	€195.00
Material	3D printer filament: PETG	1 kg	€25.00
	Acrylic sheet, 5 mm	0.5 m^2	€20.00
	Fasteners		€10.00
	Cables and connectors		€15.00
	General consumables, solder, heat shrink, etc		€15.00
External Production	Power distribution PCB (10×)	1	€10.00
	Undervoltage PCB (10×)	1	€10.00
Total	Indication value, excluding shipping		€1550

Table 3-4: Component and indication price overview, excluding shipping.

Chapter 4

LED Matrix Recognition

When a human looks at a situation it can recognize objects accurately, almost instantaneously and can estimate interaction with other recognized objects within a glance (with certain boundaries conditions). In computer vision this is a non-straightforward task to implement. In this study the main objective for the system to localize LED matrices in the captured camera images.

4-1 Detection Algorithm

The detection of objects is done in a number of stages;

1. **Capture Images**
   First a set of images is captured, a left and a right images, using which a depth map is determined (using the ZED camera toolbox).

2. **Prefilter Images**
   Optionally the images are prefiltered using a LaB-filter in order to highlight the LED matrix features.

3. **Detect LED matrices**
   Apply the LED matrix detector to images. This is be done in two different approaches:

   (a) **Global detection**
   Apply the detector to complete images (two images of 1280 × 720 pixels).

   (b) **Local detection**
   Apply the detector to partions of images. The complete image is divided in subimages with a predefined overlap. The detector is run on all these subimages.

4. **Cluster Results**
   Cluster the found LED matrices in order to remove possible duplicates in one images and if applicable between subimages.

5. **Match Clusters**
   Match the found clusters between left and right images.

6. **Determine Depth**
   Will be discussed in chapter 5.

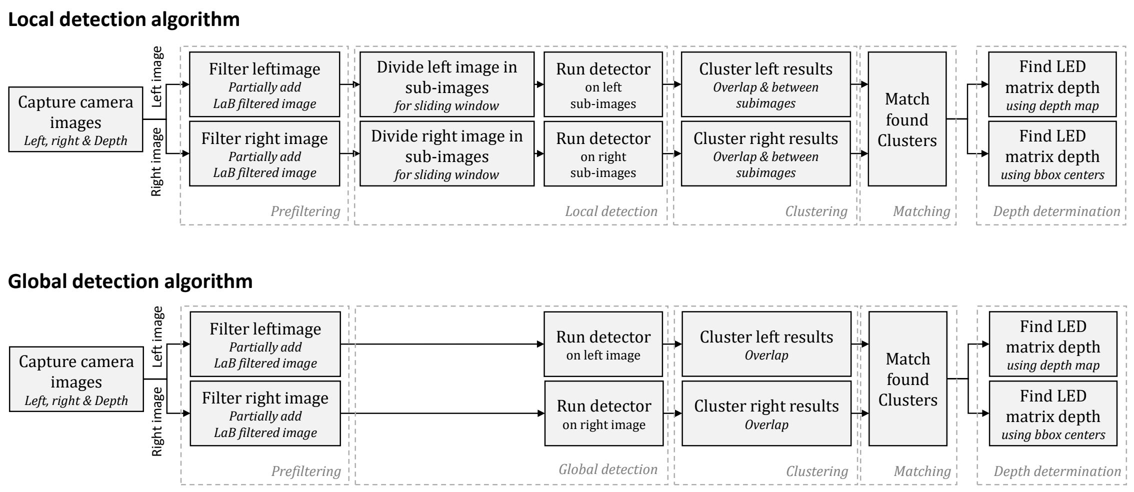

Local detection algorithm
Capture camera images Left, right & Depth
Left image
Right image
Filter leftimage Partially add LaB filtered image
Filter right image Partially add LaB filtered image
Divide left image in sub-images for sliding window
Divide right image in sub-images for sliding window
Run detector on left sub-images
Run detector on right sub-images
Cluster left results Overlap & between subimages
Cluster right results Overlap & between subimages
Match found Clusters
Find LED matrix depth using depth map
Find LED matrix depth using bbox centers
Prefiltering
Local detection
Clustering
Matching
Depth determination
Global detection algorithm
Capture camera images Left, right & Depth
Left image
Right image
Filter leftimage Partially add LaB filtered image
Filter right image Partially add LaB filtered image
Run detector on left image
Run detector on right image
Cluster left results Overlap
Cluster right results Overlap
Match found Clusters
Find LED matrix depth using depth map
Find LED matrix depth using bbox centers
Prefiltering
Global detection
Clustering
Matching
Depth determination

Figure 4-1: Detection and depth measurement algorithm.

4-2 Preprocessing

In order to find an object, such as the LED matrix, in the image, it can be useful to preprocess an image. Preprocessing operations used in the trainingset and using the live camera feed include croppping and image filtering.

4-2-1 Cropping

In the image set a large amount of images are cropped to a size of 224×224, the input size of some later discussed neural networks, around the LED matrix. When cropping the images special care should be taken to not have the object in standard positions, as a detector might be trained to have a unintentional preference to such positions. As without a working detector there is no viable way to automatize this cropping, this tends to be time consuming task, especially for larger data sets.

4-2-2 LaB Filter

In order to highlight the LED matrix in an image, an image filter can be used. The filtering is done in CIE 1976 L*a*b* color space[15], wherein the L* approximated the light intensity in the image, $a*$ the green-red range and $b*$ the blue-yellow. The colors of interest in the LED matrix are mainly white and red[1]. In order to not lose too much context information a fraction of the filtered image is added to a fraction of the original image, the sum of these fraction should equal one. Image fig. 4-2 shows the process and an illustrative example of a LaB filter application in this context.

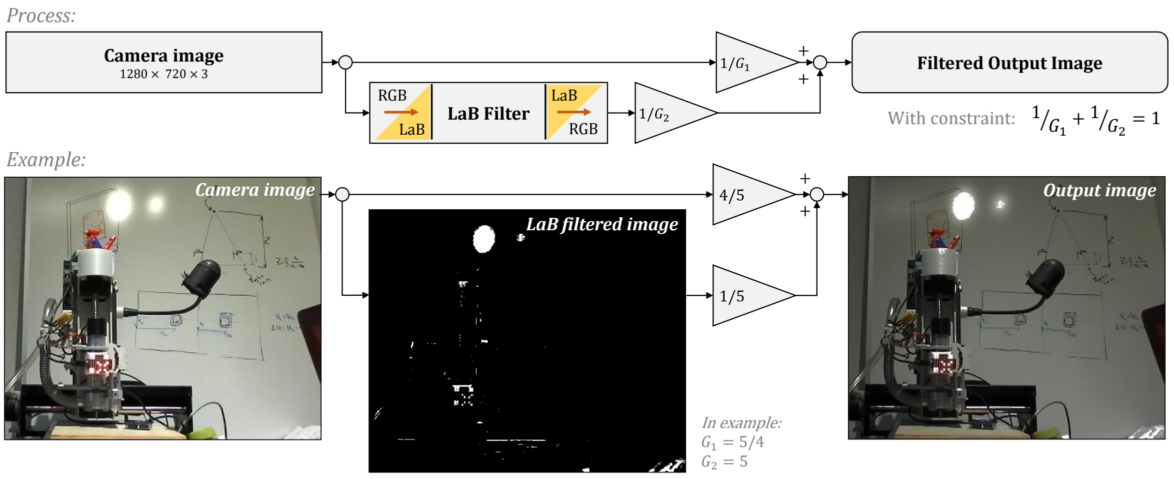


Figure 4-2: LaB filtering procedure.

[1]For the design of the filter in this case the Matlab Computer Vision: Color Thresholding toolbox was used

4-3 Introduction Neural Networks

Humans can process complex tasks, such as pattern recognition, reasoning and visual perception, in a efficient way, learning from examples and adapting from mistakes. In order to have an artificial system do similar tasks neural networks were designed.

In biological neurons, shown in fig. 4-3, some signals $\{x_1, ..., x_n\}$ enter the synapses and only if these signals are higher than a voltage threshold, the neuron fires, in the form of a current spike.

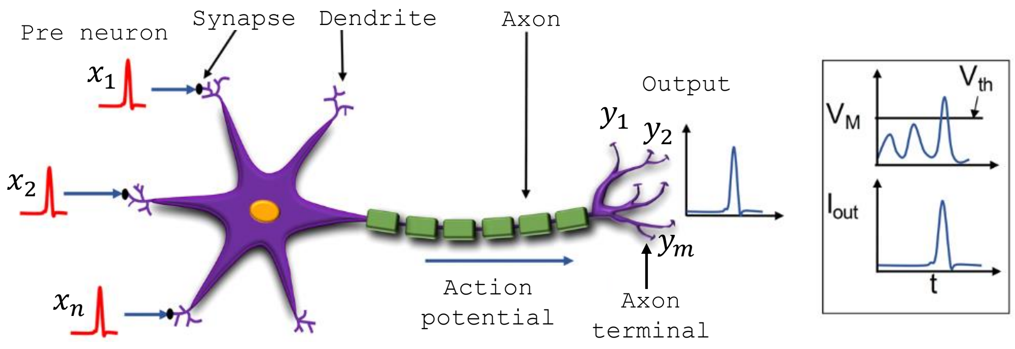
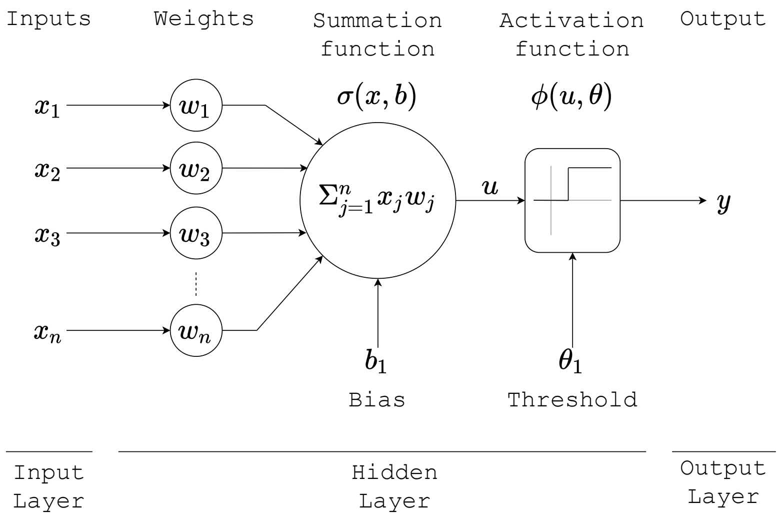


Figure 4-3: Example of a biological neuron[2].

Artificial neural networks that mimic the biological counterparts, refered to as Hebbian Networks, are considered inefficient[16]. Instead models as shown in fig. 4-4 are applied. The output of this neuron can be decribed as in eq. (4-1). An example of an activation function, a thresholding function, is shown in eq. (4-2).



$$y = \phi\left(b_1 + \Sigma_{j=1}^n w_j x_j\right) \quad (4\text{-}1)$$

$$\phi(u,\theta) = \begin{cases} 1 \text{ if } u \geq \theta \\ 0 \text{ if } u < \theta \end{cases} \quad (4\text{-}2)$$

Figure 4-4: Example of an artificial neuron.

[2]Image courtesy of Kalita, H., Krishnaprasad, A., Choudhary, N., Das, S., Dev, D., Ding, Y., ... & Roy, T. (2019). Artificial Neuron using Vertical MoS 2/Graphene Threshold Switching Memristors. Scientific reports, 9(1), 1-8.

The weights $\{w_1, w_2, ..., w_n\}$, biases and thresholds $\{b, \theta\}$ are optimized in a 'learning' process, as will be discussed later in this chapter. Combining multiple neurons in a configuration of choice in a network can yield a Multi Layered Artificial Neural Network (ANN) as shown in fig. 4-5. The shown network is fully connected, but this is no given constraint. Each connection is assigned a weight, $w_i/$. Neurons can have multiple in and output connections as shown in the image. When two or more hidden layers exist the ANN is refered to as a Deep Neural Network (DNN).

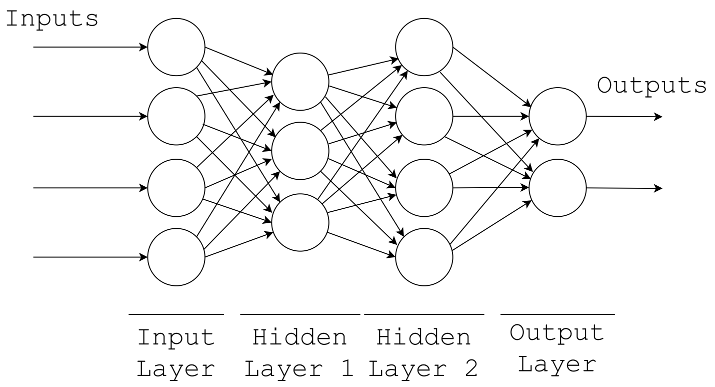


Figure 4-5: Example of a Multi Layered Artificial Neural Network (ANN).

The *propagation function* of a neuron, in fig. 4-4 refered to as u, calculates the input to a neuron from the output of the previous neurons and optional biases.

Neurons are organized in layers, wherein neurons typically only interact with other neurons in directly adjacent layers. The first layer, which interacts with external input, is referred to as the input layer, and the layer which produces the output is referred to as the output layer. Networks with connections in the same or to previous layers are referred to as *recurrent networks.*

In a typical classification problem a networks input can be an image, and the network purpose is to determine whether the image contains one or multiple instances of the classes which the network is designed to classify.

4-4 Introduction to Detectors

Modern detection systems repurpose classifiers in order to perform detection; to detect an object a classifier for this object is taken for evaluation at different locations and scales in a test image. Some systems such as Deformable Parts Models (DPM) use an approach based on a sliding window, where the classifiers are run at an even interval. Other approaches like R-CNN resize patches of the image in order to detect regions of interest [17], [18].

4-5 Detector Design

In order to design and train a suitable detector there are a few prerequisites, which will be discussed in this section. A summary is shown below:

1. **Training Data**
   In order to train a network to recognize LED matrices a dataset of prelabelled images is required to teach the network what a 'LED matrix' is, section 4-5-1 discusses this further.

2. **Network Layout**
   The structure of the network is of course very important to the functionality of the detector. This network layout is discussed in section 4-5-2.

3. **Training Capacity**
   Training a network is computationally intensive task, running it on the platform itself could be a time demanding task with reduced performance.

4. **Selection of Optimizer and Hyperparameters**
   Training networks is done using optimization algorithms, which are discussed in section 4-5-3. The working of these algorithms depend on Hyperparameters, which include the maximum amount of training cycles, the learning rate and similar parameters. The used hyperparameters will be discussed together with the results.

4-5-1 Training Data

In order to train the network a database of around 2500 annotated images was created[3], where in each image a Region-of-Interest (ROI) was noted. A limitation of the used software toolboxes is that all ROIs are square and non rotatable. The training data generation set-up discussed in section 3-4 and fig. 3-16 was used to create a large portion of the dataset. Other training data was generated based on scenarios which are not common in the setup in order to diversify the data set. Examples of setting included images in front of a window, aquarium and next to reflective surfaces, all with changing lighting conditions. Furthermore data was added with filtered images in order to provide a more abstract understanding of the object.

[3]The Matlab image labelling app was used, however serious limitations and quirks were found, so if given a choice a different labeller would be advisable. For more information and the complete dataset, please check my Github repository `https://github.com/SimonvanderMarel/datasetLEDMatrixDetector`.

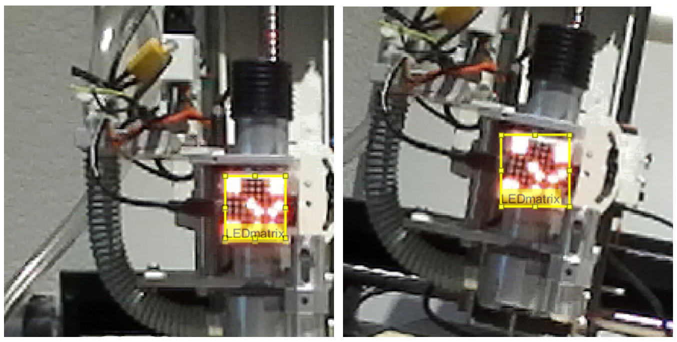


Figure 4-6: Examples of labeled training images, if the object to be labeled is rotated, it is still required to apply a non rotated ROI, as shown in the right image. The shown images are cropped to 224×224 pixels and filtered.

The quality of the training dataset determines to a great extent the quality of the detector, there a several factors influencing the quality of the trainingsdata and therefore the quality of the detector.

- **Quantity**
  A large amount of images in the dataset is paramount to train a high performance detector.
- **Diversity**
  The diverseness of the data is influencing the ability of the detector to work in more than one specific environment, angle, lighting etc. If one would make a large dataset of images taken in similar settings the detector would probably only work in those exact circumstances.
- **Representativeness**
  However the dataset should be diverse, it should represent the use case. If one has a large very diverse dataset which includes little data representing the situation wherein the detector is going to be applied, the trained detector will probably preform worse than one trained on a less diverse but more representative set. Also constraints should be placed on artifacts which are placed around the object of interest, certain mounting constructs or brackets might intentionally be included in trainingsdata. This is also true for the object location in the images, a unintentional preference for a certain location might be trained if in the data the object is often shown in this location in the image.
- **Label Accuracy**
  The quality of the labels in the training data is a factor in the performance of the detector; if the detector is trained on inaccurate labelled data, the detector will detect similarly inaccurate.

4-5-2 Network Layout

In order to train an image detector one could design an deep learning network, but a more efficient method could be to retrain an existing network. A large amount of pre-trained network are available, such as the ones available in Matlab shown in fig. 4-7. The majority of these is trained on subsets of the ImageNet database [19], containing over a million images, with over a thousand different classification categories, such as mugs, keyboards and different animals.

These pre-trained networks have different characteristics, which influence the suitability for different applications. These characteristics include accuracy, size and speed, where one in general has to evaluate a suited trade-off between these properties. The image shown in fig. 4-7 shows these characteristics .

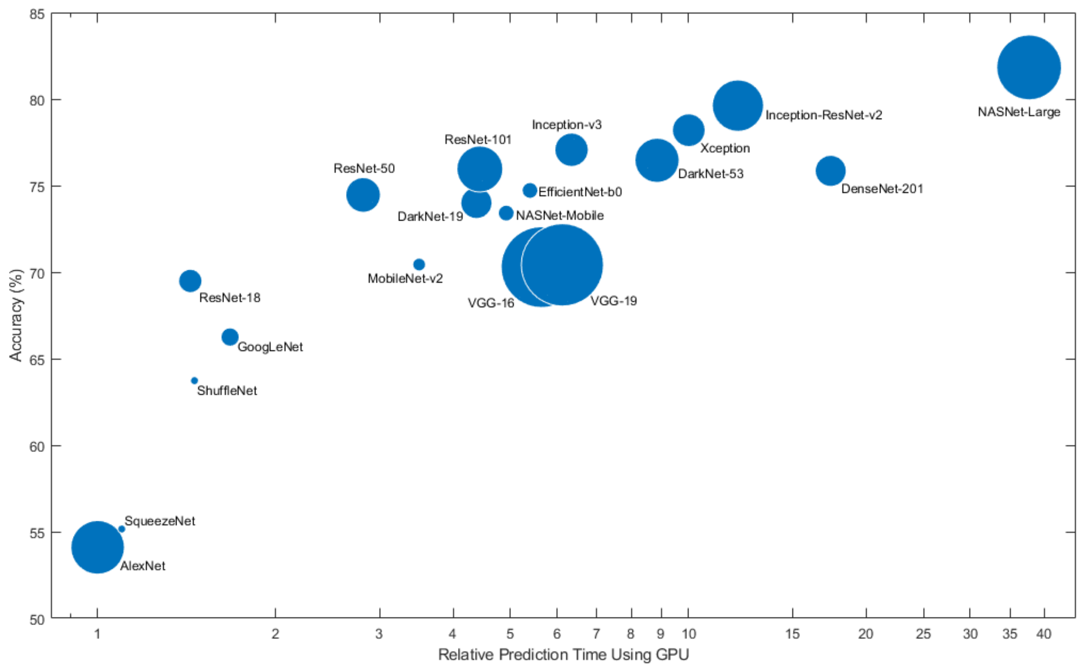


Figure 4-7: List of pre-trained networks available in Matlab [20].

For the experiment several pre-trained networks will be evaluated:

- SqueezeNet
- DarkNet19
- DarkNet53

SqueezeNet

SqueezeNet is a deep neural network released in 2016 [21]. It is developed for embedded applications, with a focus on a smaller size and speed instead of accuracy, achieving AlexNet accuracy with fifty times fewer parameters. The net layout is shown in fig. 4-9.

DarkNet19

DarkNet19 is a 19 layered net based on the Extraction net (by the same author[22], based on GoogLeNet), merged with parts of publications such as Network in Network, Inception and Batch Normalization, prosoped in the publication of YOLOv2[22], [23]. The net layout is shown fig. 4-10.

Figure 4-8: Darknet logo[22].

DarkNet53

DarkNet53 is a 53 layered net, improved on DarkNet19[22], by the same author, proposed in the publication on YOLOv3[24]. The net layout, shown in fig. 4-11, has more layers then the DarkNet19 variant and performs generally more accurate, but significantly slower.

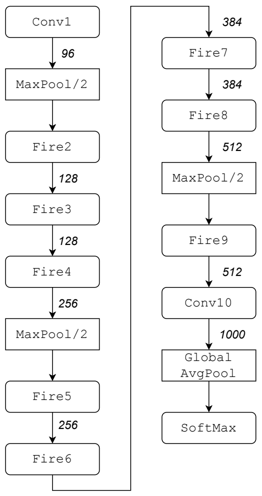
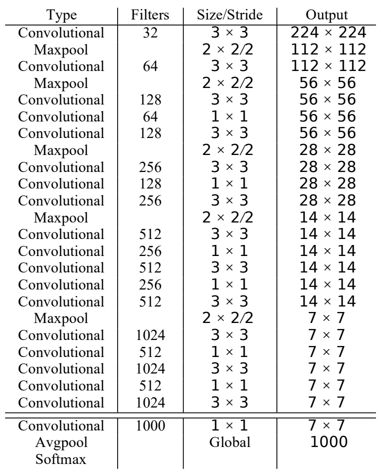
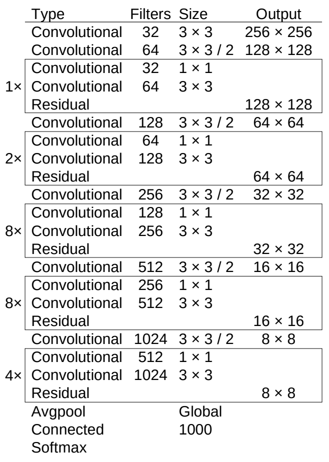


Figure 4-9: SqueezeNet layout[21].



Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Figure 4-10: Darknet19 layout[23].



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4-11: Darknet53 layout[24].

4-5-3 Optimization Algorithms

After a pretrained network is selected for repurposing as detector a selection needs to made for the optization algorithm. There is a wide range of optimizers available for the teaching of deep neural network. For the sake of conciseness only the algorithms available in the used Matlab environment will be discussed in this subsection, and are listed below.

- Stochatics Gradient Descend Method (SGDM)
- Root Mean Square Propagation (RMSprop)
- Adaptive Moment Estimation (ADAM)

SGDM

The Stochastic Gradient Descent Method (SGDM) performs a parameter update for each training iteration

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t, x^{(i)}, j^{(j)}) \tag{4-3}$$

Herein the variables θ_t and θ_{t+1} denote the parameter vectors of the current and next iteration steps, η denotes the learning rate, ∇_θ the gradient with respect to θ and $J(\theta_t, x^{(i)}, j^{(j)})$ denotes a cost function.

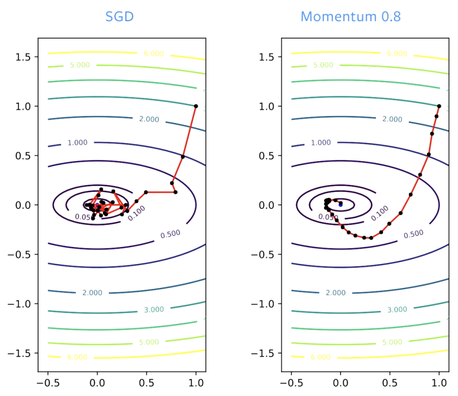


Figure 4-12: Example of SGDM behaviour without and with momentum [25]

SGDM has problems with curvature behaviour around local minima [26] [27]. In order to improve this behaviour [28] proposed a momentum term, ν_t, which takes into account the previous search directions. As shown in eq. (4-4) the parameters are updated by a combination of the current gradient and a fraction γ of the previous update term. Typical values of γ are $\gamma = 0.9$.

$$\begin{aligned} \nu_t &= \gamma \nu_{t-1} + \eta \cdot \nabla(\theta) J(\theta_t, x^{(i)}, j^{(j)}) \\ \theta_{t+1} &= \theta_t - \nu_t \end{aligned} \tag{4-4}$$

RMSprop

Root Mean Square Propagation (RMSprop) is an unpublished optimization designed for neural networks, proposed by Geoff Hinton in his lecture slides, designed to overcome the shortcomings of the AdaGrad algorithm. The algorithm is shown in eq. (4-5)[4]. The term m_t is used to scale the learning rate in each dimension, based on the gradient in each dimension and previous values.

$$\begin{aligned} m_t &= \beta m_{t-1} + (1-\beta)\nabla_\theta J(\theta_t) \odot \nabla_\theta J(\theta_t) \\ \theta_{t+1} &= \theta_t - \eta \nabla_\theta J(\theta_t) \oslash \sqrt{m_t + \epsilon} \end{aligned} \tag{4-5}$$

In order to manage the size of m_t RMS prop uses exponential decay. RMSprop decays the contribution of older gradients at each iteration in order to prevent m_t becoming to large that it stops learning. β denotes a decay factor, η the learning rate and ϵ a smoothing factor to prevent division by zero. Typical values, proposed by Hinton, are $\beta = 0.9, \eta = 0.001$ and $\epsilon = 10^{-8}$. In practice RMSprop was the preferred optimizer until the introduction of ADAM, which outperforms the former.

ADAM

Adaptive Moment Estimation (ADAM)[29] computes adaptive learning rates for each parameter. Adam keeps a exponentially decaying average of squared past gradients, $\nu(t)$, similarly to RMSprop, but also keeps an exponentially decaying average of past gradients $m(t)$, calculated as shown in eq. (4-6). The former acts as the first moment, the mean, where the latter acts the second moment, the uncentered variance[30]. Therefrom the methods name is derived.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t) \\ \nu_t &= \beta_2 \nu_{t-1} + (1-\beta_2)\nabla_\theta J(\theta_t) \odot \nabla_\theta J(\theta_t) \end{aligned} \tag{4-6}$$

Since both moments are initialized as zero vectors, [29] observed that they are biased towards zero, especially in the first iteration steps and when the decay rates are small, in other words, $\beta_1 \to 1, \beta_2 \to 1$. In order to counteract these biases, the bias corrected moment estimates are calculated as shown in eq. (4-7).

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1} \\ \hat{\nu}_t &= \frac{\nu_t}{1-\beta_2} \end{aligned} \tag{4-7}$$

Using these estimates the parameters are updated as shown in eq. (4-8).

$$\theta_{t+1} = \theta_t + \eta \ \hat{m}_t \oslash \sqrt{\hat{\nu}_t + \epsilon} \tag{4-8}$$

Typical hyperparameter values, proposed by the authors, are $\beta_1 = 0.9, \beta_2 = 0.999, \eta = 0.001$ and $\epsilon = 10^{-8}$. RMSprop is still used in instances wherein ADAM tends to osculate around an optimal solution.

[4]Herein $\odot$ refers to the Hadamard product, or element wise multiplication, and $\oslash$ refers to Hadamard division, or element wise division.

4-5-4 Detector Training

This subsection describes an algorithm to train a detector on the earlier discussed labeled training data. A dataset is prepared by first and optionally filtering images are as will be discussed in section 4-2-2, afterwards these images will cropped if the designed detector will be used in combination with a sliding window, as will be discussed in section 6-5. After this the images are annotated, in this case by hand and the dataset is exported for training.

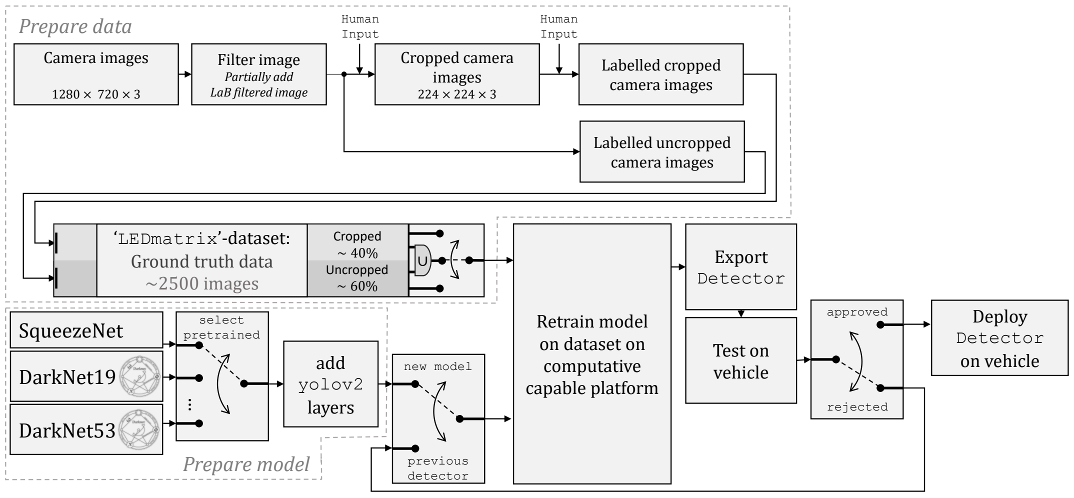


Figure 4-13: Visualization of the detector training algorithm.

The training algorithm is shown in fig. 4-13 and described using pseudocode in algorithm 1 and a brief summary of the specifications of the computer used for training is shown in table 4-1.

Algorithm 1 Train Detector

```
 1: procedure PREPARE DATA
 2:     Take batch of camera images
 3:     Select images for dataset
 4:     (Optional) Apply filters
 5:     (Optional) Crop camera images
 6:     Annotate images
 7:     Export Groundtruth-file
 8: procedure TRAIN DETECTOR ON DATA
 9:     Prepare model and add yolov2 layers
10:     Select optimizer and hyperparameters
11:     (Re)train model on dataset
12:     Export Detector
```

Part	Description
CPU	Intel Core i5-4570 CPU @ 3.20 GHz
GPU	NVIDIA GeForce GTX 1070ti 8 GB
Installed RAM	16 GB

Table 4-1: Brief overview of the used training computer hardware. It should be noted that the used training computer was a bit underpowered for this purpose, leading to relative small batch sizes in training.

4-6 YOLO

The approach YOLO uses is simpler in comparison to previously discussed methods. Multiple bounding boxes and their class probabilities are predicted simultaneously by a single convolutional network. YOLO is generally trained on full images, as opposed to cropped images around the object of interest, and optimizes detection performance directly. This section will first discuss the original YOLO publication, before expanding to YOLOv2.

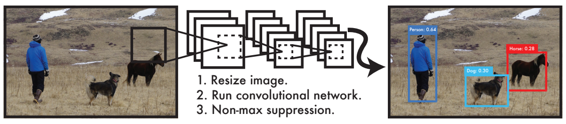


Figure 4-14: The YOLO detection model[17].

4-6-1 Grid Cell

YOLO divides an input image into a $S \times S$ grid. Each grid cell predicts only one object and a fixed number, later referred to as B, of bounding boxes. For example, in image fig. 4-15 the yellow marked grid cell proposes an object 'person', whose center falls in this cell. This grid cell then proposes two bounding boxes, both the blue squares, with the centers marked as blue dots. This 'one object per grid cell'-rule limits how close objects can be detected, as illustrated in fig. 4-16. There are more people in the left bottom corner than there are detections, as there are no more grid cells available.

.

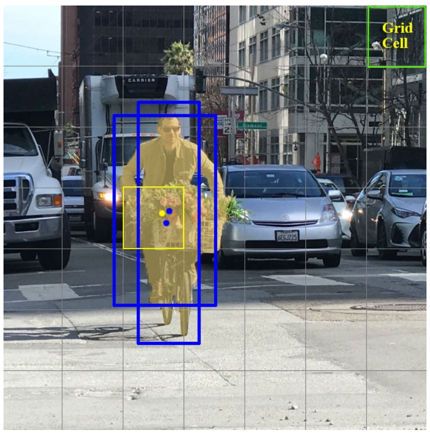


Figure 4-15: An example of detection using YOLO[5].

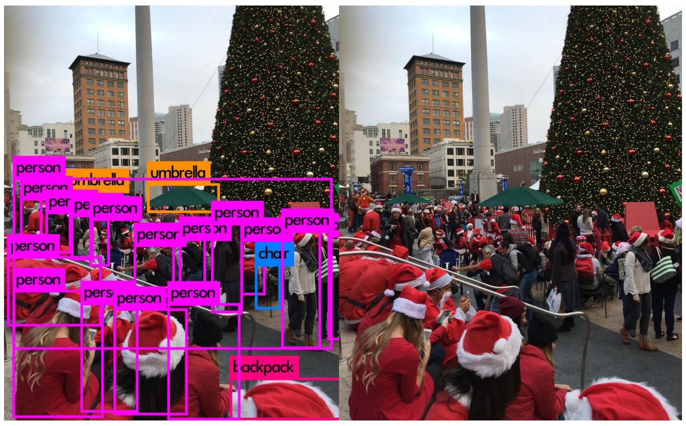


Figure 4-16: An example of detection using YOLO with multiple overlapping objects[6].

[5] Courtesy of Jonathan Hui, https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088, accesed on 2021-04-27

[6]See previous footnote

4-6-2 Unified Detection

As discussed earlier each cell predicts B bounding boxes and a confidence score for each of these boxes. This confidence score indicates the likelihood of the box containing an object, the so called objectness, and the accuracy of the bounding box, formally defined as in eq. (4-10). If there is no object in the discussed grid cell, all its confidence scores should be zero, otherwise this score should equal the Intersection over Union (IoU) between the predicted box, noted as A_d, and the ground truth, noted as A_g[17]. The IoU is defined as shown in eq. (4-9), and is visualized in fig. 4-18.

$$\text{IoU} = \frac{A_g \cap A_f}{A_g \cup A_f} \tag{4-9}$$

Each bounding box contains five elements, $\{x, y, u, v, s\}$, the first four elements being the location, width and height of the bounding box, and the fifth element being a box confidence score. The $\{x, y\}$ are offsets relative to bounds of the corresponding grid cell and $\{u, v\}$ elements are normalized to image size. There all five bounding box elements range between zero and one. Each cell predicts C conditional class probabilities, which are conditioned on a grid cell containing an object. Regardless of the number of bounding boxes B there is only one set of class probabilities C. For each class the class confidence score is calculated, as shown in eq. (4-12). YOLO prediction is a $S \times S \times B \cdot N + C$ tensor, where N denotes the number of classes. The algorithm is visualized in fig. 4-17.

$$\text{box confidence score} \equiv \Pr(\text{Object}) * \text{IoU}^{\text{truth}}_{\text{pred}} \tag{4-10}$$

$$\text{conditional class probability} \equiv \Pr(\text{Class}_i|\text{Object}) \tag{4-11}$$

$$\text{class condifidence scores} = \Pr(\text{Class}_i) * \text{IoU}^{\text{truth}}_{\text{pred}} \tag{4-12}$$

$$\begin{aligned}
&\text{Herein:} & & \\
&\Pr(\text{Object}) & & \text{probability predicted box contains an object} \\
&\text{IoU}^{\text{truth}}_{\text{pred}} & & \text{IoU between predicted box and ground truth} \\
&\Pr(\text{Class}_i|\text{Object}) & & \text{given an object is present, the probability} \\
& & & \text{this object belongs to Class}_i \\
&\Pr(\text{Class}_i) & & \text{probability an object belongs to Class}_i
\end{aligned}$$

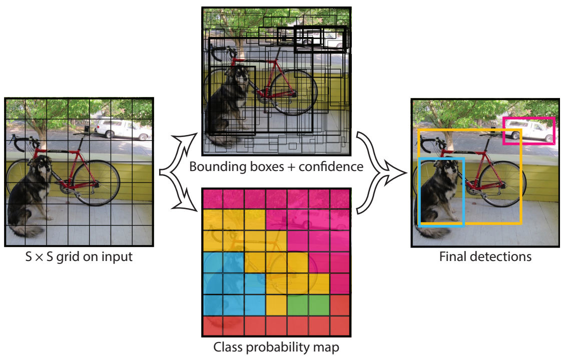


Figure 4-17: The YOLO detection algorithm visualized[17].

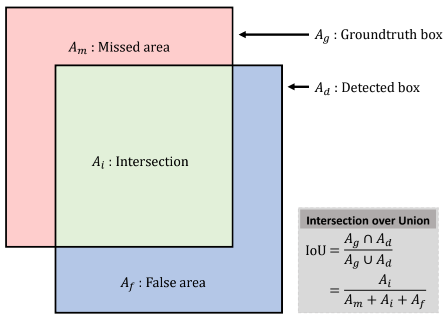


Figure 4-18: The intersection over union (IoU) visualized.

4-6-3 Training

The YOLO algorithm predicts multiple bounding boxes per grid cell. In order to determine the loss for the true positive match, only one grid cell is responsible for the detection. In order to achieve this the grid cell with the highest IoU with the ground truth is selected. This strategy leads to specialization between the bounding box predictions as each prediction gets better at predicting certain object sizes and aspect ratios.

Independent of the base network used, the final layer should output both the bounding box coordinates and the class probability scores of these boxes. The output of the model is optimized using sum-squared error. This error does not suit the purpose ideally however, as it weighs localization equally with missclassifications. Another problem is that in images not all grid cells contain objects. As the confidence of these cells should be zero, this can overpower the gradient from cells containing objects, often resulting in model instability, causing the training to diverge early on[17].

In order to counteract these problems the loss from bounding box coordinate predictions are increased and the loss from confidence predictions for empty grid cells is decreased. Two parameters are introduced to accomplish this, λ_{coord} and λ_{noobj}, both of which typically are defined as five.

The following loss function, shown in eq. (4-13), is used in the optimization, as discussed in section 4-5-3[17].

$$\begin{aligned} L =\ & \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \\ & \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{u_i} - \sqrt{\hat{u}_i} \right)^2 + \left(\sqrt{v_i} - \sqrt{\hat{v}_i} \right)^2 \right] + \\ & \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \\ & \lambda_{\text{noobj}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 + \\ & \sum_{i=0}^{s^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{4-13}$$

Herein $\mathbb{1}_{i}^{\text{obj}}$ denotes whether an object appears in grid cell i. Element $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the j^{th} bounding box in grid cell i is responsible for its prediction. Therefore the loss function only penalizes a classification error when an object is present in that grid cell. The bounding box coordinate error is only penalized if a predictor is responsible for the ground truth box, in other words, if it has the largest IoU[17].

4-6-4 Improvements in YOLOv2

Compared to similar detection systems YOLO suffers a number of shortcomings. These include a significant number of localization errors compared to Fast R-CNN[18] and a relative low recall compared to region-proposal based methods[18]. In contrast to the trend of combining multiple networks or training larger networks, the author of YOLOv2 proposed to simplify the network and make the representation easier to learn. In order to accomplish this a number of ideas of existing work and new concepts are implemented in YOLOv2, of which the mayor improvements are listed below:

- **Batch Normalization**
  Implementing batch normalization yield improvments in convergence and eliminates the need for other forms of regularization[18], [31].
- **Convolutional with Anchor Boxes**
  In contrast to YOLO, which predicts the bounding box coordinates directly using fully connected layers on top of the convolutional feature extractor, YOLOv2 uses anchor boxes to predict bounding boxes, similar to Faster R-CNN[18], [32]. These anchor boxes can be either designed manually or determined automatically from input data. When determining the anchor boxes from input data k-means clustering is used on the training data, in order to automatically extract suitable priors. The used distance metrics proposed by the author is as shown in eq. (4-14)[18].

$$d\left(\text{box}, \text{centroid}\right) = 1 - \text{IoU}\left(\text{box}, \text{centroid}\right) \tag{4-14}$$

- **Direct Location Prediction**
  Implementing anchor boxes in YOLO leads to the problem of model instability, especially in early iterations, mainly due to the predictions of $\{x, y\}$ locations of the box. Typically in region proposal networks the network predicts values $\{t_x, t_y\}$ and the coordinates $\{x, y\}$ are determined as shown in eq. (4-15). These values $\{t_x, t_y\}$ shift the box proportional to the size of the anchor box. As this shift is unconstrained the box can be placed anywhere in the image, regardless of which grid cell proposed the box.

$$\begin{aligned} x &= (t_x \cdot u_a) - x_a \\ y &= (t_y \cdot v_a) - y_a \end{aligned} \tag{4-15}$$

  YOLOv2 implements a different approach, where instead of predicting offsets, location coordinates relative to the location of the grid cell are determined. This casts the value in bounds between zero and one, similar to the approach in YOLO. The network predicts five coordinates for each bounding box, $\{t_x, t_y, t_u, t_v, t_o\}$. The offsets to the top left corner of the image are denoted by $\{c_x, c_y\}$ and the anchor box has a width and height $\{p_u, p_v\}$. Using this notation the prediction $\{b_x, b_y, b_u, b_v, b_0\}$ corresponds to the set of equations shown in eq. (4-16)[18]. As the location prediction now is constrained, the parametrization should be easier to learn, making the network more stable[18].

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_u &= p_u e^{t_u} \\ b_v &= p_v e^{t_v} \\ b_o &= \text{Pr}\left(\text{Object}\right) * \text{IoU}\left(b, \text{Object}\right) = \sigma(t_o) \end{aligned} \tag{4-16}$$

4-6-5 Advantages and Disadvantages

Compared to other approaches for object detection YOLOv2 has several benefits:

- **Processing Time**
  Compared to traditional methods the YOLOv2 algorithm is extremely fast.
- **Contextual Information**
  Since the YOLO algorithm uses the entire image, the algorithm has implicitly encoded the context of classes and their appearance. This results in fewer background errors, compared to methods which do not use this information[17], [18].
- **Generalizable Representations**
  By training YOLO on natural images of an object, it learns a general representation of the object. When applying a YOLO detector, trained on natural images, on artwork YOLO outperforms methods such as DPM and R-CNN[17], [18].

Disadvantages of YOLO include:

- **Struggles to Detect Small Objects**
  YOLO struggles to detect small objects, especially when these appear in groups.
- **Struggles to Detect Nearby Objects**
  Since each grid can only propose two bounding boxes YOLO struggles to detect nearby objects[17].
- **Struggles to Detect Objects in Unusual Aspect Ratios**
  Since YOLO learns bounding boxes aspect ratios from data it has trouble detecting objects in unusual aspect ratios[17].
- **YOLOv2 Struggles with Scaling of Object**
  The version used, YOLOv2, struggles with scaling of objects, which is less problematic in newer versions.

Chapter 5

Depth Estimation

Since the location of the LED matrix can be determined in pixel locations using the information in the previous sections a logical next step would be to determine the depth of the LED matrix from the camera. Building on the information described in chapter 2, this section discusses two ways to determine this distance, one using a simple disparity calculation using the bounding boxes centers and an other using the camera toolboxes depth information.

5-1 Depth Estimation using Bounding Box Centers

In order to measure the depth of the LED matrix a simple disparity calculation, as discussed in section 2-3-1, can be used using the centers of the bounding boxes,

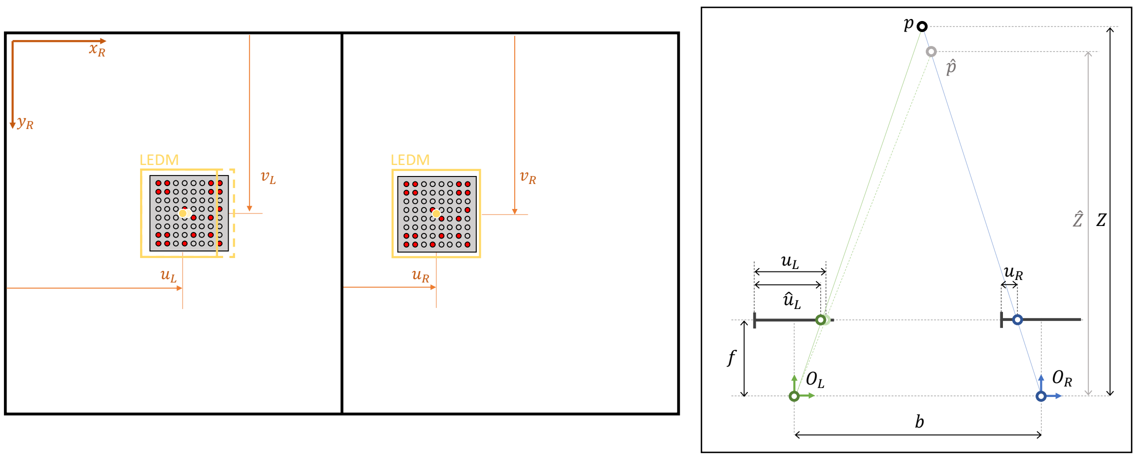


Figure 5-1: Calculating depth using the disparity.

Using a disparity calculation the depth can be calculated, as shown below:

$$Z = f\frac{b}{u_L - u_R} = f\frac{b}{\Delta u} \tag{5-1}$$

Measurement Errors

Because the bounding boxes are estimates using neural networks, these are in most cases not covering the LED matrix in a consistent manner. Imperfect bounding box estimation leads in these depth estimation cases cause an offset in the disparity, Δu and therefore to an offset in depth, Z, as shown in fig. 5-1. Since the accuracy of the depth measurement method based on the centers of the bounding boxes is directly dependent on the accuracy of the detection, this information should be used to determine how much one should trust the results. The example shown in fig. 5-2 shows examples of imperfect estimated led matrices and the influence on the used center.

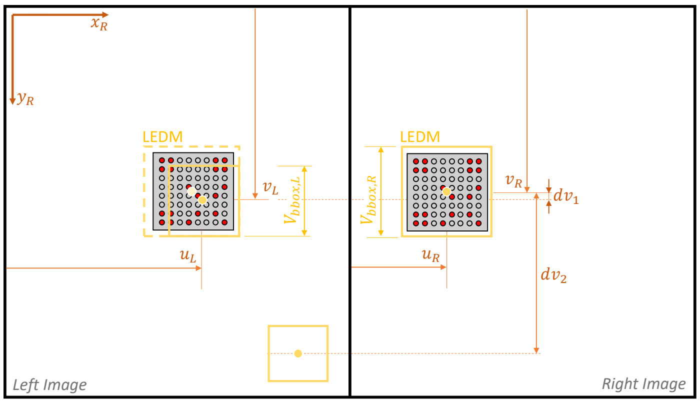


Figure 5-2: Error types in depth estimation using disparity and bounding boxes.

Misalignment w.r.t. left image:

$$C_{m,L} = \frac{|v_L - v_R|}{v_{\text{bb,L}}} = \frac{|dv|}{v_{\text{bb,L}}} \tag{5-2}$$

Size difference w.r.t. left image:

$$C_{s,L} = \frac{|v_{\text{bb,L}} - v_{\text{bb,R}}|}{v_{\text{bb,L}}} \tag{5-3}$$

Since only a simple matching algorithm is used, there is little information available on the likeliness of the match. Information that is available include the sizes of the bounding boxes and the vertical offset of the centers. When the bounding boxes are estimated with different vertical sizes this is an indication that is likely that at least one of the bounding boxes is not estimated correctly. A similar thing can be said about the bounding box centers; since the images are rectified the LED matrices and therefore the bounding boxes should be on the same horizontal line in both images. Combining both confidence reducing properties lead to the equation show in eq. (5-4).

$$C_1 = \left(1 - \max\left(1, \frac{|dv|}{2\ v_{\text{bb,L}}} + \frac{|dv|}{2\ v_{\text{bb,R}}}\right)\right) \cdot \left(1 - \max\left(1, \frac{|v_{\text{bb,L}} - v_{\text{bb,R}}|}{2\ v_{\text{bb,L}}} + \frac{|v_{\text{bb,L}} - v_{\text{bb,R}}|}{2\ v_{\text{bb,R}}}\right)\right) \tag{5-4}$$

Herein:

dv	distance of bbox centers in the vertical direction in pixels
$v_{\text{bb,L}}, v_{\text{bb,R}}$	heights of left and right bboxes in pixels

5-2 Depth Estimation using Neighbouring Pixels

In order to estimate the location of the LED matrix the depth information provided by the ZED camera toolbox can be used, instead of the bounding boxes as discussed before, as that method might prove unreliable. The depth information this toolbox provides is unreliable at the LED matrix itself. Possibly because the LED matrix contains repetitive patterns, which can disturb the stereo matching process. The blooming caused by the LEDs might also influence this matching.

One workaround for this issue is to take not the depth information of the LED matrix pixels themselves, but instead use the neighbouring pixels, provided that the neighbouring pixel lie in the same plane as the LED matrix pixels. In order to enforce this constraint a modified version of the LED matrix assembly is used, as shown in fig. 5-4.

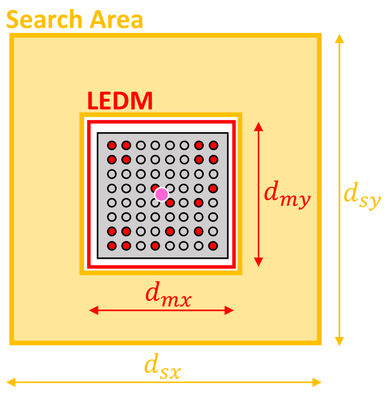


Figure 5-3: Example of a neighbourhood search area around a LED Matrix.

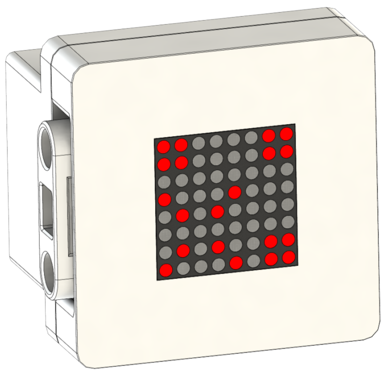


Figure 5-4: A render of the modified LED matrix assembly.

When a LED matrix is detected, a search area is constructed around the bounding box of the LED matrix, as shown in fig. 5-3. The size of the rectangular search area is proportional to the size of the LED matrix bounding box in both axis respectively, where unless specified otherwise $d_s = 1.8 \cdot d_m$ is used.

Determining Depth and Confidence metric

An example of a depth map is shown in fig. 5-5, where the white/greyish areas show useful data and the black data shows bad data. The blue marked area shows the region of interest for the depth measurement.

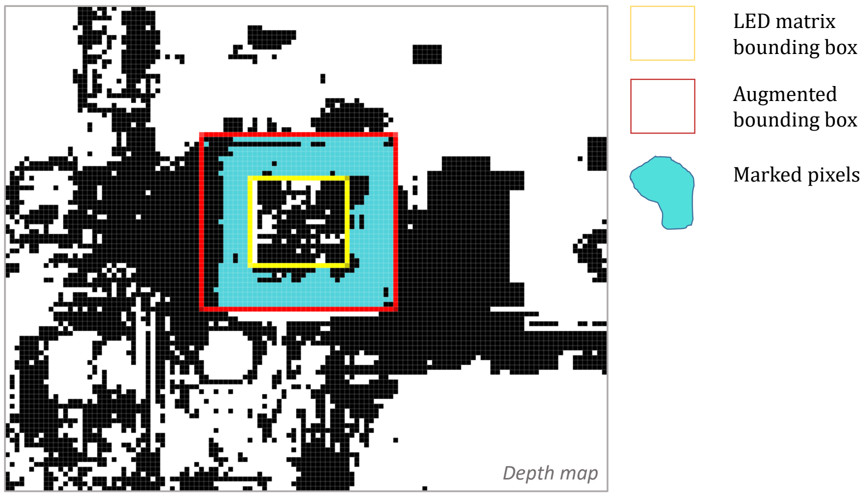


Figure 5-5: Example of usable data in depth map.

A naive approach to determine the depth would be taking the average of all data points, as shown in eq. (5-5). A disadvantage of this method is the sensitivity to bad data, as bad data can skew the mean. A better approach would be taking the median of the useful data since this is approach is less sensitive to noise.

$$Z = \frac{1}{N}\sum_{i=1}^{N} a_i, \qquad N = \text{numel}\left(S_{\text{aug,bbox}} \setminus S_{\text{bbox}}\right), \; a_i \in S_{\text{aug,bbox}} \setminus S_{\text{bbox}} \tag{5-5}$$

$$Z = \text{median}\left(S_{\text{aug,bbox}} \setminus S_{\text{bbox}}\right) \tag{5-6}$$

Herein:

S_{bbox}	set of points in the bbox
$S_{\text{aug, bbox}}$	set of points in the augmented bbox
S_{nodepth}	set of points wherein no depth was determined
numel	number of elements in set

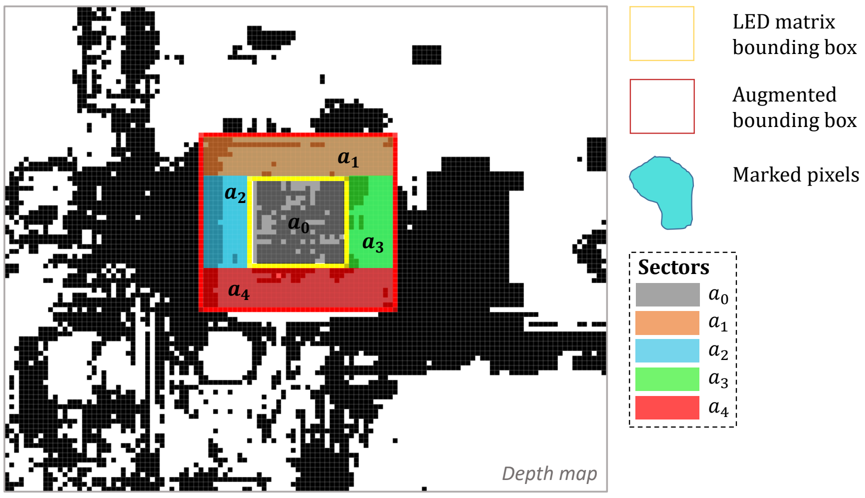


Figure 5-6: Useable data divided in sectors.

A problem with the previous methods is these do not handle rotation of the marker well, as often the amount of useable pixel on one side of the marker is larger, leading to an offset. A solution can be dividing the markers in sectors, as shown as in fig. 5-6, and take the mean of the sector medians. In this way a lack of data on one side does not skew the median or mean.

$$Z = \frac{1}{N_s} \sum_{i=1}^{N_s} \text{median}(a_i) \tag{5-7}$$

One disadvantage of eq. (5-7) is sensitivity to sectors with little data, which increases the sensitivity to noise. Applying a correction factor n_i/n_t per sector, wherein n_i denotes the amount of points in the sector, and n_t denotes the total amount of useful points in $(S_{\text{aug,bbox}} \setminus S_{\text{bbox}}) \cap S_{\text{nodepth}}$.

$$Z = \frac{1}{n_t} \sum_{i=1}^{N_s} (n_i \cdot \text{median}(a_i)) \tag{5-8}$$

As the median of the available usual data is used, one could describe the confidence in as the fraction of usable data. As shown in eq. (5-9) this confidence is determined by dividing the number of elements in the intersection between the sets of points with no data and the data in the region between the augmented bounding box and the LED matrix bounding box by the total amount of points in this region.

$$C_2 = 1 - \frac{\text{numel}\left((S_{\text{aug,bbox}} \setminus S_{\text{bbox}}) \cap S_{\text{nodepth}}\right)}{\text{numel}\left(S_{\text{aug,bbox}} \setminus S_{\text{bbox}}\right)} \tag{5-9}$$

Minimal Reliable Distance

In fig. 5-7 an example is shown of two camera images and the depth maps belonging to those two images. In the first set of images the depth information is clearly unreliable, because the object (the LED matrix) is to close to the camera.

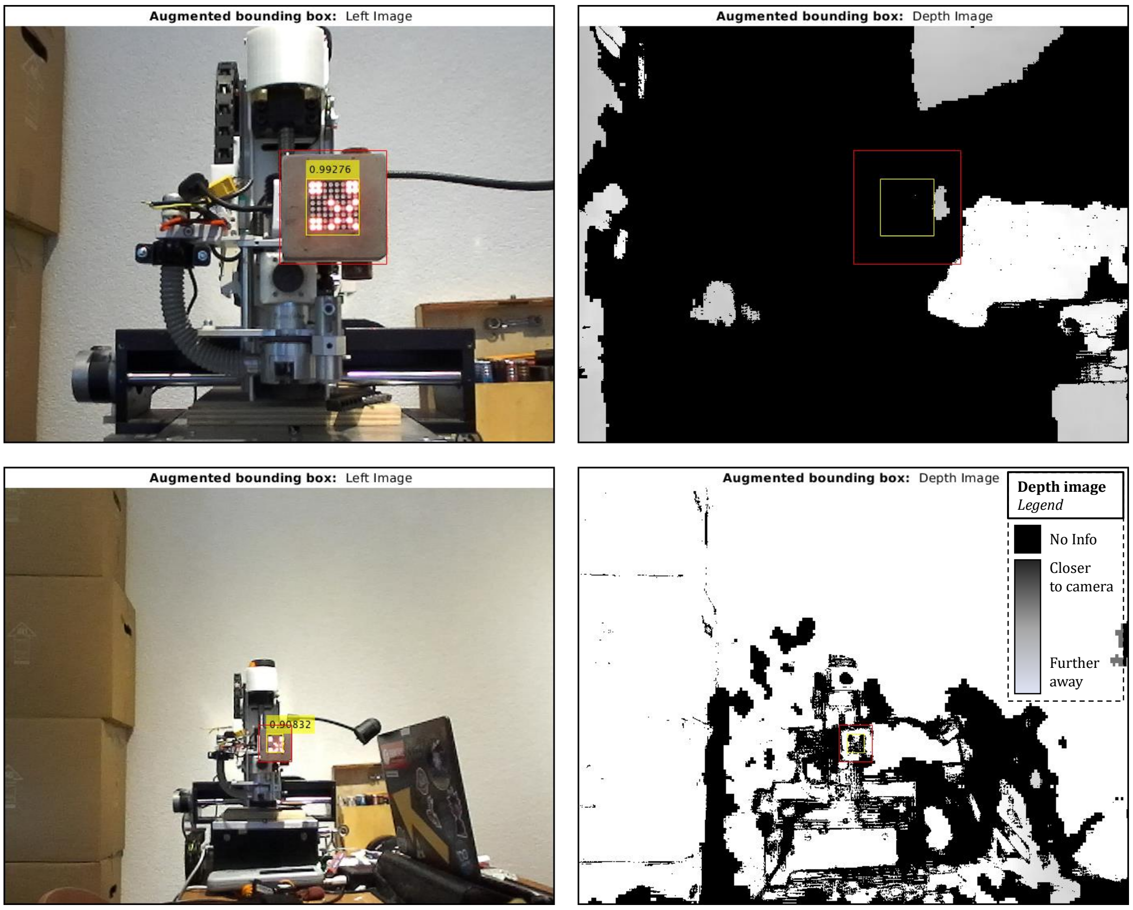
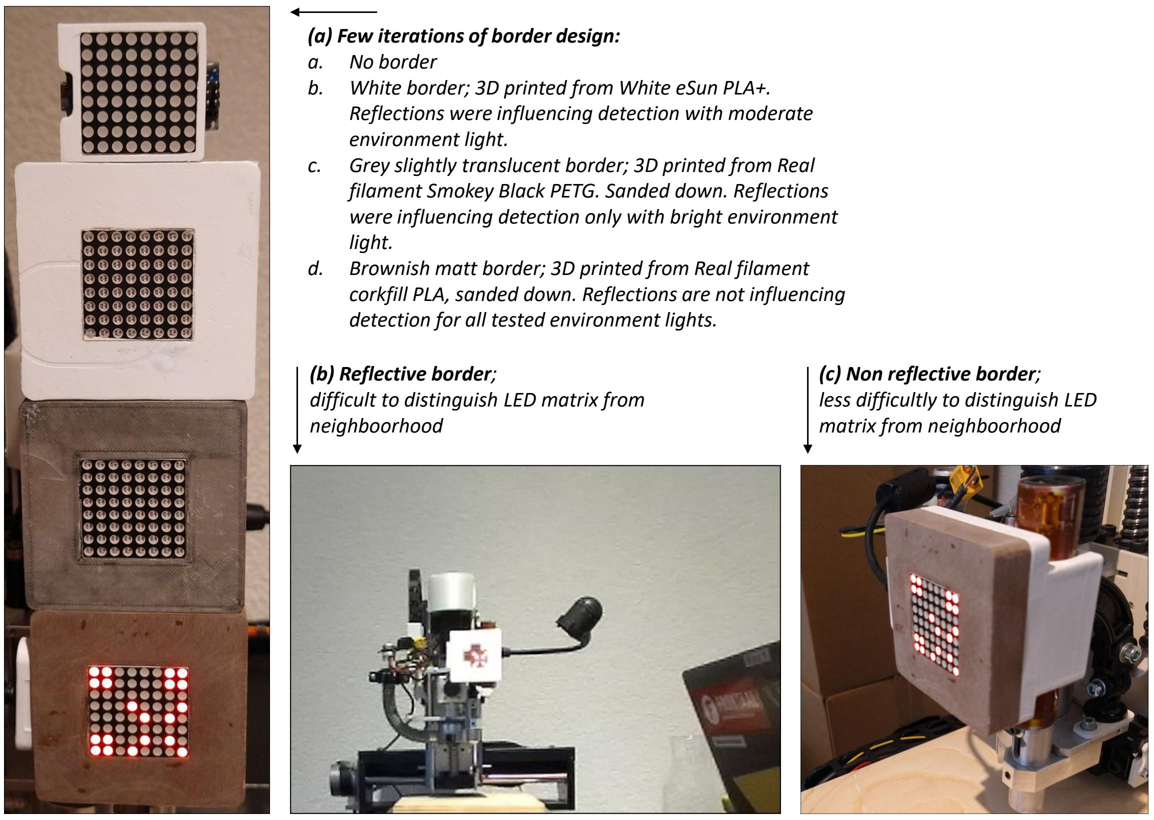

Augmented bounding box: Left Image
0.99276
Augmented bounding box: Depth Image
Augmented bounding box: Left Image
0.90832
Augmented bounding box: Depth Image
Depth image
Legend
No Info
Closer to camera
Further away

Figure 5-7: Example of a neighbourhood search area around a LED Matrix for a close by ($\sim 700\,\mathrm{mm}$) and further away ($\sim 1400\,\mathrm{mm}$) scenario.

Reflectiveness of the Neighbourhood

When the neighbourhood of the LED matrix is reflective, this can influence the distinctiveness of LED matrix on the camera image During the set up of the experiment a number of different iterations of the border were used, some of which are shown in fig. 5-8.



(a) Few iterations of border design:

a. *No border*

b. *White border; 3D printed from White eSun PLA+. Reflections were influencing detection with moderate environment light.*

c. *Grey slightly translucent border; 3D printed from Real filament Smokey Black PETG. Sanded down. Reflections were influencing detection only with bright environment light.*

d. *Brownish matt border; 3D printed from Real filament corkfill PLA, sanded down. Reflections are not influencing detection for all tested environment lights.*

(b) Reflective border; *difficult to distinguish LED matrix from neighboorhood*

(c) Non reflective border; *less difficultly to distinguish LED matrix from neighboorhood*

Figure 5-8: Influence of the material of the neighbourhood of the LED matrix on the reflectiveness.

Under circumstances with a lot of light exposure the best results, using a simple visual inspection on the camera images, were archieved using a 3D printed border of sanded 'corkfill'-PLA, which results in a matte brownish finish as shown in the picture above. For the rest of the experiment this border was used.

Chapter 6

Experiment

In an experiment to test the visual communication one device will design and display a simple pseudo QR code message on a LED matrix. The vehicle will attempt to read and localize this message using the ZED stereo camera. An overview of the protocol is shown in fig. 6-1.

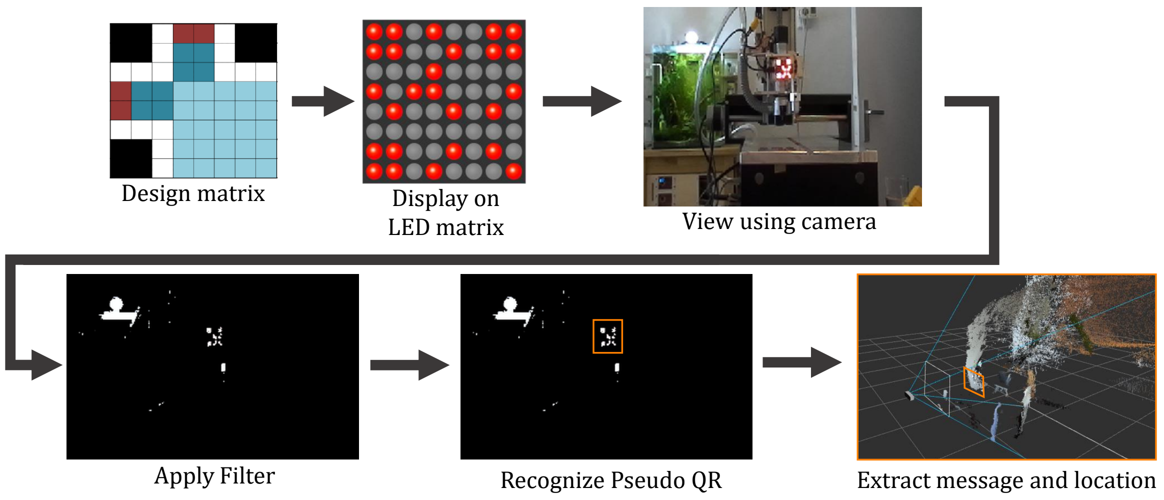


Figure 6-1: LED matrix recognition experiment.

6-1 Setup

In order to test the accuracy and precision of the algorithms, the LED matrix will be moved along one axis. This known movement will be compared with the measured distance. This allows an error to be measured for both algorithms and therefore a comparison can be drawn between those.

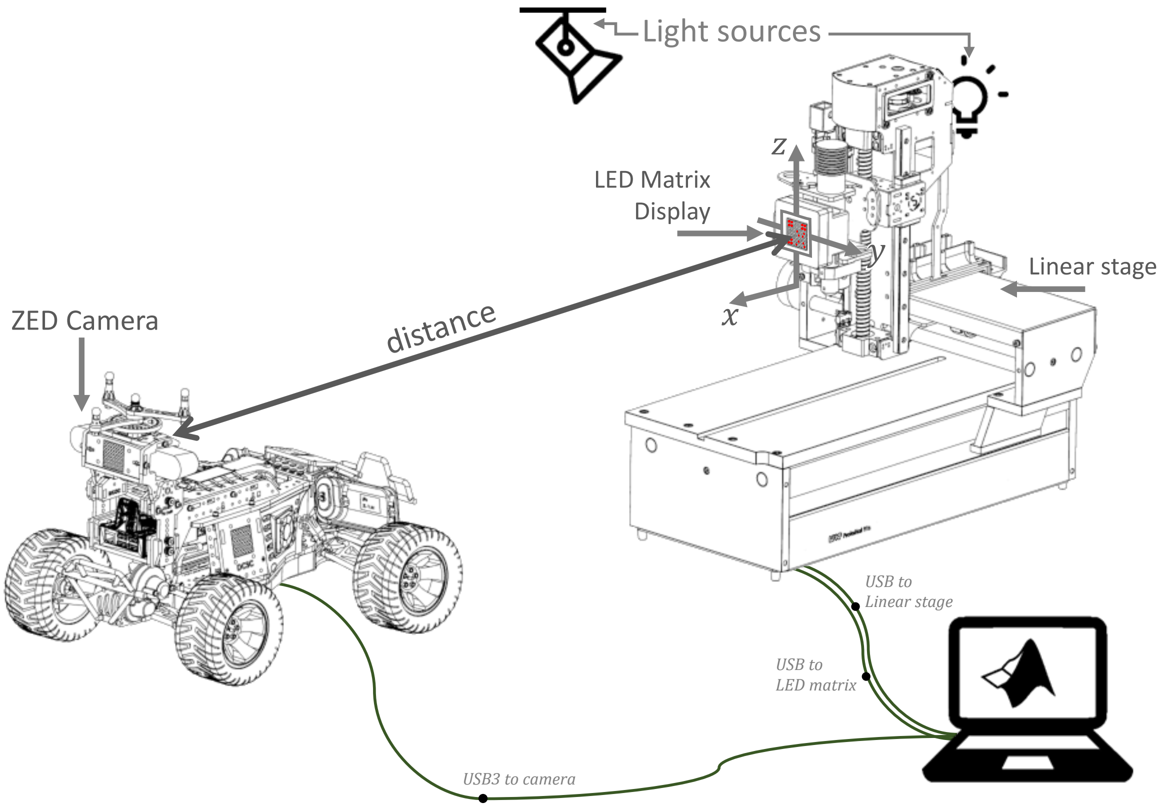


Figure 6-2: LED matrix recognition experimental setup.

The LED matrix is mounted on a linear stage (a modified LPKF ProtoMat 91 PCB mill), which allow the LED matrix to be translated by 330 mm in x direction, 200 mm in y direction and 150 mm in z direction. Only the x direction will be used.

The initial distance will be set up using a one point five meter rod between the starting position and the faceplate of the camera on the vehicle. All other positions will be calculated from that point, introducing a set up error of $\sim 3 - 4$ mm.

Evaluated Methods

The experimental set-up will be used to evaluate several approaches, this includes detectors based on multiple types of pretrained networks (SqueezeNet, DarkNets, etc) and will test local and global approaches. A local approach in this context denotes using a moving sliding window to attempt to detect a LED matrix in a small sub image in the image, whereas a global approach uses the entire image.

Error metrics

The used metrics will be the error and the absolute relative error as show in eqs. (6-1) and (6-2). In these equations $\hat{Z}$ refers to the measured depth, Z refers to the actual depth (plus or minus the earlier discussed setup error).

$$e_1 = \hat{Z} - Z \tag{6-1}$$

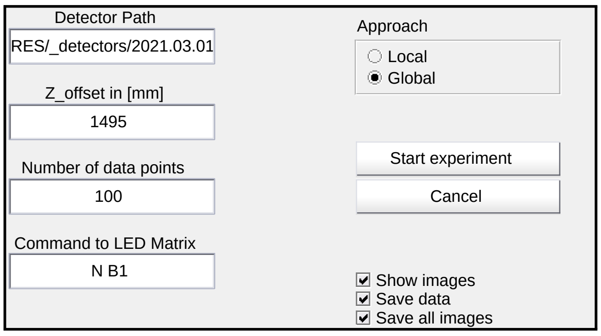
$$e_2 = \frac{|\hat{Z} - Z|}{Z} \tag{6-2}$$

Confidence metrics

As confidence metrics the confidence scores of the YOLO detection algorithms are used, as defined in section 4-6-2. Furthermore the in eqs. (5-4) and (5-9) proposed confidence metrics are used as indications whether their measurements are reliable. However these metrics are not suited to compare both methods as no viable normalization is used.

6-2 Testing Protocol: Online experiment

The experimental protocol is shown in algorithm 2, the protocol is divided in a setup procedure and a main loop procedure. In the experimental setup the user has to input certain experimental parameters in the start-up GUI, as shown in fig. 6-3, afterward dependencies are imported and external hardware is initialised.


Detector Path: RES/_detectors/2021.03.01

Z_offset in [mm]: 1495

Number of data points: 100

Command to LED Matrix: N B1

Approach: Local [✓ Global]

[✓ Show images]
[✓ Save data]
[✓ Save all images]

Figure 6-3: Graphical User Interface (GUI) used at experiment start.

In the main loop the algorithm discussed in section 4-1 and fig. 4-1 is run. The camera is opened, takes a set of pictures, and is closed afterwards[1].

[1]Because of hardware/software limitations the camera can not be run at the same time instance as the YOLOv2 detector

Algorithm 2 Experimental Protocol: Online Experiment

```
 1: procedure SETUP
 2:     User input in GUI                          ▷ Using GUI shown in fig. 6-3
 3:     Import dependencies
          • experiment settings, non user input parameters
          • detector
 4:     System Initialization
          • initialize LED Matrix
          • initialize linear stage
              – home
              – move to initial position
 5:     User: move camera in correct position  ▷ 1.5 m rod between camera and LED matrix
 6:     Generate linear stage trajectory      ▷ Randomized $N$ points between start and end
 7: procedure MAIN LOOP
 8:     for $N$ positions do
 9:         Linear stage: Move to position
10:         Camera
              • open
              • create left, right and depth images
              • close                          ▷ Failing to close requires a system reboot
11:         if approach = Global then
12:             Run Detector on Left and Right images      ▷ One type of Detector per trial
13:         else if approach = Local then
14:             for $k$ subimages do
15:                 Run Detector on Left and Right sub images
16:         Cluster LED matrices         ▷ Overlap & Duplicate LEDm in sliding windows
17:         Match found clusters
18:         Determine depth for matched clusters
              • bounding box centers method
              • depth map method
19:         Plot found results for current iteration               ▷ for real time check
              • Left and right images with all found bboxes
              • Left and depth map images with matched found augmented bboxes
20: procedure FINALISE
21:     Plot experiment results
          • Raw data plot
          • Boxplot error
          • Boxplot absolute error
          • Confidence plot
22:     Save experiment parameters and results to .mat-file
```

6-3 Testing Protocol: Offline experiment

In order be able to validate the detected LED matrices against a human annotated one, a second offline experiment will be done on the same set up. The detectors of interest will be applied to a validation dataset[2] of annotated images. The detected LED matrix bounding boxes will be compared with the groundtruth bounding boxes in order to gain multiple metrics:

- **Intersection over Union**
  The Intersection over Union (IoU) is the metric capturing the overall accuracy of the detector and is used by the author of YOLO.
- **Normalized missed area**
  The missed area of the groundtruth box normalized to the size of the ground truth box.
- **Normalized false area**
  The falsely labeled area outside of the groundtruth box, normalized to the size of the groundtruth box.

Counteracting Hand Labeling Errors

Annotations created by hand are not consistently labeled throughout the datasets. Generally a offset of ± 2 pixels captures all of these hand labeling fluctuations. In order to counteract this influence the groundtruth bounding box is extended with 2 pixels on either side, within this area, as visualized in fig. 6-4, all possible bounding boxes are labeled as groundtruth - allowing for translation, resizing and a combination of these. These groundtruth bounding boxes are then compared with the computer detected bounding box. The best matching groundtruth bounding box is the 'true groundtruth' used in the further calculations.

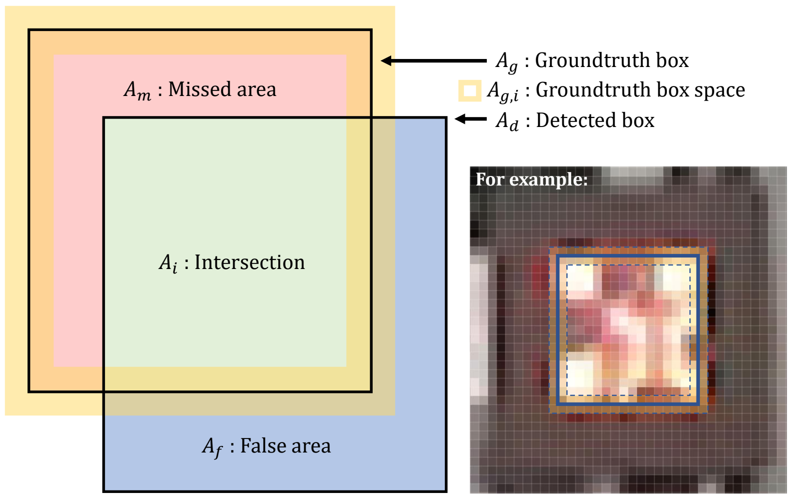


Figure 6-4: The intersection over union (IoU) visualized, with a range of possible groundtruth images. The size of this range depends on the maxium deviation, in the shown example this deviation is one pixel.

[2]Dataset not used in the training of the detectors, and one complete trajectory. A dataset of N datapoints, includes $2N$ annotated images, as both left and right images are captured and annotated.

6-3-1 Creating Validation Dataset

First the validation dataset will be created. This is done by first capturing images and saving data about these images. Afterwards these images are hand annotated. This procedure is described in algorithm 3.

For this experiment a dataset of 400 datapoints, and therefore 800 images, will be created in one trajectory using the set up shown in fig. 6-2.

Algorithm 3 Experimental Protocol: Offline Experiment, Prepare Data

```
 1: procedure CREATE DATASET
 2:     User input in GUI
 3:     System Initialization
          • initialize LED Matrix
          • initialize linear stage
              – home
              – move to initial position
 4:     User: move camera in correct position   ▷ 1.5 m rod between camera and LED matrix
 5:     Generate linear stage trajectory        ▷ Randomized $N$ points between start and end
 6:     for $N$ positions do
 7:         Linear stage: Move to position
 8:         Create new datapoint instance
 9:         Camera
              • open
              • create left, right and depth images
              • close                            ▷ Failing to close requires a system reboot
10:         Add to datapoint instance: position, time, paths to images, camera parameters
11:         Save data
              • in image (.bmp) files: left, right and depth
              • in ‘maindata’ file(.mat): datapoint instance
12: procedure USER ANNOTATE DATASET
13:     Using Matlab ‘Image Annotator’, import all left and right images
14:     for $2N$ images do   ▷ For each datapoint left and right images need to be annotated
15:         User: Zoom to correct view
16:         User: Annotate
17:     Export to ‘Groundtruth’ file (.mat)
18: procedure MATCH ANNOTATIONS DATASET TO DATA
19:     import ‘maindata’ and ‘groundtruth’ files
20:     for $N$ datapoints do
21:         lookfor datapoints’ left and right image filenames in ‘Groundtruth’ file
22:         import annotations to datapoint instance
23:     Update ‘maindata’ file (.mat)
```

6-3-2 Detection Validation

This part of the experiment compares hand annotated LED matrices in the images, the groundtruth, to the detected ones. These LED matrices will be detected using all detectors of interest, in order to compare these detectors with each other.

The validation based on the created dataset is explained in algorithm 4, which follows the same general structure of algorithm 2. The results of this experiment will be shown in section 7-4.

Algorithm 4 Experimental Protocol: Offline Experiment, Validation

```
1: procedure SETUP
2:     Import dependencies
         • experiment settings, non user input parameters
         • detectors
         • validation dataset
3: procedure MAIN LOOP
4:     for $Q$ detectors do
5:         for $N$ positions do
6:             open datapoint from dataset
7:             if approach = Global then
8:                 Run Detector on Left and Right images    ▷ One type of Detector per trial
9:             else if approach = Local then
10:                for $k$ subimages do
11:                    Run Detector on Left and Right sub images
12:            Cluster LED matrices          ▷ Overlap & Duplicate LEDm in sliding windows
13:            Compare LED matrix: groundtruth and detection
                 • IoU
                 • normalized missed area
                 • normalized false area
14:            Add results to datapoint instance
15: procedure FINALISE
16:     Plot experiment results
          • Raw data plot: IoU, missed and false area vs position
17:     Save experiment parameters and results to .mat-file
```

6-4 Training Datasets for Experiment

The LED matrix detectors are trained on subsets of the complete annotated dataset. The general requirements on these kinds of datasets are discussed in section 4-5-1, and the training procedure in section 4-5-4.

Training Data for Local Detectors

During preliminary testing the local detectors, trained on only cropped 224×224 images, showed viable detections on a relatively small dataset, starting from $\sim 200 - 300$ images. During the experiment design this cropped images database was increased to around 1000 annotated images, with a high degree of diversity of environment, lighting, filtering and rotation. A selection of such images is shown in fig. 6-5.

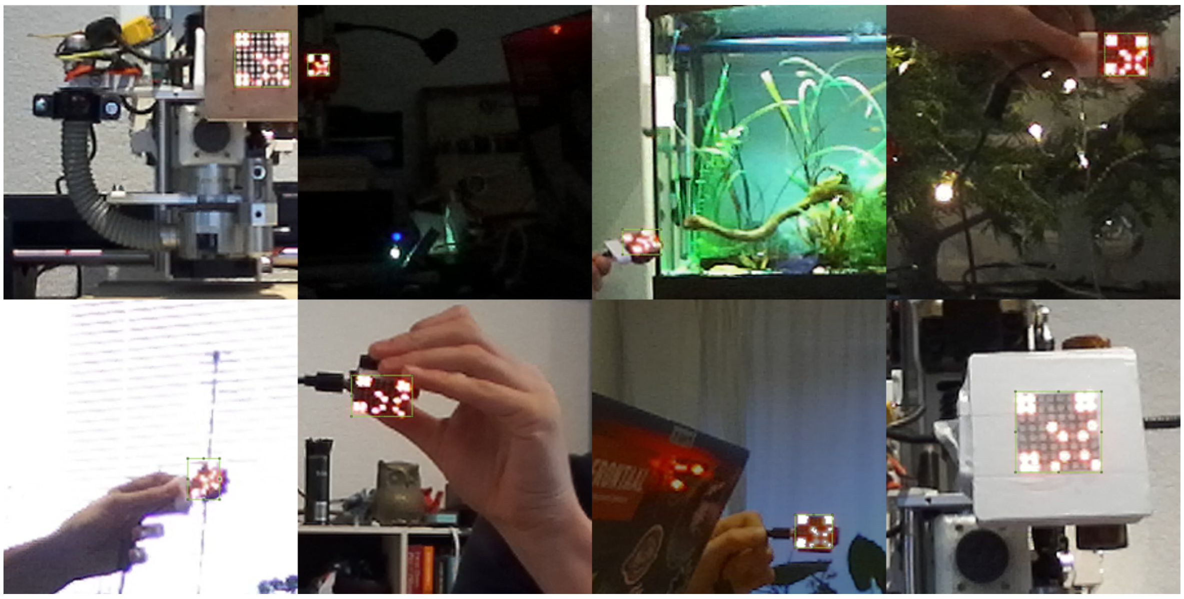


Figure 6-5: Typical cropped and annotated training images. Images are taken in different environments, with different lighting and elements such as reflections.

Training Data for Global Detectors

For the global detectors only entire images were used, as combining data with the cropped images reduced the performance of the trained detectors. The size and aspect ratios are different to the entire images, and while techniques exist to overcome these problems, such as (zero) padding the images, these were not implemented. During preliminary testing the global detector was shown to require orders of magnitudes more of training data in order to perform on a similar level as the local detectors. As annotating by hand is a time consuming task certain sacrifices were made to keep the task approachable in favor of the experiment. The dataset for the global detector was created with focus on the experiment, still using diverse lighting and filtering conditions, but with less orientation and environments changes. By creating the dataset in this way it was hoped to achieve a detector with an abstract description of the object in one orientation. Despite of these compromises it is still expected to have a detector with lower IoU and confidence scores as the desired amount of data required for a detector, which is comparable to the local ones, seems infeasible for this thesis.

6-5 Sliding Window

Since the input size of the network is $224 \times 224(\times 3)$ and the camera image is larger, for the ZED camera (two images of) $720 \times 1280(\times 3)$, this image need to be handled in a certain way. The image is divided in a grid and a sliding window passes each square and attempts to detect a LED matrix in each square.

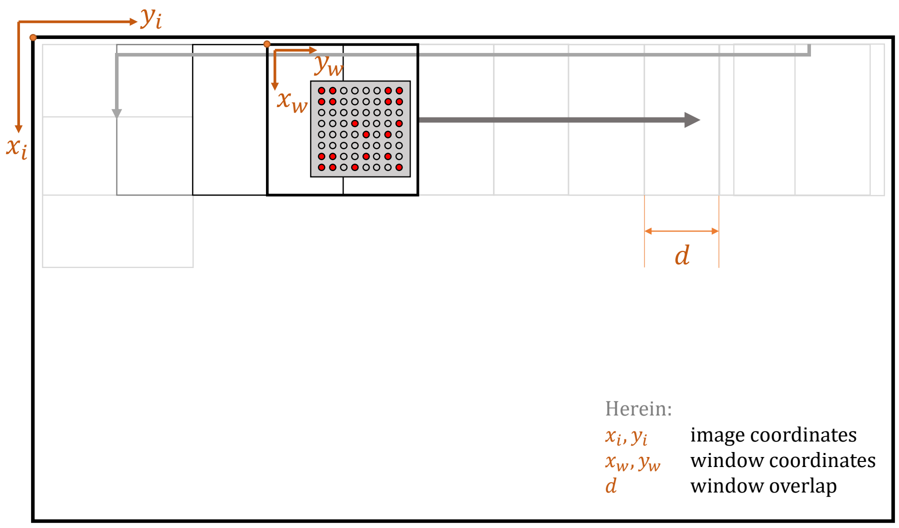


Figure 6-6: Sliding window.

The window overlap is needed for images wherein the LED matrix is in not completely captures in one subimage. This window overlap is a set-up parameter set at $d = 1/2$.

Chapter 7

Results

This chapter discusses the results of the measurement set-up discussed in the previous chapter. The first section discusses the results for the online experiment per detector type, the second section draws a comparison between detectors and the third section notes observations during the online experiment. The last section describes the results for the offline validation experiment.

The complete list of the evaluated detectors[1] for both online and offline experiments is shown below:

- SqueezeNet based YOLOv2 detector
  - Local Detector
- DarkNet19 based YOLOv2 detector
  - Local Detector
- DarkNet53 based YOLOv2 detector
  - Local Detector
  - Global Detector

For the depth map based depth estimation the 'Weighted mean of medians of sectors' approach, introduced in section 5-2, for depth estimation was used, which will be compared to alternatives in section 7-2-2.

[1]A quick reminder of the terminology used. The local detectors use a sliding windows over the images in both directions, with a certain overlap, running a detector on each subimage. The global detector uses the entire image, therefore using the detector only one time per iteration.

7-1 Online Experiment: Results per Detector

This section discusses the results per detector type, split in local (using the sliding window) and global (entire image) detectors.

7-1-1 SqueezeNet Based Yolov2 Detector

The first discussed detector is the SqueezeNet based Yolov2 Detector. This is the detector with the fewest amount of parameters it should be expected to run quicker compared to the DarkNet based counterparts.

Local Detector

The graphs shown in fig. 7-1 shows the online measurement using the setup described in the previous chapter. The left figure shows the error over the distance, wherein positive error denotes an object that is estimated to be further away from the camera than in reality. The right figure shows boxplot of the true error and absolute error for both methods of depth estimation. First of all a negative bias is visible for both the depth map and bounding box based methods. In early testing this bias was shown to be ambient lighting dependent. The depth map based method outperforms the bounding box based method error wise in this distance range as expected. A periodic behaviour seems to occur in the depth map based method. This could possibly happen due to interpolation between pixels, due to the requirement for subpixel accuracy, this will be discussed in section 7-3-2. Based on the confidence graph there are no datapoints visible based on bad data or mismatched LED matrices. A broader comparison between methods will be drawn in section 7-2, which will compare error, computation time and score performance between the tested methods.

Global detector

The global SqueezeNet detector returned no viable results, since in all iterations no LED matrices were detected. The decisions for the hyperparameters will be discussed in section 7-1-3, since this detector was the only global detector to return viable results. Attempts for other global detectors were based on the global DarkNet53 hyperparameters.

7-1-2 DarkNet19 Based Yolov2 Detector

Local Detector

The used detector was trained using the hyperparameters shown in section 7-1-2. The results are shown in fig. 7-2. Again a negative bias is shown for both methods. An outlier is shown between 1.20 m and 1.25 m, as the confidence for this point is very low as well, it should be safe to assume that a mismatch was done between LED matrices in the left and right image. Similar to the local SqueezeNet detector a periodic influence is visible in the depth map based method.

Global Detector

An attempt was made to train Global DarkNet19-448 (input size 448) based YOLOv2 detector, with the same parameters, optimizer and hyperparameters as for the DarkNet53-448 based detector. This attempt did not give viable results as no LED matrices were detected.

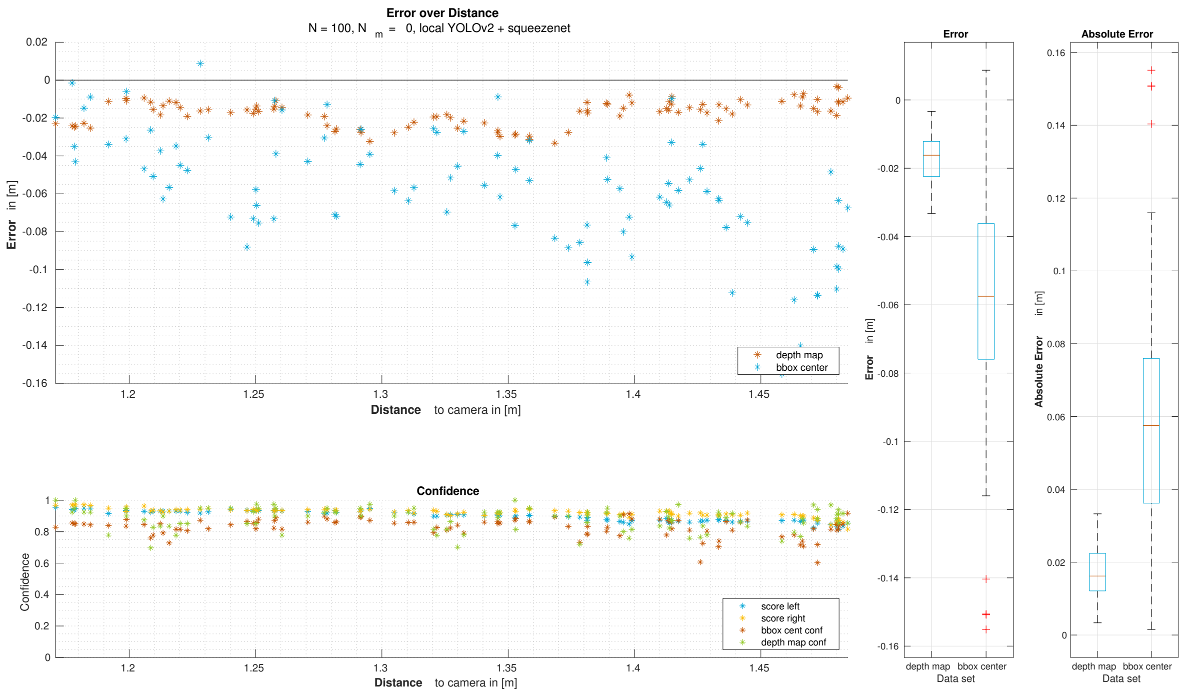


Figure 7-1: Measurement data of the previous discussed online experiment in a known environment using SqueezeNet based detector, N denotes the amount of measurements, N_m denotes the amount of missed frames.

Detector	Parameter	Value	Parameter	Value
Local	Optimizer	ADAM	Input size	$224 \times 224 \times 3$
	Base learning rate	10^{-4}	Anchor boxes	18
	Mini Batch Size	180		
	Max Epochs	300		
Global	Optimizer	RMSprop	Input size	$448 \times 448 \times 3$
	Base learning rate	10^{-5}	Anchor boxes	18
	Mini Batch Size	80		
	Max Epochs	300		

Table 7-1: Used hyperparameters in the training of the Local SqueezeNet detector.

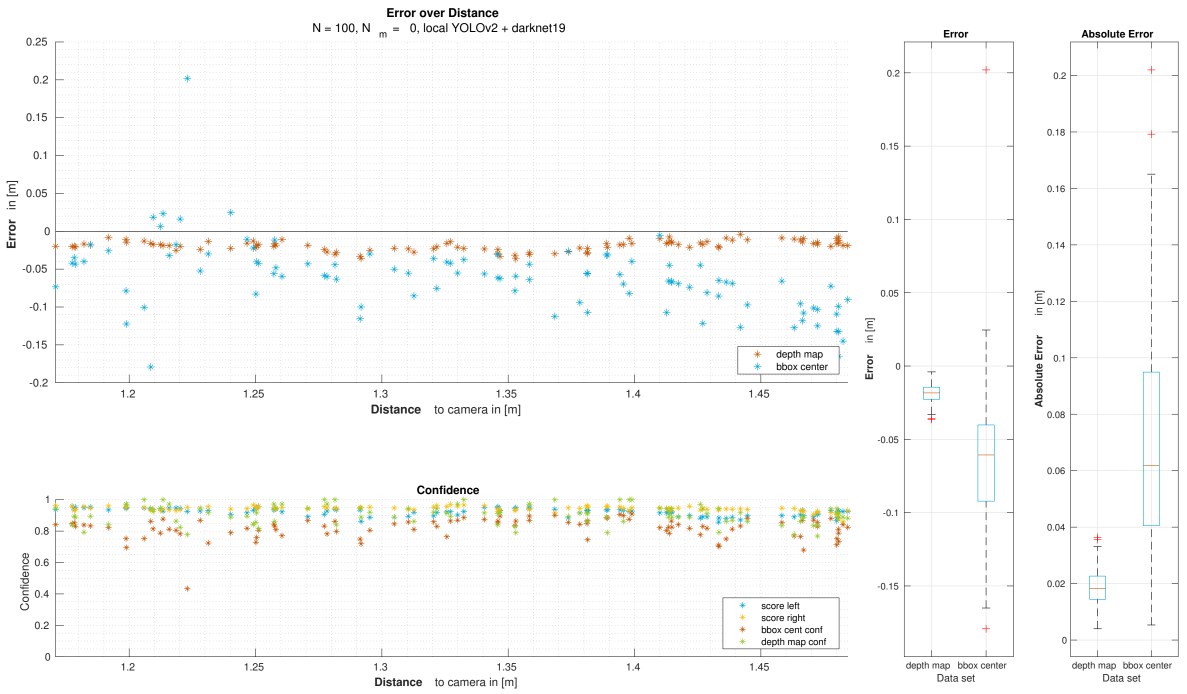


Figure 7-2: Measurement data of the previous discussed experimental setup in a known environment using a local DarkNet19 based detector, N denotes the amount of measurements, N_m denotes the amount of missed frames.

Detector	Parameter	Value	Parameter	Value
Local	Optimizer	ADAM	Input size	$224 \times 224 \times 3$
	Base learning rate	10^{-4}	Anchor boxes	18
	Mini Batch Size	80		
	Max Epochs	300		
Global	Optimizer	RMSprop	Input size	$448 \times 448 \times 3$
	Base learning rate	10^{-5}	Anchor boxes	18
	Mini Batch Size	20		
	Max Epochs	300		

Table 7-2: Used hyperparameters in the training of the DarkNet19 detectors

Detector	Parameter	Value	Parameter	Value
Local	Optimizer	ADAM	Input size	$224 \times 224 \times 3$
	Base learning rate	10^{-4}	Anchor boxes	18
	Mini batch size	40		
	Max Epochs	300		
Global	Optimizer	RMSprop	Input size	$448 \times 448 \times 3$
	Base learning rate	10^{-5}	Anchor boxes	18
	Mini batch size	8		
	Max Epochs	300		

Table 7-3: Used hyperparameters in the training of the DarkNet53 detectors.

7-1-3 DarkNet53 Based Yolov2 Detector

Local DarkNet53 Based Yolov2 Detector

The used detector was trained using the hyperparameters shown in table 7-3. The results are shown in the upper image in fig. 7-3. Again a slightly larger spread it shown towards the 1.5 m for the bounding box based method. A larger periodic influence is visible compared to the previously discussed detectors. A few points with a relative low confidence are shown, but no large influence is shown in the measurement error.

Global DarkNet53-448 Based Yolov2 Detector

In order to have a viable global detector the input size was increased to $448 \times 448 \times 3$ after the standard input size gave no viable results[2]. The resulting detector gives viable results, as shown in fig. 7-3.

The net was trained using the optimizer and hyperparameters shown in table 7-3 using only complete training images. The mini batch parameter is low compared to typical values, as the training computer does not allow a higher value in combination with the rest of the hyperparameters. This makes the training more susceptible to oscillations since using a low number images increases the influence of one difficult to classify image or one ‘bad’ annotation[3]. Since these oscillations were observed during training using the Adaptive Moment Estimation (ADAM) optimizer, a switch was made to the Root Mean Square Propagation (RMSprop) optimizer , since this one is less susceptible to oscillations, as discussed in section 4-5-3. Using RMSprop the net did converge to a suitable detector.

In fig. 7-3 a positive bias is shown for the global detector measurement data. The bias might have changed because this dataset was created under slightly different lighting condition, this will be further discussed in section 7-3-3.

For all found LED matrices low scores were assigned, which is according to expectations and will be further discussed in section 7-2-1.

[2]The increased size DarkNet53-448 was not included in the Matlab software. Although it is possible to change the input size of the one included in Matlab the weights are absent. The increased size net was sourced from the website of the author of [17], and imported using a plugin for Matlab; the "Deep Learning: Darknet Importer" by Kei Otsuka.

[3]However during labeling by hand care was taken to make all annotations as precise and consistent as possible. Some images are easier for the net to classify then others leading to higher scores for one mini batch compared to others using the same detectors. If this effect occurs during training oscillations may occur.

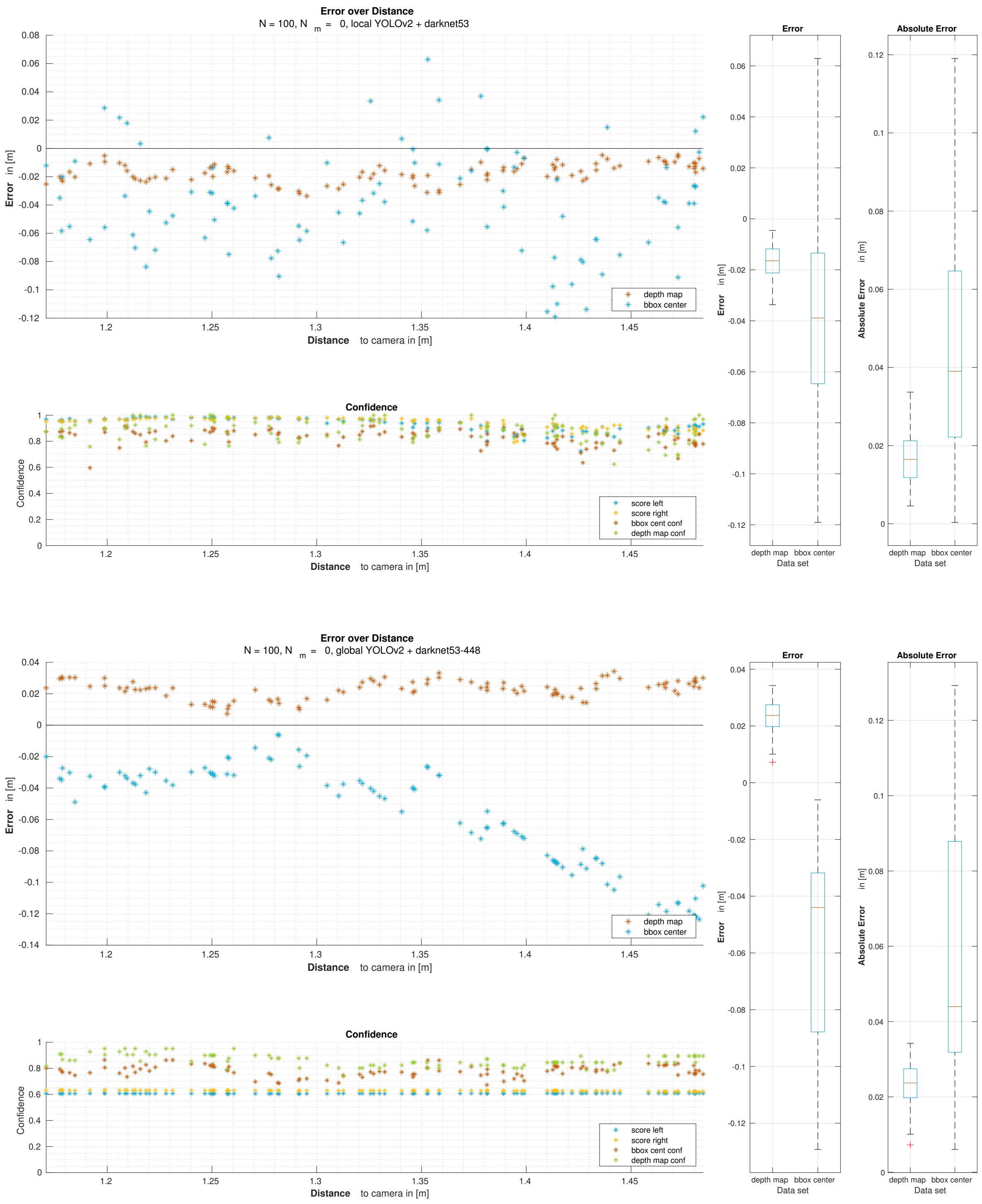


Figure 7-3: Measurement data of the online experiment in a known environment using a local DarkNet53 based detector and a global DarkNet53-448 based detector.

7-2 Online Experiment: Comparison between Methods

In this section comparisons are drawn between different methods; the first subsection discusses performance and computation time between detectors using different base networks. The second subsection draws a comparison between approaches to depth approximation using the depth map, which were introduced in section 5-2.

7-2-1 Base Net Comparison

Performance Comparison

In fig. 7-4 the results are shown for all used approaches for depth measurements, the left image shows the results using the depth map and the right image using the bounding boxes. The first thing to note is the depth map based methods outperforms the bounding box based method in all cases in accuracy. Both sets of data tend to have a bias for both the depth map and bounding box based methods. For the local approaches the error is similar for the depth map based estimation.

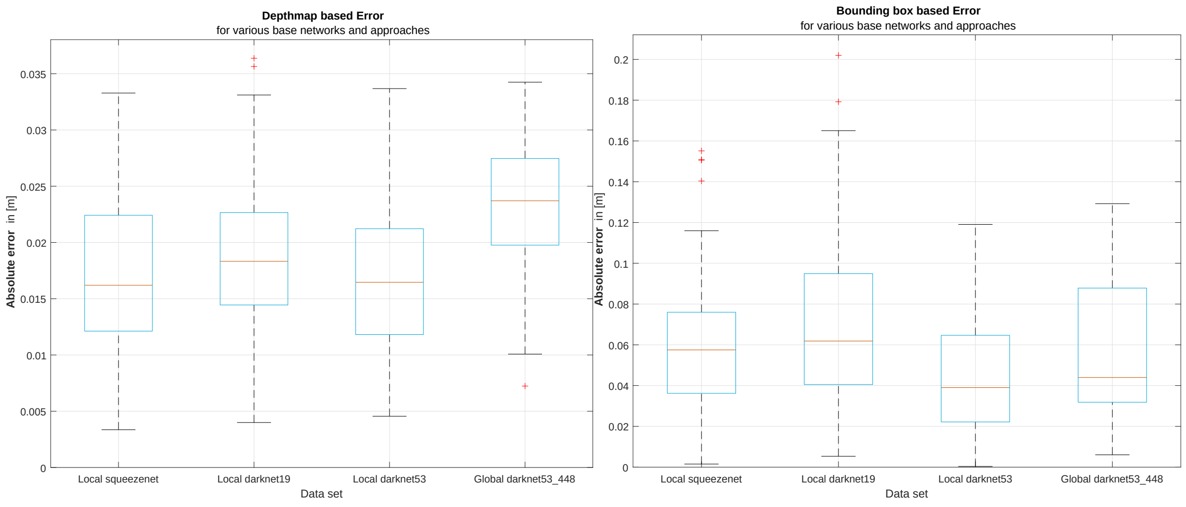


Figure 7-4: Measurement data of the depth map *(left)* and bounding boxes *(right)* based method for the experimental setup in a known environment using various approaches and base networks.

The detectors seems to perform worse with increasing distance for the bounding box based depth estimation, with the global Darknet53-448 detector in particular, as shown in fig. 7-3. The depth map based method does not share this property. Since the detectors tend to have more difficulties with detecting objects further away, this falls in the line of expectations. The DarkNet53 based method has a slightly more accurate estimation for the bounding boxes based approach.

Computation Time Comparison

During the experiment operation times were measured[4]. An overview of mean operation times is shown in table 7-4. The operation that takes most time is the detection, by a wide margin.

	Detection in [ms]	Clustering in [ms]	Matching [ms]	Depth bbox in [ms]	Depth map in [ms]
Local squeezenet	5745.684	0.440	0.159	0.925	0.925
Local darknet19	7593.779	0.406	0.112	0.885	0.885
Local darknet53	12553.591	0.355	0.106	0.809	0.809
Global darknet53-448	266.289	0.550	0.212	2.538	2.538

Table 7-4: Computation times per operation.

The box plot shown in fig. 7-5 shows the detection times for various base nets and approaches. The local methods, using a sliding window, take a significant longer time to detect compared to the global DarkNet53-448 approach. This makes sense as using the global detector the detection network is used one time, while using the local approaches the net is run dozens of times. It can be problematic to apply the local detectors to real world applications because of the significant computation time, as these times are prohibitively high - a vehicle can not have a five second delay in braking, because it possible will have crashed in the meantime.

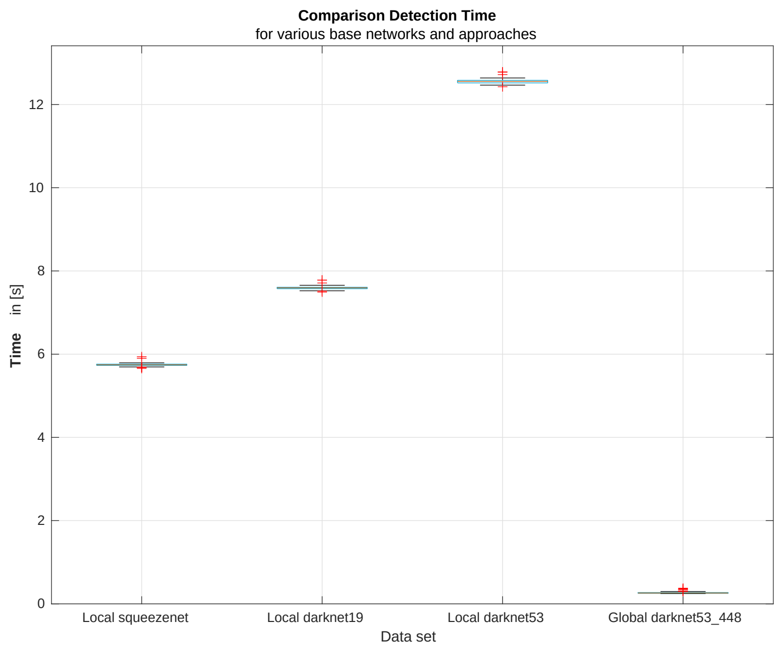


Figure 7-5: Detection times for the experimental setup in a known environment using various approaches and base networks.

It should be noted that this might not reflect well to real life applications, since the unoptimized Matlab code was run on a HP 'Zbook' laptop[5], leaving much room for improvements in this perspective. Improvements in algorithms, such as only scanning a region of interest in the next image, might speed up the local approaches as well by orders of magnitude.

[4]The are measured using Matlab `tic` and `toc` commands.
[5]A laptop with the specifications:*Intel i7, NVIDIA M1200 Quadro GPU 4GB vRAM, 8GB RAM.*

Detection Score Comparison

In table 7-5 and fig. 7-6 the detection scores, the YOLOv2 confidence scores as described in section 4-6-2, are shown. The scores for the local detectors are significantly higher compared to the global detector. These results is in line with the in section 6-4 described expectations.

	Error depthmap in [cm]	Error bbox in [cm]	Score	Detection time in [ms]
Local squeezenet	-1.734	-5.965	0.911	5745.684
Local darknet19	-1.894	-6.234	0.931	7593.779
Local darknet53	-1.708	-3.992	0.932	12553.591
Global darknet53-448	2.291	-5.975	0.618	266.289

Table 7-5: Overview of mean score results.

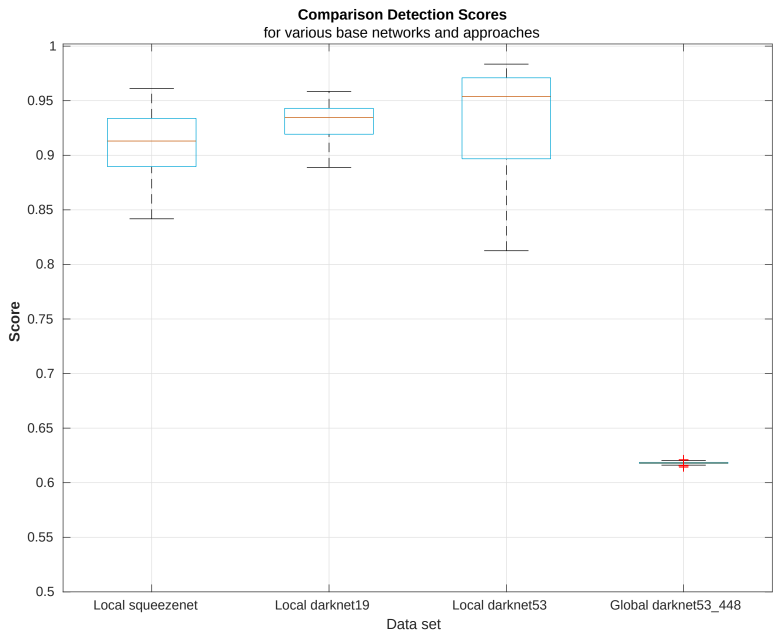


Figure 7-6: Detection scores for the experimental setup in a known environment using various approaches and base networks. Per category 100 data points are used.

7-2-2 Depth Map Measurement Comparison

In order to draw a comparison between the different depth map based algorithms, these algorithms, shown below, were applied on one dataset. In the experiment five sectors are used. These are used in the configuration as shown in fig. 7-7, for 100 measurement points in a distance range between 1180 mm and 1500 mm. The results for all detectors are shown in fig. 7-8. For all datasets the unweighted sector based method performs best. Comparing this unweighted sector based method between approaches (local/global and different nets) yield similar absolute errors with similar variances. The depth estimation from the previous results was based on the 'Weighted mean of medians of sectors' and will stand to improve by switching to the 'Mean of medians of sectors'-approach in further experiments.

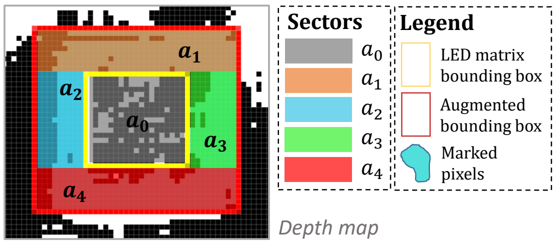


Figure 7-7: Recap of a sector division of the LED matrix marker.

1. **Mean of complete marker area,** as shown in eq. (5-5).
2. **Median of complete marker area,** as shown in eq. (5-6).
3. **Mean of medians of sectors,** as shown in eq. (5-6) and fig. 7-7.
4. **Weighted mean of medians of sectors,** as shown in eq. (5-6) and fig. 7-7.

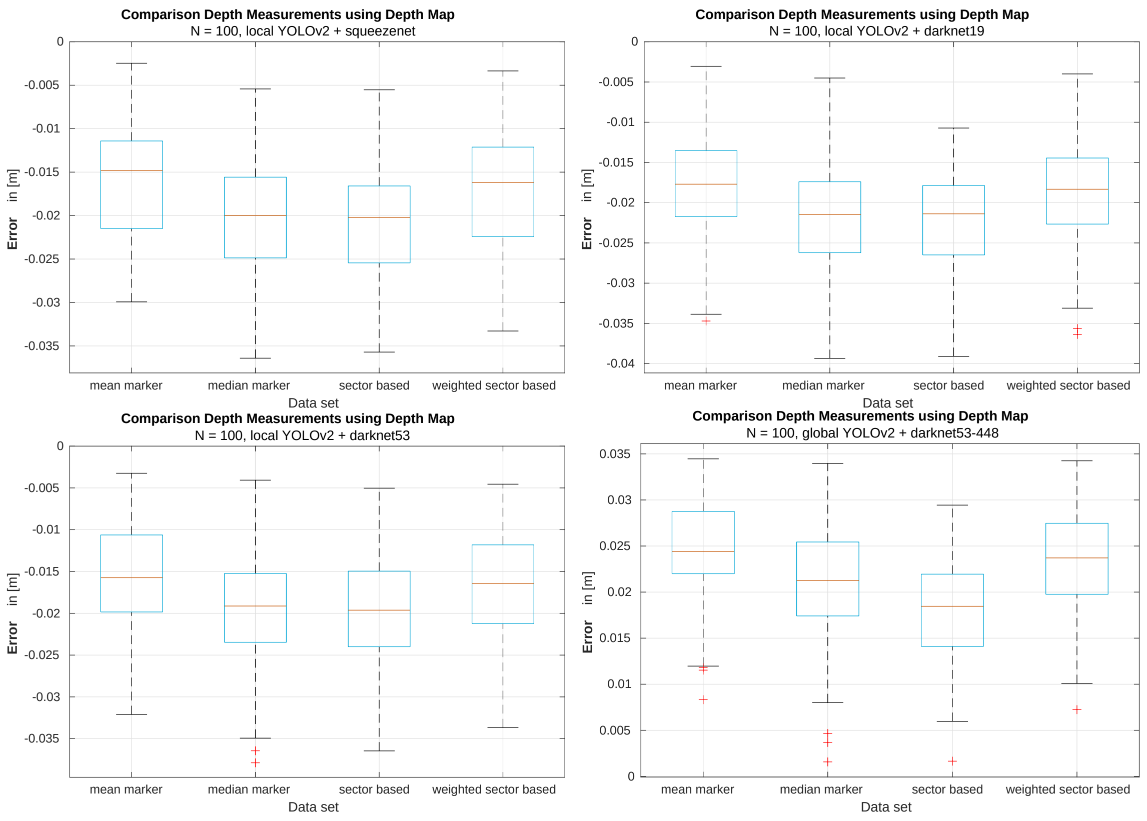


Figure 7-8: Measurement data for all used approaches for all depth map based determinations.

7-3 Online Experiment: Analysis of Observations

7-3-1 Detection Robustness Local versus Global

During the experiment the observation was made that the global detectors perform worse compared to the local detector in cases where the scene changes; for the change shown in fig. 7-9 the angle for the measurement was changed. The local detectors, of which only the the SqueezeNet based one is shown for conciseness, have a comparable detection performance compared to the case of normal experimental conditions. The global detector in contrast has a similar detection score to the experimental setup, but the overlap of bounding boxes with the actual LED matrix is less then using the local detectors, of which an example is shown in fig. 7-9.

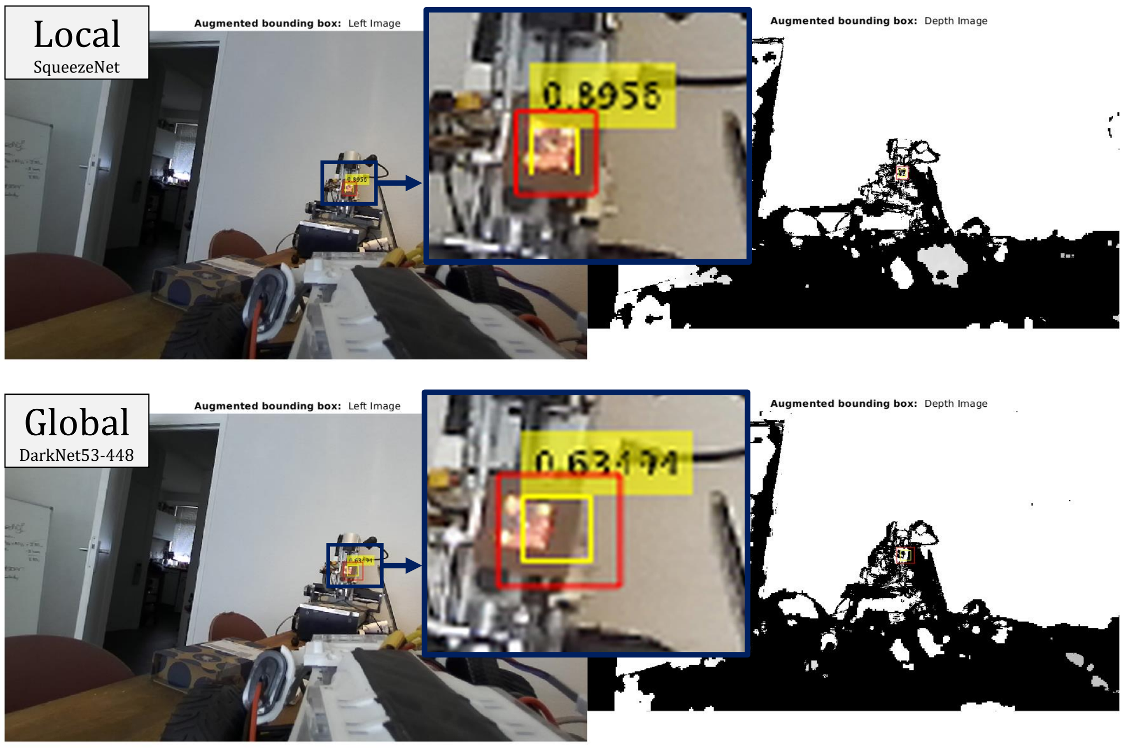

Local
SqueezeNet
Augmented bounding box: Left Image
0.8956
Augmented bounding box: Depth Image
Global
DarkNet53-448
Augmented bounding box: Left Image
0.63494
Augmented bounding box: Depth Image

Figure 7-9: Detection under slight angle using a local squeezenet and a global DarkNet53-448 based YOLOv2 detector.

This is in line with the expectations, as the datasets for the global were significantly less diverse concerning rotations, as described in section 6-4. Of course in applications this would result in problems and therefore in further research the dataset would need to be extended with images containing the LED matrix in different orientations and environments. This will be discussed later, together with different recommendations for further research, in section 8-2-2.

7-3-2 Depth Map Estimation Error: Periodic Behaviour

There seems to be a periodic behaviour in the error for the depth map based approaches. One hypothesis is due to the need for subpixel accuracy in the disparity; an interpolation is done between pixels, which could lead to larger errors. In order to investigate whether the periodic nature of the measurement errors could be caused by, or at least be contributed by, these interpolations, the errors are plotted versus the disparity[6] as shown in fig. 7-10.

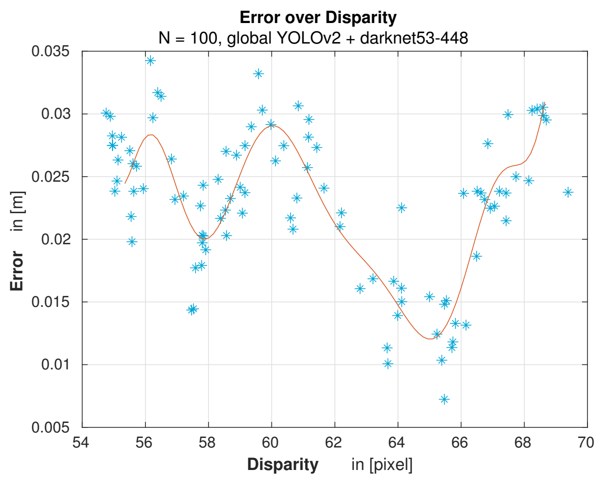


Figure 7-10: The disparity versus the error for the global DarkNet53-448 based YOLOv2 detector.

In fig. 7-10 disparity versus the error is shown, however a periodic influence is visible, this seems to have no correlation with the interpolation between integer values. Since the used toolbox is proprietary and closed source further investigation might prove difficult.

7-3-3 Depth Estimation Bias

In the depth map based approaches there seems to be a bias in all measurements. During experiments this bias seems to change depending on ambient light conditions, for example the time of the day, curtains open or closed, and the ceiling lamps on or off. In order to keep a constant measurement the experiment was done in the evening were only artificial light was added. The possible bias introduced by the set-up method ($\sim 3-4\,\text{mm}$) could contribute to the bias, but can not be the sole cause, as it is a order of magnitude lower. In preliminary test the biases were in the range of $[-5,5]$ mm even in extreme light conditions. Finding the exact source of the bias might prove difficult due to the proprietary nature of the toolbox.

Biases in this range should probably cause little harm in real life applications wherein a vehicle is tracked.

[6]Since the toolbox has no readily available or properly documented disparity map, this disparity is back-calculated from the distance.

7-4 Offline Experiment: Results

The LED matrix detection validation results of the offline experiment, as introduced in section 6-3, on one dataset of $N = 400$ datapoints[7]. The missed and false areas are shown normalized to the size of the groundtruth bounding box. For all sets there is stepwise behavior in the results, due to the type of error; as a bounding box is right or wrong per column or row of pixels, this error occurs mostly stepwise, as shown in fig. 7-11. Because of the normalization to the groundtruth box these errors are not complete steps.

Local SqueezeNet + YOLOv2
For the results of the local SqueezeNet YOLOv2 detector, the missed and falsely detected areas show similar trends generally. Closer by, in the $1.20 - 1.25\,\mathrm{m}$ range the missed area is larger than the falsely detected area, but in the rest of the distance range these are similar. Outside of this area the normalized missed and false areas are generally in the 0 to 10% range, and the Intersection over Union (IoU) is generally in the 85 to 100% range.

LocaL DarkNet19 + YOLOv2
The validation of the LED matrix detection shows a decrease of performance compared to the SqueezeNet based YOLOv2 detector over most of the range. This is in contrast to the results in the online experiment, wherein the local DarkNet19 YOLOv2 detector had slightly better results. This discrepancy could have come from different lighting conditions, due to being captured at a different time of the day, different ambient lighting, weather or similar causes.

Local DarkNet53 + YOLOv2
The local DarkNet53 YOLOv2 detector has a better performance compared to the DarkNet19 one, with an IoU generally in the 75 to 100% range. The normalized missed area is generally slighty larger compared to the falsely detected area.

Global DarkNet53-448 + YOLOv2
The global DarkNet53-448 YOLOv2 detector peforms the worst from the tested detectors IoU-wise. The detector has the tendency to create too large bounding boxes, which reflects back in the results shown in fig. 7-11. Especially at larger distances the missed area goes to zero, while the falsely detected area becomes larger, behavior visible if a too large estimated bounding box is detected on the LED matrix and it groundtruth label. Furthermore the IoU, which is linked to the confidence scores, is similarly lower compared to the results of the online experiment.

In all detectors the results were better at further distances. During early experiments there were issues with detecting consistently at the end of the range. To counteract this a larger part of annotated images in this range was added to the local and global training datasets. It is possible that due to the larger percentage of far away images, the detector is better trained at those distances.

[7]Which includes 800 groundtruth images, as both the left and right images are annotated and included.

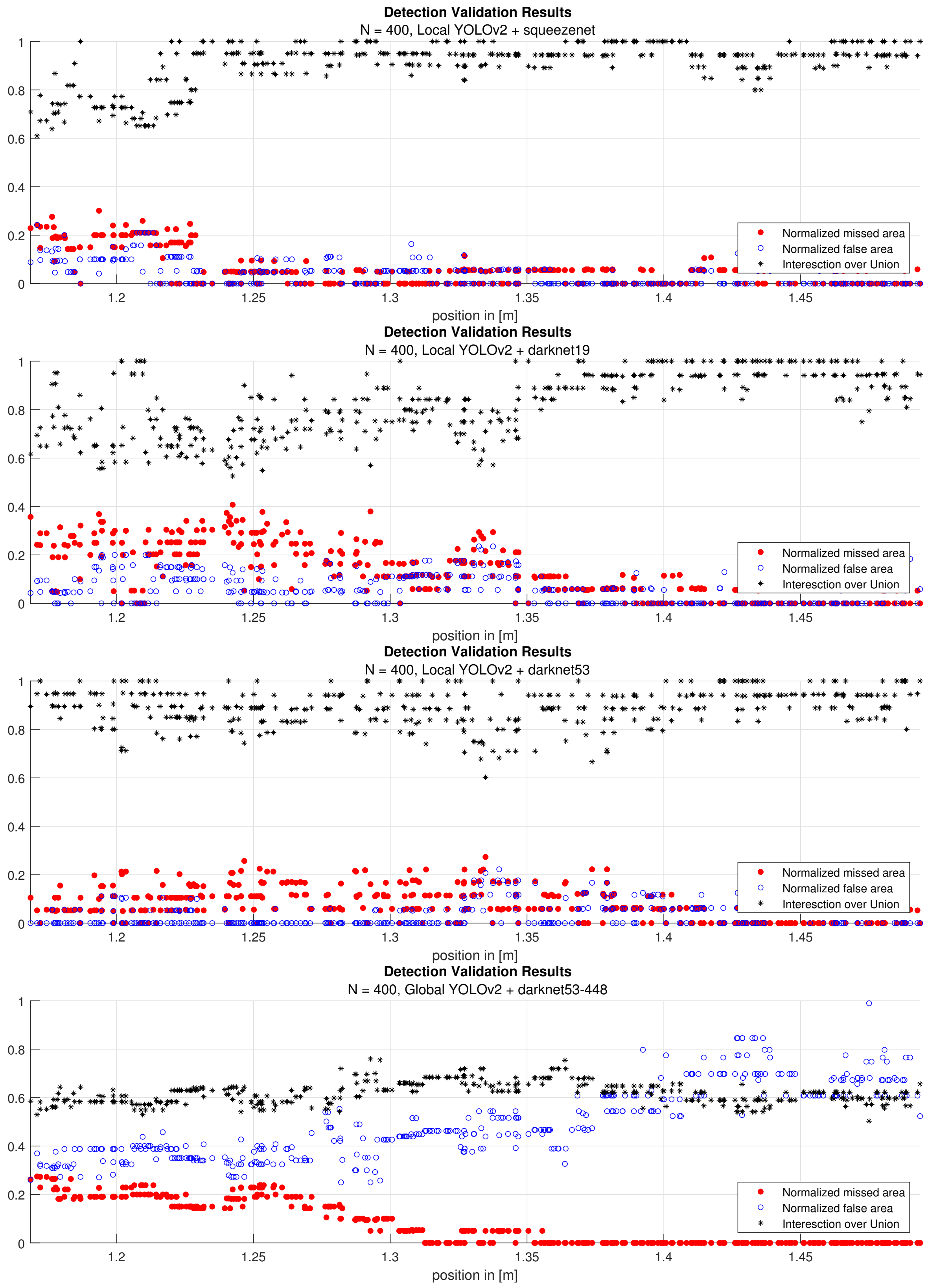


Figure 7-11: Detection results of offline experiment using multiple detectors and approaches, as labeled in the image tile, N denotes the amount of measurements.

Chapter 8

Conclusions and Discussion

This chapter draws conclusions based on the results in the previous chapter and discusses improvements on the the experimental setup or research apporach for further research.

8-1 Conclusions

8-1-1 Conclusions about Platform Development

This subsection has as aim to answer research question about the platform development, 𝔸, which are stated below.

𝔸. What is a suited design of the vehicle for platooning purposes while satisfying the design criteria of being easy to reproduce, having a balanced weight and flexible camera placement, while minimizing cost?

A driving platform was developed with stereo vision and processing, visual messaging capabilities, which can be localized by a motion capture system using visual markers. In order to enable these capabilities the following parts where implemented:

- **StereoLabs ZED Camera**
  The ZED stereo camera used for robot vision and depth estimation.
- **NVIDIA Jetson TX2 and ConnectTech Orbitty**
  The NVIDIA Jetson TX2 is a high performance system on module used for camera processing. The ConnectTech Orbitty carrier board allows interfacing of the NVIDIA Jetson TX2 with the other vehicle parts.
- **Erle Robotics Erle Rover**
  The Erle Robotics Erle Rover is vehicle base used for robotics research, and is equipped with the ErleBrain 3. The ErleBrain 3 is a Raspberry Pi based computer which controls

the vehicles motion. Due to the bankruptcy of Erle Robotics a replacement for these parts is required, the LRP S10 twister 2 2WD (which is base from Erle Rover), and a Pixhawk PX4 might prove a suitable alternative.

- **Batteries**
  The system uses two batteries. One NiMH battery mainly for the Erle Rover, including the ErleBrain, one 3S LiPo battery mainly for the NVIDIA Jetson and camera.
- **LED Matrices**
  Four LED matrices are mounted on the front, left, right and backside of the vehicle in order to convey visual messages. These LED matrices each have a dedicated driver and are controlled by an Arduino Nano.
- **MoCap Markers**
  In order for the motion camera system to estimate the location of the vehicle, this vehicle needs to be equipped with visual markers. These markers need to have a unique configuration per used vehicle in order to differentiate between these.
- **Custom Power Electronic Management Board**
  The system uses a custom power electronic management board, which distributed power and cuts power from the battery when the voltage gets to low. Furthermore it allows the system to be run on one battery instead of two, should the need arise.

These parts are mounted on a frame which is mounted on the Erle Rover body. This frame is mainly constructed from acrylic and 3D printed parts. These parts should be easy reproducible for a low cost, allowing to for little added costs on top of the previously mentioned parts. The design of this frame places the camera at the back of the platform, allowing sharp vision just in front of the car. In order to compensate for this weight added to the back, the batteries are places in the front, yielding a center of mass in the middle of the robot.

8-1-2 Conclusions about Visual Communication

This subsection has as aim to answer research question about the visual communication, $\mathbb{B}$, which is stated below. In order to answer this question, first a few conclusions will be drawn from the previous chapters, before applying these to the research question.

$\mathbb{B}$. *Using current available vision toolboxes, is it viable to consistently localize a LED matrix in a real time scenario?*

$\mathbb{C}$. *Having found a LED matrix, is it viable to consistently estimate the depth of this LED matrix from the camera in a real time scenario?*

Two experiments were conducted, in both causes the vehicle was placed in front of a linear stage (modified LPKF ProtoMat 91 PCB mill), on which a LED matrix was mounted. This LED matrix was moved in one direction in a distance range of 1.2 to 1.5 m, with no rotation and a plain background.

1. **Online Experiment:** Distance Estimation

   An online experiment wherein each iteration the linear stage moves and the platform detects LED matrices and estimates distances. These distances are compared with the known position of the linear stage to determine the estimation errors.

2. **Offline Experiment:** Detection Validation

   An offline experiment wherein the detection algorithms are run on a validation dataset of annotated images, captured over the entire distance range. These detections are compared with the hand annotated groundtruth in order to determine the main metric, the Intersection over Union (IoU).

From the results, for both the online and offline experiments, shown in the previous chapter the following conclusions are be drawn:

- **Depth map based approaches outperform bounding box based approaches in the tested distance range.**

  In the tested distance range, between 1.2 m and 1.5 m, the depth map based approach, which uses the camera toolbox method based on covariances, outperforms the bounding box based approach. The error using the depth map is on average a factor five lower compared to the bounding based method, with lower variances and less outliers.

- **Local detectors have significantly higher confidence scores, but little added accuracy in experimental setup.**

  The confidence scores of the LED matrix detection for the global approach are considerably lower compared to the local approach based scores. According to the measurement data, the depth accuracy based on the depth map based approach is comparable between local and global approaches in the experimental setup. It was observed that in measurements outside the experimental setup the bounding box estimation of the global approach is less reliable, compared to the local detectors. One possible solution could be to create training data of more situations, leading to a more diverse dataset.

- **Local detectors are too slow for real time platooning, global DarkNet53 based detector is possibly viable, using the development hardware, using the current approach.**

  The local detectors take between 5.75 s and 12.5 s per iteration, depending on base net, to detect a LED matrix. This is in real time applications an unviable amount of time. The global DarkNet53-448 based YOLOv2 detector takes a more suited 0.266 s, which, after optimization, could be a viable computation time. The depth estimation time is generally negligible in time consumption compared to the detection time, taking generally 0.8 to 2.5 ms.
  It should be noted that this might not reflect well to real life applications, since the unoptimized Matlab code was run on a HP 'Zbook' laptop.

- **Unweighted sector based depth estimation has the highest performance for depth map based approaches.**

  The sector based approach for depth estimation using the depth map had the lowest error and therefore the highest performance for the tested approaches.

- **LED matrices can be consistently found, however quality of matches vary.**

  In the offline experiment, wherein on a validation dataset, with hand annotated LED matrices, the detection algorithms are applied, all LED matrices were localized for all detectors. The quality of these detections, mainly quantified using the IoU, varies per detector and over distance. The best results in terms of IoU were achieved for the local SqueezeNet YOLOv2 detector, with IoUs generally between the 85 and 100%. The worst results were achieved by the global DarkNet53-448 YOLOv2 detector, which had IoUs in the 50 to 75%. These low IoU were caused by too large estimated bounding boxes, which missed no groundtruth area, but falsely labeled neighborhood pixels around the LED matrix as well.

The next step is to apply these conclusion to the research question $\mathbb{B}$. It is shown in the previous conclusion that it is indeed possible to consistently detect and localize LED matrices, and it is indeed possible to localize LED matrices in a real time scenario. Combining the two properties however might prove troublesome using the current research, as the global detector which is fast enough for real world scenarios, have trouble with situations which are not exactly the experimental set-up. The local detectors do not share this limitation, but are too slow for real world scenarios. When expanding the training image database this problem with the global detector might be solved in further research. For the global detectors a solution might lie in vastly expanding the training database, for local detectors parallel computing and only rescanning part of an image.

8-2 Discussion and Future Research

8-2-1 Hardware

This subjection discusses limitations of the current hardware of the vehicle and proposes possible solutions.

- **Erle Robotics dependency: Robot base.**
  For the scalability there is a dependency on the Erle Robotics base platform. During the research period, after the design was finished, Erle Robotics went bankrupt. It should probably be possible to find a other similar robot base. In fact Erle Robotics used a prebuilt vehicle from LRP (Lautenbach Racing Products GmbH), a vehcile very similar to the LRP S10 twister 2 2WD, which still sells the vehicle. Since RC cars use a largely standardised control of the steering and forward motion motors, changing to a different brand may prove trivial. Therefore the bankruptcy or Erle robotics should not prove a unsolvable problem.

- **Erle Robotics dependency: ErleBrain 3.**
  Continuing on the last item, the ErleBrain 3 embedded computer is a part less trivially replaced. Other RC autopilots are available, such as the PixHawk. Connecting such a system to the NVIDIA Jetson and a RC vehicle would solve this problem.

- **Implementing motion capture camera system.**
  On the platform the motion capture markers were mounted, but due practical limitations these were not implemented and tested yet. In applications where a reference to the vehicles own locations is required, this might be a useful tool.

- **Automatic pitch correction of the camera.**
  In the current state the camera is in a fixed, but adjustable, position. When at some point in time the vehicle will be used on non-flat surfaces, this might be problematic. Having an a system that can change the pitch angle while driving might be beneficial.

8-2-2 Object Recognition

This subjection discusses potential improvements of LED matrix detection.

- **Switch to more recent YOLO implementation.**

  The decision for YOLOv2 and not the more recent version was based among other reasons on the maturity of the support of YOLOv2 in the Matlab environment. If one switched to a other environment, the newer versions of YOLO, such as YOLOv3 could yield great improvements. These imporvements include for example that the YOLOv3 handles scaling much better compared to YOLOv2, which is very useful in this application.

- **Generate more training data automatically.**

  Since the local detectors became adept at estimating correct bounding boxes, all be it slowly, these could be used to automatically expand the training data base. It would be advisable to check all generated images for false positives or negatives, wrongfully estimated borders and other causes for bad data. Implementing such methods might allow for faster deployment or higher performance of global detectors without having to label vast amounts of images.

- **Determine LED matrix message.**

  The readout of the actual LED matrix message was outside the scope of the current research, but yields a possibility for follow up research. A basic communication protocol was developed, but a more sophisticated approach should be required in further research. This could either developing a more elaborate communication protocol or switching to a LED matrix type with a higher resolution, allowing for standard QR code protocols.

- **Rescan only Part of Image.**

  Using the local detectors, using a sliding window over the entire image is computationally demanding. Only applying the detector on a few image frames, where the LED matrix is expected based on previous frames, might prove an approach to cut computation time to just a fraction of the current computation times.

8-2-3 Distance Estimation

- **Investigate estimate depth based on focusing on LED.**
  There is recent research in high speed high accuracy depth estimation on a light source by adjusting the focus[33], [34]. In the context of this research this may be a more suited solution for depth estimation compared to the currently proposed one.

- **Investigate the source of biases in depth estimation.**
  Since in all tested situations biases occurred, research into the source of these may prove valuable. However, due to the closed source nature of the toolbox, this may prove difficult.

References

[1] M. Green and J. Senders, “Human error in road accidents,” *Visual Expert*, 2004.

[2] A. Al Alam, A. Gattami, and K. H. Johansson, “An experimental study on the fuel reduction potential of heavy duty vehicle platooning,” in *2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2010, pp. 306–311.

[3] R. Szeliski *et al.*, “Image alignment and stitching: A tutorial,” *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2007.

[4] G. Xu and Z. Zhang, *Epipolar geometry in stereo, motion and object recognition: a unified approach*. Springer Science & Business Media, 2013, vol. 6.

[5] R. Szeliski, *Computer vision: algorithms and applications*, ser. Texts in computer science. 2011, ISBN: 978-1-84882-934-3.

[6] A. Geiger, M. Roser, and R. Urtasun, “Efficient large-scale stereo matching,” in *Asian Conference on Computer Vision (ACCV)*, 2010.

[7] A. Geiger, J. Ziegler, and C. Stiller, “Stereoscan: Dense 3d reconstruction in real-time,” in *Intelligent Vehicles Symposium (IV)*, 2011.

[8] A. Vedaldi and A. Zisserman, “Recognition of object instances practical,” Accessed 16-februari-2019, 2018. [Online]. Available: `http://www.robots.ox.ac.uk/~vgg/practicals/instance-recognition/index.html`.

[9] J. Gonzales, F. Zhang, K. Li, and F. Borrelli, “Autonomous drifting with onboard sensors,” in *Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC16)*, 2016, p. 133.

[10] *Rapid autonomous complex-environment competing ackermann-steering robot (race-car)*, accessed 30-april-2021. [Online]. Available: `https://mit-racecar.github.io/`.

[11] M. O’Kelly, V. Sukhil, H. Abbas, *et al.*, “F1/10: An open-source autonomous cyber-physical platform,” *arXiv preprint arXiv:1901.08567*, 2019.

[12] B. Goldfain, P. Drews, C. You, *et al.*, *Autorally an open platform for aggressive autonomous driving*, 2018. arXiv: `1806.00678 [cs.RO]`.

[13] "Information technology automatic identification and data capture techniques qr code bar code symbology specification," International Organization for Standardization, Geneva, CH, Standard, Feb. 2015.

[14] S. Tiwari, "An introduction to qr code technology," in *2016 International Conference on Information Technology (ICIT)*, IEEE, 2016, pp. 39–44.

[15] A. R. Robertson, "The cie 1976 color-difference formulae," *Color Research & Application*, vol. 2, no. 1, pp. 7–11, 1977. [Online]. Available: `https://doi.org/10.1002/j.1520-6378.1977.tb00104.x`.

[16] F. Schönsberg, Y. Roudi, and A. Treves, "Efficiency of local learning rules in threshold-linear associative networks," *Physical Review Letters*, vol. 126, no. 1, Jan. 2021, ISSN: 1079-7114. DOI: `10.1103/physrevlett.126.018301`.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. DOI: `10.1109/CVPR.2016.91`.

[18] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. DOI: `10.1109/CVPR.2017.690`.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[20] MathWorks, "Pretrained deep neural networks," [Online; accessed 02-march-2021]. [Online]. Available: `https://nl.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html`.

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size*, 2016. arXiv: `1602.07360 [cs.CV]`.

[22] J. Redmon, *Darknet: Open source neural networks in c*, `http://pjreddie.com/darknet/`, 2013.

[23] J. Redmon and A. Farhadi, "Yolo9000 better, faster, stronger," 2017, pp. 7263–7271.

[24] A. Farhadi and J. Redmon, "Yolov3 an incremental improvement," *Computer Vision and Pattern Recognition, cite as*, 2018.

[25] A. Defazio, "Optimisation techniques i," [accessed 06-march-2021], Feb. 2020. [Online]. Available: `https://atcold.github.io/pytorch-Deep-Learning/en/week05/05-1/`.

[26] R. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, 1986, pp. 823–832.

[27] A. Cutkosky and H. Mehta, "Momentum improves normalized SGD," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, Jul. 2020, pp. 2260–2268. [Online]. Available: `http://proceedings.mlr.press/v119/cutkosky20b.html`.

[28] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. arXiv: 1412.6980 [cs.LG].

[30] S. Ruder, *An overview of gradient descent optimization algorithms*, 2017. arXiv: 1609.04747 [cs.LG].

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[33] J. Ens and P. Lawrence, "An investigation of methods for determining depth from focus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 2, pp. 97–108, 1993. DOI: 10.1109/34.192482.

[34] W. Fuhl, T. Santini, and E. Kasneci, *Fast camera focus estimation for gaze-based focus control*, 2017. arXiv: 1711.03306 [cs.CV].

Glossary

List of Acronyms

A/D	Analog to Digital
ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
CCD	Charge-Coupled Device
CFA	Color Filter Array
CMOS	Complementary Metal-Oxide-Semiconductor
DNN	Deep Neural Network
ESC	Electronic Speed Controller
FoV	Field of View
IoU	Intersection over Union
PDB	Power Distribution Board
RMSprop	Root Mean Square Propagation
ROI	Region-of-Interest
SDK	Software Development Kit
SoM	System on Module