Master's thesis

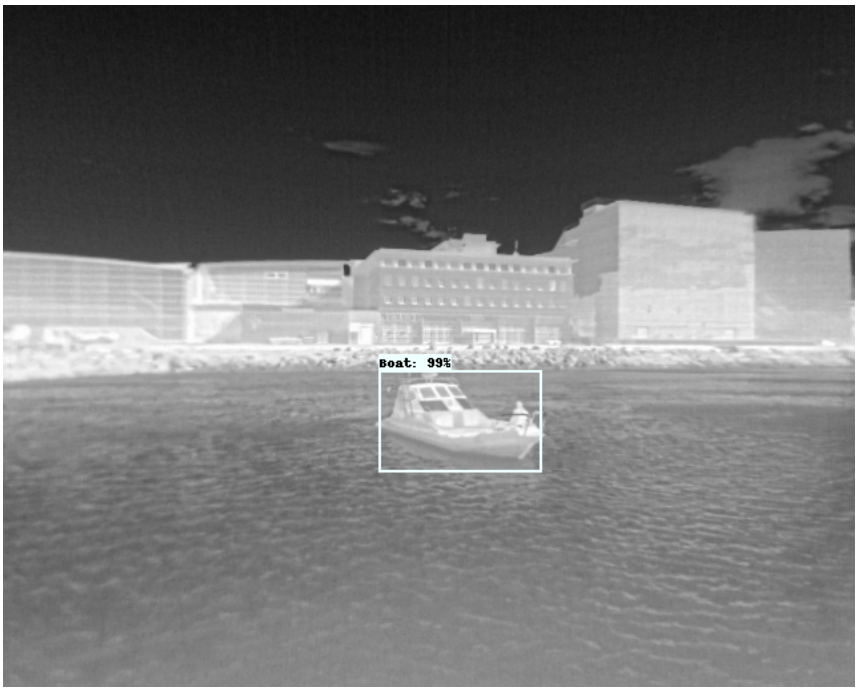
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Eskil Hatling Hølland

Maritime object detection using infrared cameras

A pipeline for object detection in 360 degree long wave infrared image data for use in navigation and collision avoidance

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
June 2019

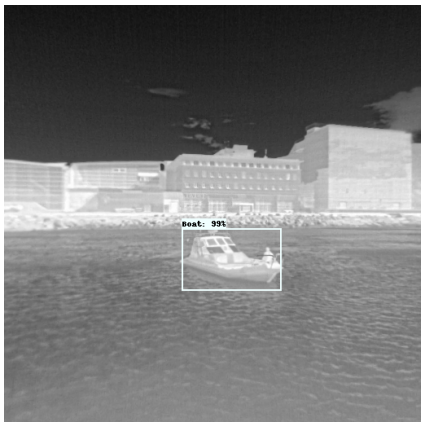


NTNU
Norwegian University of Science and Technology

Eskil Hatling Hølland

Maritime object detection using infrared cameras

A pipeline for object detection in 360 degree long wave infrared image data for use in navigation and collision avoidance



Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
June 2019

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

NTNU
Norwegian University of
Science and Technology

Problem description

Autonomous surface vehicles (ASVs) can use a variety of sensors for sense-and-avoid and navigation purposes. To add important redundancy, the ASV should be equipped with both active and passive sensors. Long wave infrared (LWIR) sensitive cameras are important components of the sensor suite for the autonomous ferry MilliAmpere, and a tailor-made 360 degree sensor rig including 5 LWIR cameras has therefore been developed by Maritime Robotics (MR). The rich information from LWIR sensitive cameras might add redundancy needed for safe autonomous operation.

The purpose of this project is to develop an object detection pipeline using the LWIR layer of the MR sensor rig for sense-and-avoid purposes onboard MilliAmpere. The project builds on the author's specialization project [1] written during the Autumn 2018, where data acquisition and methods for improving object detection performance in LWIR data were studied. This Masters thesis shall address the following tasks:

- Become familiar with the LWIR-layer of the MR sensor rig for MilliAmpere and complete necessary image reading software in collaboration with MR employees.
- Record LWIR data using the MR sensor rig at sea in collaboration with Autoferry PhD candidates.
- Calibrate the LWIR sensitive cameras of the MR sensor rig and based on this make a camera model that can translate pixels to rays of camera-frame coordinates and vice versa.
- Based on the results in the author's specialization project [1], implement a detector for the LWIR data.
- Evaluate the uncertainties of the detections in terms of detection probabilities.

Preface

With this thesis I have made my final work in my education leading to the degree Master of Science in Kybernetikk og Robotikk. The thesis is submitted to the Department of Engineering Cybernetics at the Norwegian University of Science and Technology.

This thesis is closely linked with the Autoferry project. Demonstrating a working object detection pipeline utilizing infrared cameras with 360° field of view as done in this thesis is a key stepping stone in the Autoferry project. Autoferry is a cross-disciplinary NTNU project which aims to develop groundbreaking new concepts and methods which will enable the development of autonomous passenger ferries for transport of people.

Working with this thesis has allowed me to combine knowledge and skills from several areas, ranging from electro-optics to deep learning. The work spans 3 technology categories; Geometric camera calibration of long wave infrared sensitive cameras, real-time object detection and perspective projection.

I would like to extend my gratitude to several people for making this thesis possible. First of all i would like to thank my supervisor Edmund Førland Brekke as well as my co-supervisors Kenan Trnka, Øystein Kaarstad Helgesen and Håkon Hagen Helgesen for providing invaluable guidance and discussion. I would also like to thank Kasper Trolle Borup for supplying thermal coating, Glenn Angell for help with 3D printing and laser cutting, the guys at Elektroverkstedet for letting me use their equipment as well as Tobias Torben and Runar André Olsen for piloting MilliAmpere and Telemetron respectively. Lastly, I wish to thank my parents and girlfriend for support during my masters thesis.

Eskil Hatling Hølland

Trondheim, June, 2019

Abstract

Autonomous transport is on the rise, both on land, in the air and at sea. Urban water channels are however underused in today's cities. The main ways of crossing channels are either by bridge or by a manned passenger ferry. Both methods require expensive investments, in short- as well as long-term. The world has yet to see a cost-effective, fully autonomous and environmental friendly passenger ferry.

For any autonomous system, situational awareness is paramount. With the rapid advancement in both camera technology, computing power and computer vision, new approaches are attainable. In contrast to ordinary electro-optical cameras, long wave infrared (LWIR) sensitive cameras are useful without illumination. While the resolution is more coarse than for electro-optical cameras, it can still be superior to the resolution of a radar and is less susceptible to weather detoriation than a lidar. The main weakness of LWIR cameras are background clutter and the fact that it is a passive sensor, making direct range measurements impossible with a mono setup. As a separate sensor layer, it might however add important redundancy to an autonomous system.

Using cameras as sensors, electro-optical as well as LWIR sensitive, requires accurate calibration. While several methods for geometric calibration of LWIR cameras have been proposed, a standardized method does not exist. Therefore, a new method for geometric calibration of cameras sensitive to LWIR radiation utilizing emissivity differences in materials is proposed in this thesis. The method is based on Leslie's cube [2] and mimics the standardized method for electro-optical cameras and can be used with Matlab's camera calibration application among others. The average error in pixel length using distortion coefficients and intrinsic parameters from this method were 1.91 px or 1 px horizontally and vertically. Based on this method, a camera model translating pixel coordinates to rays of camera-frame coordinates is implemented as part of a real time object detection pipeline. The pipeline uses data from a sensor rig with 5 wide angle LWIR cameras arranged in a configuration resulting in full 360° field of view.

The thesis demonstrates a functional Robot Operating System (ROS) network publishing normalized rays of camera-frame coordinates pointing to detected objects found in infrared data from the sensor rig. The network is tailored to MilliAmpere, the prototype autonomous passenger ferry developed as part of the Autoferry project.

Sammendrag

Autonom transport blir stadig mer tilgjengelig, både landbasert, luftbasert og til havs. Urbane vannveier er på den andre siden underutnyttet i dagens byer. Den vanlige måten å krysse en kanal er enten via bro eller en bemannet ferge, som er dyrt på både kort og på lang sikt. Verden har til overs å se en kost-effektiv, fullt autonom og miljøvennlig personferge.

For ethvert autonomt system er situasjonsforståelse kritisk. Nyvinninger innen kamerateknologi, beregningskraft og datasyn muliggjør nye framgangsmåter. I motsetning til vanlige elektrooptiske kameraer, er langbølge infrarød (LWIR) sensitive kameraer nyttige uten belysning. Selvom oppløsningen er lavere enn for elektro-optiske kameraer kan de fortsatt utklassere oppløsningen til en radar og være mindre utsatt for dårlig vær enn en lidar. Hovedsvakhetene til LWIR kameraer er bakgrunnsrot og faktum at det er passive sensorer, som gjør direkte avstandsmålinger umulig med et mono oppsett. Som et separat lag av sensorer vil det mulig kunne øke viktig redundans i autonome systemer.

Når kameraer, både elektro-optiske og LWIR-sensitive, skal brukes som sensorer er det viktig med nøyaktig kalibrering. Selvom flere metoder for geometrisk kalibrering av LWIR kameraer har blitt lagt fram finnes det ikke standariserte metoder. Derfor blir en ny metode for geometrisk kalibrering av LWIR sensitive kameraer som utnytter emissivitetsforskjeller i materialer lagt fram. Metoden er basert på Leslie's kube [2] og etterligner den standardiserte metoden for elektro-optiske kameraer og kan blant annet bli brukt med Matlab sin kamera-kalibrerings applikasjon. Gjennomsnittlig feil i piksellengde ved bruk av forvrengnings-koeffisienter og indre kameraparametre fra denne metoden var 1.91 px eller 1 px horisontalt og vertikalt. Basert på denne metoden har en kameramodell som oversetter pikselkoordinater til stråler i kamerakoordinater blitt implementert som en del av objekt-deteksjon samlebåndet. Samlebåndet bruker data fra en sensorrigg med 5 vidvinkel LWIR sensitive kameraer arrangert i en konfigurasjon som resulterer i 360° synsvinkel.

Avhandlingen demonstrerer et funksjonelt Robot Operating System (ROS) nettverk som publiserer stråler i kamerakoordinater som peker til detekterte objekter funnet i infrarød data fra sensorriggen. Nettverket er skreddersydd MilliAmpere, prototypen til en autonom personferge utviklet som en del av Autoferry prosjektet.

Table of Contents

Abbreviations

- IR - Infrared
- EO - Electro Optical
- ROS - Robot Operating System
- MR - Maritime Robotics
- SSH - Secure SHell
- GPS - Global Positioning System
- OBC - On Board Computer
- IMU - Inertial Measurement Unit
- CNN - Convolutional Neural Network
- RTK - Real-Time Kinematic
- VHF - Very High Frequency
- API - Application Programming Interface
- IDE - Integrated Development Environment
- IPC - InterProcess Communication
- GPU - Graphics Processing Unit
- mAP - mean Average Precision
- RGB - Red Green Blue
- FOV - Field Of View
- LiDAR - Light Detection And Ranging

Introduction

Motivation

Autonomous transport is on the rise, both on land, in the air and at sea. Urban water channels are however underused in today's cities. The main ways of crossing channels are by bridge or by manned passenger ferries. Both methods require expensive investments, in short- as well as long term. The world has yet to see a cost-effective, fully autonomous and environmental friendly passenger ferry.

For any autonomous system, situational awareness is paramount. Situational awareness can be broken down in three separate levels [3]; Perception of the elements in the environment, comprehension of the current situation and projection of the future situation. The complexity of urban environments means that the rich information from cameras will be important. With the rapid advancement in both camera technology, computing power and computer vision, new approaches for enhancing the first level of situational awareness, perception, are attainable.

LWIR sensitive cameras have a very promising potential in sense-and-avoid for autonomous surface vehicles and have proven to increase detection and tracking performance as an additional sensor layer [4]. From [1], LWIR sensitive cameras have shown promise in acting as sensors for use in object detection. In contrast to ordinary cameras they are useful during both day and night. While the resolution is more coarse than for ordinary cameras, it can still be much more accurate than the resolution of a radar. LWIR is generally less susceptible to weather deterioration than other sensors, e.g. LiDAR. The main weakness is that it is a passive sensor, and therefore cannot measure range directly. However, this may to some extent be remedied by sensor fusion, semantic information or georeferencing. Therefore, LWIR sensitive cameras are included as a separate passive exteroceptive sensor layer of MilliAmpere, the prototype autonomous passenger ferry developed by the Autoferry team.

The Autoferry project

Autoferry is a cross-disciplinary NTNU project which aims to develop groundbreaking new concepts and methods which will enable the development of autonomous passenger ferries for transport of people in urban water channels [5]. The concept of small, autonomous passenger ferries is a more flexible, cost-effective and environmental friendly alternative to bridges and manned passenger ferries. The ferries can either be summoned by the press of a button, or coordinated with public transport making urban commute more effective. The project involves 19 researchers from three faculties and all three NTNU campuses.

In Norway, there is a limited amount of cable ferries. The project aims to remove the cable and enable autonomous operation. The scalability of the technology also opens up possibilities for coastal transport. A large part of Norway's population lives along the coast. Reducing the cost of ferry services using autonomous ferries will enable revitalization and further development of the coastal areas. The development of autonomous ferries may additionally create markets not existing currently due to high crew costs.

The goal of the project is safe autonomous operation of passenger ferries alongside other vessels in confined and congested environments, e.g. water channels.

LWIR sensitive cameras are an important component of the sensor suite for MilliAmpere, and a tailor-made 360 degree sensor rig including LWIR sensitive cameras have therefore been developed by Maritime Robotics. When using cameras as sensors, accurate calibration is essential. As standardized methods for geometric calibration of LWIR sensitive cameras does not exist, two methods were developed and thoroughly tested during the work with the thesis. A ROS network tailored to the existing ROS network on MilliAmpere were developed and implemented. The network needs to be able to sample relevant data concerning camera-pose, detect and classify objects in LWIR data and calculate rays of camera-frame coordinates to the resulting detections. This pipeline gathers data from all five LWIR sensitive cameras and process it in real time to enable safe autonomous ferry operation. A picture of MilliAmpere as sensor platform for the sensor rig is included in Figure 1.

Figure 1: MilliAmpere with the MR sensor rig mounted.

Contributions

The main contributions of this thesis are listed below.

- From the author's specialization project [1], a dataset containing labeled IR data has been created as well as techniques for improving object detection performance of detectors utilizing IR data.
- All 5 LWIR sensitive cameras are geometrically calibrated, and a method for easy and accurate calibration of such cameras is proposed. A camera model translating detections to normalized rays of camera frame coordinates is formulated.
- Installation and integration of the IR layer of the sensor rig provided by MR into the existing system onboard MilliAmpere.
- A lightweight model has been trained and implemented with a detector enabling real time multiclass object detection using all 5 LWIR sensitive cameras.
- A series of experiments using a real target with ground truth world position have been executed using MilliAmpere as sensor platform. The resulting datasets contains all data published from the ROS network onboard as well as IR data from the sensor rig. The dataset has the potential to be used in future research projects both for Autoferry as well as other NTNU projects.

Thesis outline

The structure of this thesis is separated into parts as follows.

Part 1 presents a literature review of work related to the areas explored in this thesis. Relevant background theory concerning properties of infrared radiation and computer vision, including the pinhole camera model and image processing techniques, are presented.

Part 2 presents the sensors utilized, calibration procedures of LWIR sensitive cameras as well as the structure of the proposed object detection pipeline. The coordinate transformation to each camera given MilliAmpere pose are derived.

Part 3 presents the conducted experiments and the motivation behind them. The results are presented and discussed. The last chapter concludes this thesis and gives suggestions for areas which may benefit from further research given the results.

Part I

Related work and theory

Chapter 1

Literature review

1.1 Calibration of infrared cameras

There are two ways an infrared camera may need to be calibrated. This includes calibration of pixel intensity in correlation to temperature, i.e. thermal calibration, and geometric calibration.

- In [6] the authors proposed a method for calibration of IR-cameras utilizing a passive calibration field with 8x7 drilled holes. The plate was cooled to around 0°C to distinguish the holes from the plate in an infrared spectrum. Configuring the IR-camera to be sensitive in the temperature range 0°- 15°C yielded images resembling binary images. An OpenCV Calibration tool was used in order to calibrate the camera.
- In [7] the authors calibrated an IR-camera using a passive calibration field of aluminium with adhesive coded targets, including elevated targets for a better geometric configuration.
- In [8] the authors proposed a method for calibrating IR cameras using an active calibration field consisting of an 8x8 grid of lamps embedded in a wooden board. The calibration tool used was Photomodeler.
- In [9] the authors calibrated IR cameras using a very thin stainless steel plate with marks of known size and locations etched into it. The plate was bonded on to a glass support plate. Heating the glass from behind resulted in excellent targets in the IR spectre.
- In [10] the authors tested both an active calibration field with burning lamps and a passive calibration field made of aluminium which acts almost as a mirror in the IR spectre. Target points were made of self-adhesive foil which only emits radiation

relating to its own temperature.

1.2 Computer vision using infrared data

- In [4] the author demonstrated a working sensor fusion system for tracking and detection of other vessels at sea using radar, lidar, electro-optical and infrared cameras. It was found that adding an infrared camera always yielded positive results with respect to long range performance and tracking accuracy.
- In [11], the authors presented a survey on maritime object detection and tracking approaches. It was found that static background subtraction worked better with LWIR sensitive camera than for visible spectrum cameras.
- In [12], the authors proposed a thermal-infrared simultaneous localization and tracking (SLAM) system enhanced by sparse depth measurements from LiDAR. It was demonstrated that the system was robust under various lighting conditions as well as overcoming the scale problem of monocular cameras.

Chapter 2

Infrared Radiation

Infrared radiation have several properties different to visible radiation. Some of these properties will be presented in the following chapter. The information in this chapter is based on work from the author's specialization project [1].

2.1 Infrared and thermal radiation

IR radiation is EM radiation with wavelengths longer than visible red at 700 nm. There are several subdivisions of IR radiation, a commonly used scheme is presented in the table below [13].

Subdivision	Wavelength [μm]
NIR (Near Infrared)	0.7 - 1
SWIR (Short Wave Infrared)	1 - 3
MWIR (Mid Wave Infrared)	3 - 5
LWIR (Long Wave Infrared)	8 - 12

Most of the electromagnetic radiation with wavelength between 5 μm and 8 μm are attenuated by the atmosphere. Other definitions exists as well, such as LWIR and MWIR being referred to as TIR (Thermal InfraRed). TIR cameras are more sensitive to emitted radiation in everyday temperatures while NIR and SWIR cameras are more sensitive to reflected radiation. NIR and SWIR cameras behave in a similar way as visual cameras as they are more dependent on illumination.

When electromagnetic radiation interacts with matter, it can be absorbed, transmitted and reflected. The radiation balance law [14] states that

$$1 = \alpha + \rho + \tau \tag{2.1}$$

where α is the absorbed radiation, ρ is the reflected radiation and τ is the transmitted radiation and α, ρ, $\tau \in [0,1]$. In addition, thermal energy can be converted to EM-radiation, called thermal radiation. All bodies above absolute zero, 0 K, emit thermal radiation to different extent depending on material and its temperature. Emissivity (ϵ) is the ratio of actual emittance of an object to a black body, a hypothetical body with $\alpha = 1$ used for approximation, at equal temperature. Kirchoffs law of thermal radiation states that $\alpha = \epsilon$, which in turn means that $\epsilon = 1$ for a black body. Because emissivity is material dependent, it is an important property when measuring temperatures via thermal radiation. This means that an object composed of materials with different emissivity may read different temperature based on a thermal image even if the temperature is uniform. This is exemplified by Leslie's cube [2]. This was originally an experiment conducted by John Leslie where a cube of metal with a cavity was filled with hot water. The sides of the cube was coated with different materials with different emissivities. Even though all sides of the cube had approximately uniform temperature, the thermal detector registered different amounts of infrared radiation for each coating. Measured apparent temperature will also vary with respect to distance due to scattering of radiation in the atmosphere.

2.2 Thermal imaging

Thermal images are displays of measured emitted, reflected and transmitted thermal radiation within an area. The thermal radiation is often represented with pixel intensity. It is common to map the gray scale intensity to a color map to better visualize details [13]. Exact measurements of temperature can be challenging due to the amount of thermal radiation emitted depends on the objects emissivity as discussed in section 2.1. In addition, thermal radiation from the surroundings can be reflected on the surface of the object. Therefore, it is also important to know the reflectivity ρ of the object to be measured. If measuring over longer distances through the atmosphere a noticeable amount of radiation will be transmitted, absorbed and emitted by the atmosphere itself. This should be taken into consideration for precise measurements.

2.3 Advantages and limitations of thermal imaging

Thermal cameras may be favourable over visual cameras in conditions where there is a temperature differential connected to the object in question, e.g. humans, fire or animals. Thermal cameras are especially advantageous in dark environments or difficult weather conditions like snow or fog. This is again due to the sensitivity of emitted radiation as opposed to visual cameras which fully relies on reflected EM radiation and are therefore reliant on external illumination. The exception is objects hot enough to emit EM radiation in the visible spectrum.

Thermal cameras are generally expensive and have relatively low resolution compared to visual cameras. This means that details quickly vanish with increasing distance. Uncooled thermal cameras can be sensitive to movement in the sense that they may capture blurry

images if the camera is not steadily mounted or the object being imaged is in rapid movement [15]. This is due to the electrical signal in the pixel of a microbolometer detector decays exponentially, which may lead to a moving object being mapped to multiple pixels.

2.4 Infrared camera

Materials may have different properties in the infrared spectrum compared to the visual spectrum. As an example, while glass is transparent in the visual spectrum, it is opaque in the infrared spectrum. This is why infrared cameras have lenses made of germanium, which is transparent in the infrared spectrum and opaque and reflective in the visible spectrum, rather than glass. This can be seen in Figure 2.1

(a) LWIR spectrum **(b)** Visible spectrum

Figure 2.1: Difference in transparency of glass in the IR and visible spectrum

Thermal cameras is either cooled or uncooled. High-end cooled cameras can deliver hundreds of HD frames per second with a temperature sensitivity of 20 mK. Pixels are typically described by 16 bits to allow a large dynamic range. Uncooled cameras usually have microbolometer detectors and operate in LWIR spectrum. While they produce more noisy images at a lower framerate, they are smaller, silent and less expensive.

Chapter 3

Computer vision

Computer vision is a field of computer science that enables computers to see, identify and process visual data. The field has seen a rapid advancement and continues to do so after the emergence of deep learning and artificial neural networks. The information in sections 3.5-3.9 is based on work from the author's specialization project [1].

3.1 Pinhole camera model

A pinhole camera can be designed by placing a light-proof barrier with an aperture between the 3D object and a photographic film or sensor. Due to the barrier, only a few rays of light passes through from each point from the 3D object. Therefore, a one-to-one mapping between points on the 3D object and the sensor may be established. The result is that the sensor gets exposed by an image of the 3D object by means of this mapping. This model is known as the pinhole camera model [16]. An illustration may be seen in figure 3.1 where the image plane is located at the image sensor. The image captured on this sensor is equal to the inverted virtual image, located between the 3D object and the pinhole. The virtual image is always located at the same distance to the pinhole as the image sensor is to the pinhole, also known as the focal length.

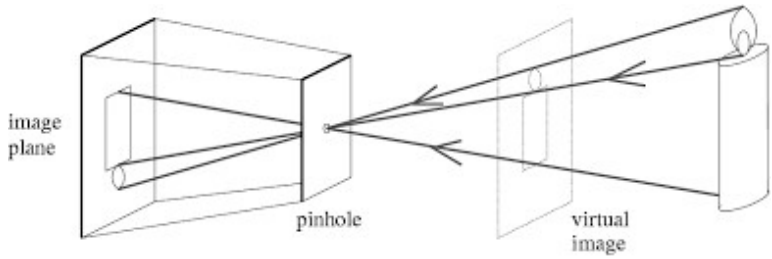


Figure 3.1: Visualization of the pinhole camera model. Originally published in [17]

3.2 Geometric camera parameters

World and camera coordinate systems are related by a set of physical parameters. These parameters can be divided into subsets of intrinsic and extrinsic parameters. The intrinsic parameters relates the camera's coordinate system to the idealized coordinate system while the extrinsic parameters relates the cameras coordinate system to a fixed world coordinate system and specify its position and orientation in space.

3.2.1 Intrinsic parameters

A pinhole camera can be associated with two different image planes; The first is the normalized plane located at a unit distance from the pinhole. This plane gets its own coordinate system with origin located at the point $\hat{C}$ where the optical axis pierces it as shown in Figure 3.2. [18]

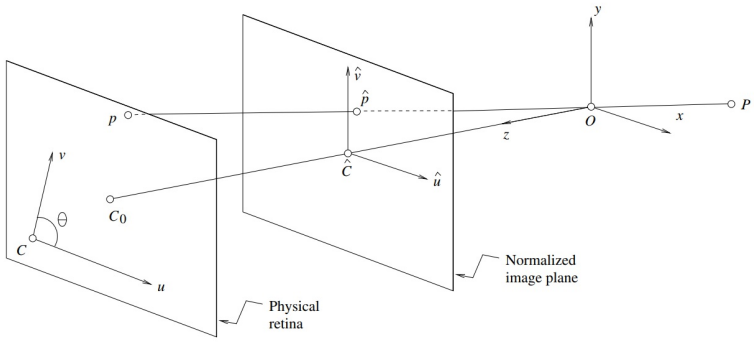


Figure 3.2: Physical and normalized image coordinate systems. Originally published in [18].

In Figure 3.2, $\hat{C}$ is the origin of the normalized image plane, C_0 is the origin of the physical image plane and O is the origin of the camera 3D frame.

The perspective projection equation can be written as in equation (3.1) in the normalized coordinate system.

$$\begin{aligned} \tilde{u}_i &= \frac{x_i}{z_i} \\ \tilde{v}_i &= \frac{y_i}{z_i} \end{aligned} \iff \hat{\boldsymbol{p}} = \frac{1}{z} \begin{pmatrix} \boldsymbol{I} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \boldsymbol{P} \\ 1 \end{pmatrix} \tag{3.1}$$

The physical retina of the camera is usually different in that it is located at a distance $f \neq 1$ together with the image coordinates u and v of the image are generally expressed in pixel units. Because pixels are usually rectangles instead of squares, the camera has two additional scale parameters k and l as shown in equations (3.2)-(3.3).

$$\tilde{u}_i = kf\frac{x_i}{z_i} \tag{3.2}$$

$$\tilde{v}_i = lf\frac{y_i}{z_i} \tag{3.3}$$

The parameters k, l and f are not independent and can be replaced by magnifications $\alpha = kf$ and $\beta = lf$ expressed in pixel units. The origin of the camera coordinate system is at a corner C of the retina, usually upper or lower left corner, and not at its center. The center of the CCD matrix usually does not coincide with the principal point C_0, thus (3.2)-(3.3) is now replaced by (3.4)-(3.5)

$$\tilde{u}_i = \alpha\frac{x_i}{z_i} + u_0 \tag{3.4}$$

$$\tilde{v}_i = \beta\frac{y_i}{z_i} + v_0 \tag{3.5}$$

where u_0 and v_0 defines the position of C_0 in the retinal coordinate system in pixel units. The camera coordinate system may also be skewed, implying the angle θ between the two image axis is not equal to 90°. Thus, (3.4)-(3.5) is replaced by (3.6)-(3.7)

$$\tilde{u}_i = \alpha\frac{x_i}{z_i} - \alpha\cot\theta\frac{y_i}{z_i} + u_0 \tag{3.6}$$

$$\tilde{v} = \frac{\beta}{\sin\theta}\frac{y_i}{z_i} + v_0 \tag{3.7}$$

Combining (3.1) and (3.6-3.7) results in

$$\boldsymbol{p} = \kappa\hat{\boldsymbol{p}}, \qquad \text{where} \qquad \boldsymbol{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \qquad \text{and} \qquad \kappa \overset{\text{def}}{=} \begin{pmatrix} \alpha & -\alpha\cot\theta & u_0 \\ 0 & \frac{\beta}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

giving

$$\boldsymbol{p} = \frac{1}{z} M\boldsymbol{P}, \qquad \text{where} \qquad M \overset{\text{def}}{=} \begin{pmatrix} \kappa & \mathbf{0} \end{pmatrix} \tag{3.8}$$

where $\boldsymbol{P}$ denotes the homogenous coordinate vector of P in the camera coordinate system. Homogenous coordinates allows representing the perspective projection mapping by the $3x4$ matrix M. κ is the *calibration matrix* of the camera and the parameters α, β, θ, u_0 and v_0 are the *intrinsic parameters* of the camera. These are the parameters which are identified when a camera is geometrically calibrated.

3.2.2 Extrinsic parameters

Consider the case where the camera frame C is distinct from the world frame W. With

$${}^{C}P = \begin{pmatrix} {}^{C}_{W}R & {}^{C}O_{W} \end{pmatrix} \begin{pmatrix} {}^{W}P \\ 1 \end{pmatrix} \tag{3.9}$$

and substituting (3.8) yields

$$\boldsymbol{p} = \frac{1}{z} M\boldsymbol{P}, \qquad \text{where} \qquad M = K \begin{pmatrix} R & t \end{pmatrix} \tag{3.10}$$

where $R = {}^{C}_{W}R$ is a rotation matrix, $\boldsymbol{t} = {}^{C}O_{W}$ is a translation vector and $\boldsymbol{P}$ denotes the vector of homogeneous coordinates of P in the frame W. M determines the position of the cameras optical center in the world frame W where R and $\boldsymbol{t}$ defines the six *extrinsic parameters* and respective degrees of freedom. The general perspective projection from (3.10) is often written $z\boldsymbol{p} = M\boldsymbol{P}$. Using homogeneous coordinates, it can be written as $\boldsymbol{p} = M\boldsymbol{P}$ when exploiting the fact that they are only defined up to scale. The actual image coordinates of image point p is defined as $\frac{u}{v}$ and $\frac{v}{w}$ if $\boldsymbol{p} = \begin{pmatrix} u, & v, & w, \end{pmatrix}^T$ in which M is also is defined up to scale. M can be rewritten explicitly as a function of the cameras intrinsic and extrinsic parameters as seen in equation (3.11)

$$M = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \frac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix} \tag{3.11}$$

where $\boldsymbol{r}_1^T$, $\boldsymbol{r}_2^T$ and $\boldsymbol{r}_3^T$ denote the rows of the matrix R and t_x, t_y and t_z are the coordinates of the vector $\boldsymbol{t}$ in the frame attached to the camera. While the intrinsic parameters are typically found through a calibration routine, the extrinsic parameters depends on the location and pose of the camera frame relative to the world frame.

3.3 Distortion

Lens distortion is a form of optical aberration that causes lenses to deviate from rectilinear projection. This phenomena is commonly caused by imperfect lenses and defects from manufacturing. A typical camera will to a varying degree exhibit radial lens distortion. A camera may also exhibit tangential distortion. Other abberations in lenses include chromatic aberration, which is wavelength-specific distortion, and spherical aberration, which is imperfect focusing of light rays incident at lens periphery [19]. Radial and tangential distortion will be discussed in the following sections.

3.3.1 Radial Distortion

Radial distortion takes form as either positive or negative radial distortion, called barrel distortion and pincushion distortion respectively. Illustrations may be seen in Figure 3.3.

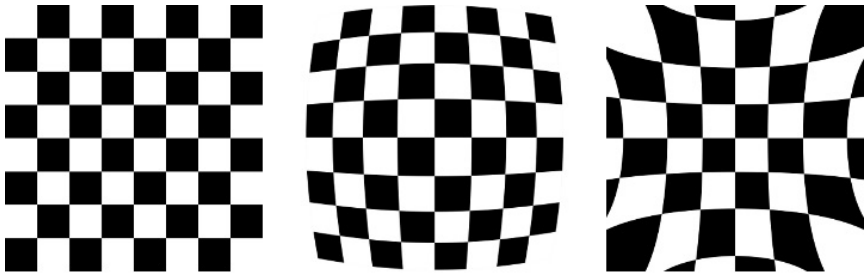


Figure 3.3: Illustration of no distortion, positive radial distortion and negative radial distortion. Originally published in [**?**]

By using the pinhole model, the projection of a point (x, y, z) to the image plane is expressed as (3.2)-(3.3). Radial distortion can be approximated by the following model [20]:

$$\begin{bmatrix} \delta u_i^{(r)} \\ \delta v_i^{(r)} \end{bmatrix} = \begin{bmatrix} \tilde{u}_i(k_1 r_i^2 + k_2 r_i^4 + k_3 r_i^6 + ...) \\ \tilde{v}_i(k_1 r_i^2 + k_2 r_i^4 + k_3 r_i^6 + ...) \end{bmatrix} \tag{3.12}$$

where k_1, k_2, k_3,... are coefficients for radial distortion and $r = (x^2 + y^2)^{\frac{1}{2}}$. Typically, two coefficients are used for correcting radial distortion. Three or more may be used for wide angle lenses or in cases where the camera suffers from severe distortion.

3.3.2 Tangential Distortion

Tangential distortion is caused by imperfections in production of a camera. This can be seen in Figure 3.4. If the image sensor is not perfectly parallel to the camera lens, tangential distortion will occur.

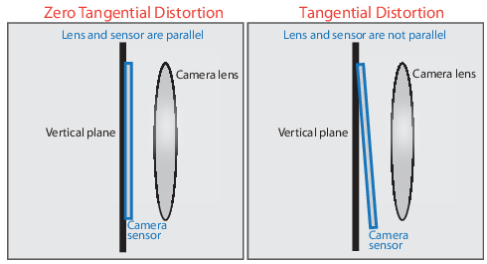


Figure 3.4: Illustration of what causes tangential distortion. Originally published in [21].

Tangential distortion can be modelled as [20]

$$\begin{bmatrix} \delta u_i^{(t)} \\ \delta v_i^{(t)} \end{bmatrix} = \begin{bmatrix} 2p_1\tilde{u}_i\tilde{v}_i + p_2(r_i^2 + 2\tilde{u}_i^2) \\ p_1(r_i^2 + 2\tilde{v}_i^2) + 2p_2\tilde{u}_i\tilde{v}_i \end{bmatrix} \tag{3.13}$$

where p_1, p_2 are coefficients for tangential distortion. The coefficients for tangential distortion will typically have a significantly smaller value than the coefficients for radial distortion.

3.4 Camera calibration

In geometrical camera calibration the objective is to determine a set of camera parameters that describe the mapping between 3D reference coordinates and 2D image coordinates.

Using equations (3.4)-(3.5), (3.12) and (3.13) the camera model for accurate calibration can be modelled as [20]:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \alpha(\tilde{u}_i + \delta u_i^r + \delta u_i^t) \\ \beta(\tilde{v}_i + \delta v_i^r + \delta v_i^t) \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{3.14}$$

where u', v' is the corrected pixel coordinates. While equation (3.14) expresses the projection of 3D points on the image plane, it does not give a direct solution to the back projection problem where line of sight is recovered from image coordinates. When both radial and tangential distortion is considered, there is no analytic solution to the inverse mapping. The inverse mapping can however be approximated. An implicit inverse model is

$$\begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \frac{1}{G} \begin{bmatrix} \tilde{u}_i' + \tilde{u}_i'(a_1 r_i^2 + a_2 r_i^4) + 2a_3\tilde{u}_i'\tilde{v}_i' + a_4(r_i^2 + 2\tilde{u}_i^{2\prime}) \\ \tilde{v}_i' + \tilde{v}_i'(a_1 r_i^2 + a_2 r_i^4) + a_3(r_i^2 + 2\tilde{v}_i^{2\prime}) + 2a_4\tilde{u}_i'\tilde{v}_i' \end{bmatrix} \tag{3.15}$$

where

$$\tilde{u}'_i = \frac{u_i - u_0}{\alpha}$$
$$\tilde{v}'_i = \frac{v_i - v_0}{\beta}$$
$$G = (a_5 r_i^2 + a_6 \tilde{u}_i + a_7 \tilde{v}_i + a_8) r_i^2 + 1$$
$$r_i = \sqrt{\tilde{u}'^2 + \tilde{v}'^2}$$

Comparing equations (3.14) and (3.15), the inverse model has components resembling radial and tangential distortion. The counterparts for the distortion parameters k_1, k_2, p_1 and p_2 are $a_1, ..., a_4$. The parameters $a_1, ..., a_8$ can be solved either iteratively using least squares, when the smallest fitting residual is obtained, or directly, when the result is very close to optimal. In order to solve the unknown parameters for the inverse model, N tie-points $(\tilde{u}_i, \tilde{v}_i)$ and $(\tilde{u}'_i, \tilde{v}'_i)$ covering the whole image area must be generated. Defining

$$\mathbf{u_i} = \begin{bmatrix} -\tilde{u}'_i r_i^2 & -\tilde{u}'_i r_i^4 & -2\tilde{u}'_i \tilde{v}'_i & -r_i^2 - 2\tilde{u}'^2_i & \tilde{u}_i r_i^4 & \tilde{u}_i \tilde{u}'_i r_i^2 & \tilde{u}_i \tilde{v}'_i r_i^2 & \tilde{u}_i r_i^2 \end{bmatrix}^T$$
$$\mathbf{v_i} = \begin{bmatrix} -\tilde{v}'_i r_i^2 & -\tilde{v}'_i r_i^4 & -2\tilde{u}'_i \tilde{v}'_i & -r_i^2 - 2\tilde{v}'^2_i & \tilde{v}_i r_i^4 & \tilde{v}_i \tilde{u}'_i r_i^2 & \tilde{v}_i \tilde{v}'_i r_i^2 & \tilde{v}_i r_i^2 \end{bmatrix}^T$$
$$\mathbf{T} = \begin{bmatrix} \mathbf{U_1} & \mathbf{V_1} & ... & \mathbf{U_i} & \mathbf{V_i} & ... & \mathbf{U_N} & \mathbf{V_N} \end{bmatrix}^T$$
$$\mathbf{p} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \end{bmatrix}^T$$
$$\mathbf{e} = \begin{bmatrix} \tilde{u}'_1 - \tilde{u}_1 & \tilde{v}'_1 - \tilde{v}_1 & ... & \tilde{u}'_i - \tilde{u}_i & \tilde{v}'_i - \tilde{v}_i & ... & \tilde{u}'_N - \tilde{u}_N & \tilde{v}'_N - \tilde{v}_N \end{bmatrix}^T$$

and using equation 3.15, the following relation is obtained

$$\mathbf{e} = \mathbf{T}\mathbf{p} \tag{3.16}$$

The vector $\mathbf{p}$ is now estimated in a least squares in the following expression

$$\hat{\mathbf{p}} = (\mathbf{T^T T})^{-1} \mathbf{T^T e} \tag{3.17}$$

The parameters computed based on (3.17) are used in (3.15) to correct arbitrary image coordinates (u, v). The actual coordinates are obtained through interpolation based on generated coordinates $(\tilde{u}_i, \tilde{v}_i)$ and $(\tilde{u}'_i, \tilde{v}'_i)$.

3.5 Deep learning and neural networks

An artificial neural network is built up of connected layers of neurons, where each neuron has one or more weighted inputs and a bias [22]. All the inputs are summed and processed by an activation. The output of the activation function is the output of the neuron. The structure is illustrated in Figure 3.5

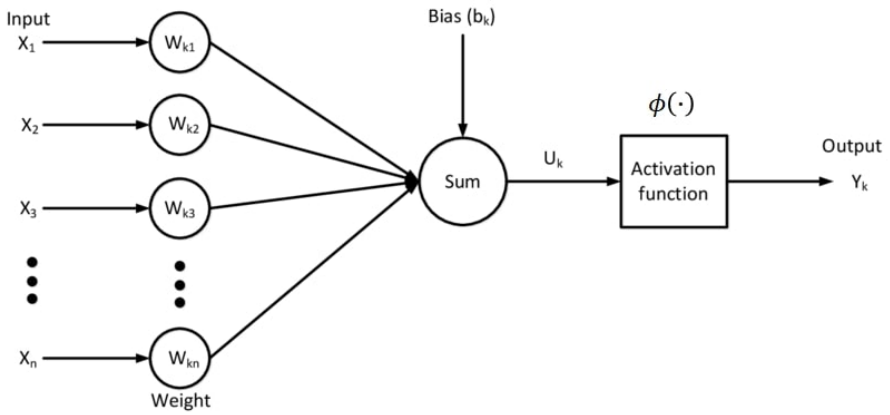


Figure 3.5: The structure of a single neuron in an artificial neural network
Figure taken from [22]

The output of a neuron is

$$y_k = \phi(\sum_{i=1}^{n} x_i w_{k_i} + b_k) \tag{3.18}$$

where ϕ is the activation function, x_i is input i, w_{k_i} is the weight to input i, and b_k is the bias. The subscript k notes that the weights and bias is updated through backpropagation. This is discussed further in section 3.9. The activation function may vary from network to network and within different layers in a network. Two of the most common activation functions, the sigmoid and ReLU functions, are shown in Figure 3.6.

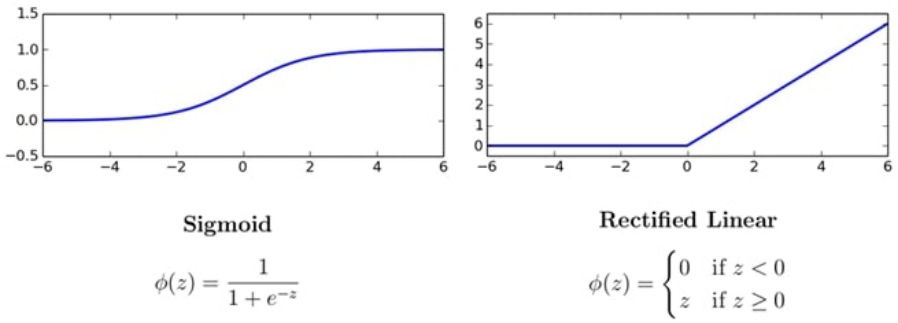


Figure 3.6: Two common activation functions, the sigmoid and the rectified linear. The rectified linear (ReLU) is a linear function where values less than 0 is zero
Figure taken from [22]

The term "network" in artificial neural network comes from the way the neurons are con-

nected together. A simple artificial neural network with two hidden layers is illustrated in Figure 3.7. Hidden layers are layers of neurons that are neither input nor output layers. The illustrated network is a network with only fully connected layers, this means that every neuron in a layer is connected to every neuron in the next.

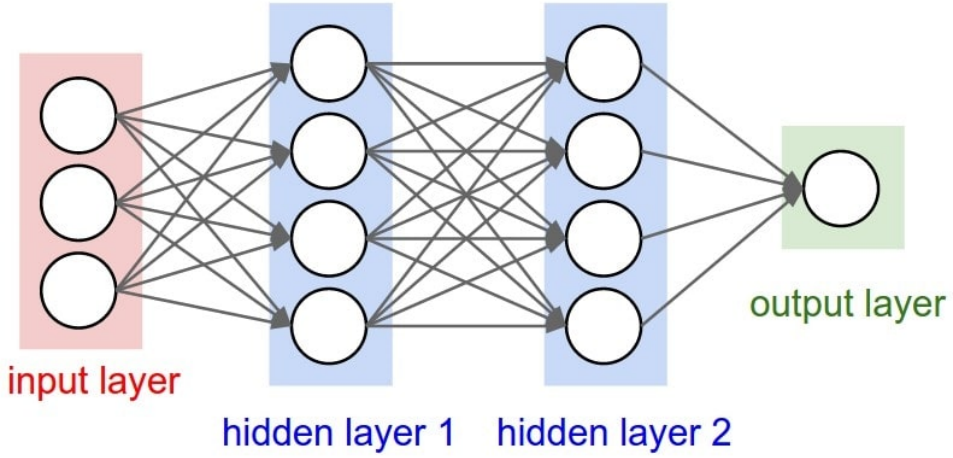


Figure 3.7: A fully connected artificial neural network with two hidden layers.
Originally published in [23]

3.6 Convolutional neural networks

Much like an artificial neural network, a CNN have learnable weights and biases. The way a CNN is distinguishing itself from an ordinary neural network, is that it assumes that the input data has a grid-like topology [23]. In Figure 3.7, the layers are depicted as a one-dimensional vertical line of neurons. In a CNN designed for computer vision, the input layers can be thought of as a three-dimensional grid of neurons. The width and height of the input neurons are represented by the pixels of the image, and the depth is the color channels of the pixels, three for color images, and one for grey-scale images. The network will now look like the one illustrated in Figure 3.8. Each sub-layer, represented by the depth, in a layer, is called a feature map. CNNs uses three basic ideas; local receptive fields, shared weights and pooling.

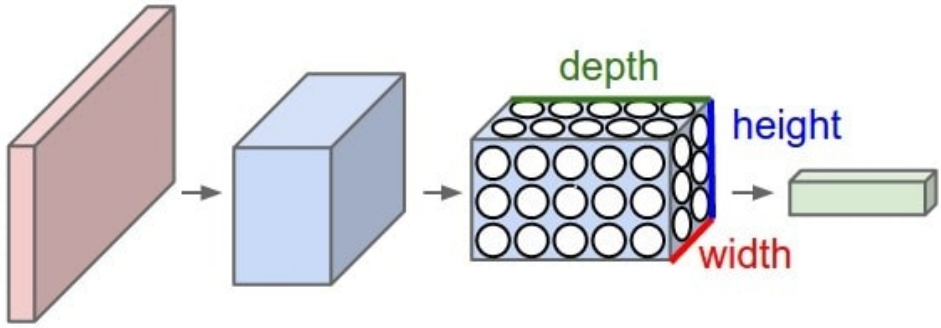


Figure 3.8: Layers of a convolutional neural network represented as three dimensional matrices with height, width and depth
Originally published in [23]

3.6.1 Local receptive fields

When dealing with high-dimensional inputs such as images, there will be an overwhelming number of weights and biases to train if all layers are fully connected. In a CNN, each neuron is only connected to a local region of the input volume, as illustrated in 3.9. This local region is the neurons receptive field [24].

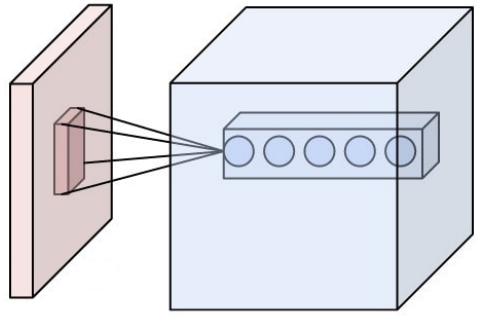


Figure 3.9: A neuron's receptive field in the input layer
Originally published in [23]

The dimensions of the receptive field of a neuron is decided by the filter size in the previous layer, a 5x5 filter yields a local receptive field of 25 neurons. To greatly reduce the number of parameters to train, CNNs uses shared weights and biases, based on one assumption; "If one feature is useful to compute at some spatial position (x,y), then it should also be useful to compute a different position (x_2,y_2)"[23]. This assumption allows each depth slice of a layer to be constrained to use the same weights and bias.

3.6.2 Pooling layers

Pooling layers are commonly used in between successive convolutional layers. The function of pooling layers is to progressively reduce the spatial size of the representation to further reduce the amount of parameters [23]. This amounts to less computational effort required to train the CNN and it helps reduce overfitting. An example of the effect of pooling can be seen in Figure 3.10. Feature maps, or depth, will always remain the same after a pooling operation. However, depending on the filter size and stride, the height and width will be condensed. From this figure, it can be calculated that the filter of the pooling layer is of dimensions 2x2 and stride 2 by using equations (3.19)-(3.21). Pooling layers with filters larger than 2x2 are usually too destructive, i.e. too many features are lost from the feature maps.

$$W_2 = \frac{W_1 - F}{S} + 1 \tag{3.19}$$

$$H_2 = \frac{H_1 - F}{S} + 1 \tag{3.20}$$

$$D_2 = D_1 \tag{3.21}$$

where F is the dimensions of the filter, which is always square. This mean a 2x2 filter yields $F = 2$. S is the stride, which describes how many pixels we slide the filter from one pool operation to the next.

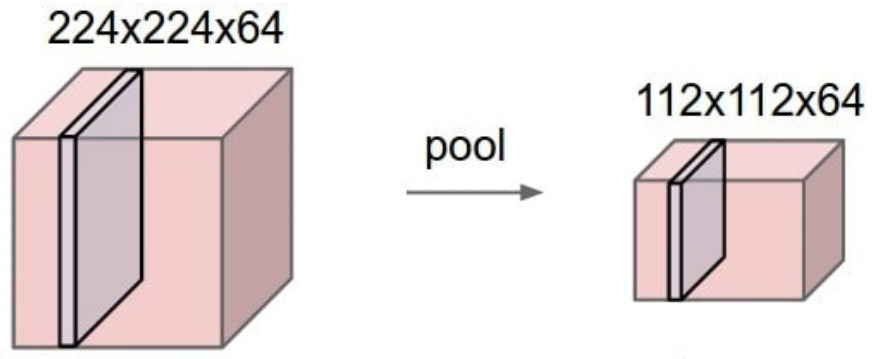


Figure 3.10: The effect of a pooling layer with input 224x224x64.
Originally published in [23]

There exists different types of pooling operations, e.g. max-, average-, and L2-norm-pooling. Because average pooling may "wash out" the features in the feature map, max-pooling usually yields better results in practice. A max-pool operation will output the maximum activation from the receptive field. An example of max-pooling is shown in Figure 3.11.

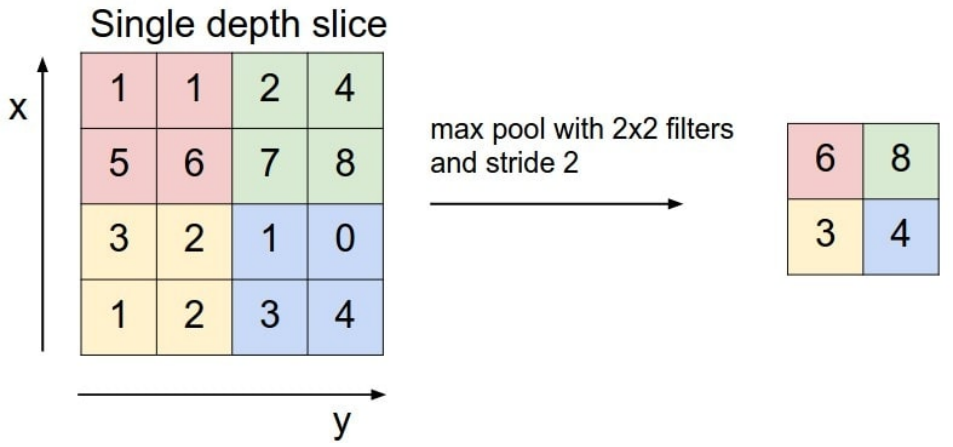


Figure 3.11: Max-pool
Originally published in [23]

3.6.3 Forward and backward pass

While the forward pass in a CNN traverses from input to output calculating all values, the backward pass performs backpropagation with the goal of updating the weights in the network minimizing the error for each output neuron, and the network as a whole [23]. This has originally been done by computing the gradient of a cost-function for a single training example. However, in practice it is common to combine backpropagation with a learning algorithm such as stochastic gradient descent where the gradient is computed for a batch of training examples. Backpropagation through max pooling layers are performed in the manner described by Figure 3.12.

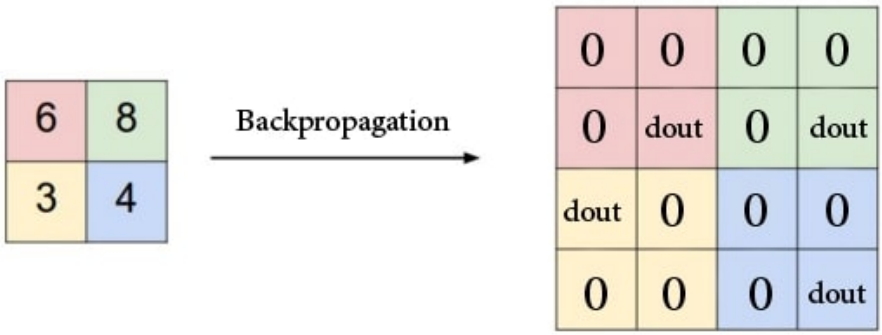


Figure 3.12: Backpropagation in a max-pooling layer
Originally published in [25]

3.7 Detector and models

Single shot multibox detector (SSD) [26] is an architecture using a single deep neural network. What separates SSD from other architectures like Faster-RCNN (Recurrent Convolutional Neural Network) [27] is that it localizes and classifies objects with just one forward pass, hence, single shot. In Figure 3.13, an implementation using VGG-16 [28] as a base, but without the fully connected layers can be seen. These layers are replaced with auxiliary convolutional layers enabling feature extraction at multiple scales and progressively decreasing the size of the input to each subsequent layer [26].

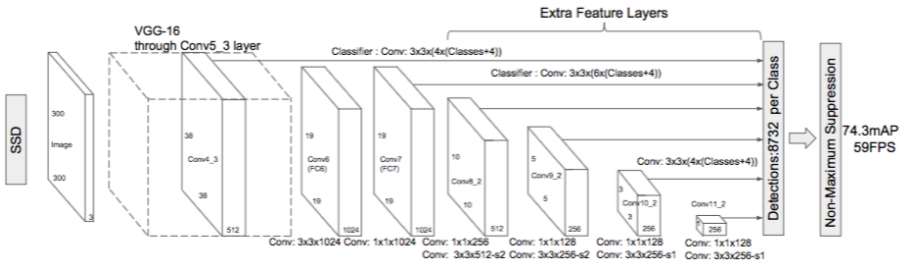
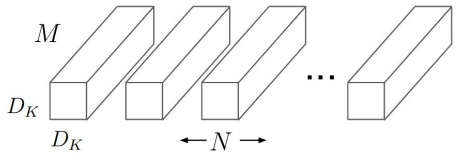
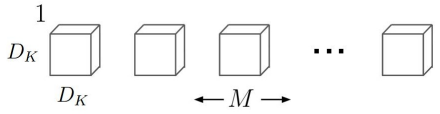


Figure 3.13: Layers of the SSD architecture with 300*300*3 input. The mAP and FPS is the result of the VOC2007 test.
Originally published in [26]

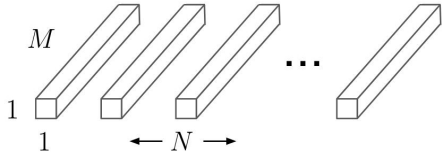
SSD may use several different models as a base instead of VGG-16. The general trend has been designing deeper and more complex networks to achieve higher accuracy. MobileNets [29] is a class of more efficient models with respect to latency and size, at the cost of accuracy. The models are designed to contribute to recognition tasks being carried out in a timely fashion on computationally limited platforms. Instead of using standard convolutional filters, exemplified in Figure 3.14a, MobileNets utilizes a combination of two layers; depthwise- and pointwise convolutional filters as seen in 3.14b and 3.14c respectively. The combination is called a depthwise separable filter. The first layer is however a full convolution.



(a) Standard convolutional filters



(b) Depthwise convolutional filters



(c) Pointwise convolutional filters

Figure 3.14: Different convolutional filters. Originally published in [29]

Compared to VGG's performance on the ImageNet benchmark, MobileNet is nearly as accurate while being 32 times smaller and 27 times less computationally expensive [29]. When implemented for object detection using the SSD 300 framework, mobilenet scores 1.8% lower than VGG on the COCO challenge.

The multibox part of the architectures name comes from the bounding box regression technique. In SSD, every feature map cell is associated with a default set of bounding boxes of different dimensions and aspect ratios [30]. This results in fast computation times of predicting bounding boxes at the expense of pixel wise precision.

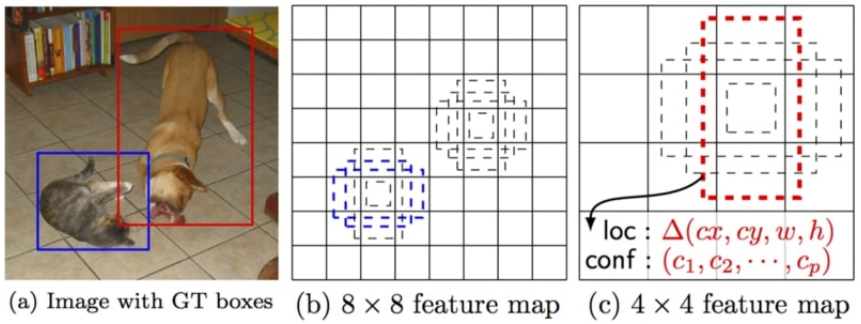


Figure 3.15: SSD default boxes at 8x8 and 4x4 feature maps
Originally published in [31]

The confidence loss L_c measures the networks confidence of the objectness, i.e. how sure the network is that it has correctly classified an object. To compute the confidence loss, categorical cross-entropy is used. This loss function increases as the predicted probability diverges from the actual label. The location loss L_l measures how far away the predicted bounding box is from the ground truth. To compute the location loss the L2-norm, or mean squared error, is used. The expression for the loss, L_m, which states how good a prediction it produced is, can be computed by combining L_c and L_l:

$$L_m = L_c + \alpha L_l$$

Where α is a parameter designed to help balance the contribution of the location loss L_l [32, 33].

3.8 Datasets

When using a CNN for deep learning, it is important to split a labeled dataset into three separate, non-overlapping subsets; a training set, a validation set and a test set. During training on the CNN, it is trained on the training set and validated continuously on the validation set. When testing the performance of the trained CNN, it is tested on the test-set. It is important that all data contained in the test set is unseen data. This is to ensure good generalization and will be discussed more in section 3.9.

A well designed dataset should have good representation of each class and diversity between classes. This will help the model generalize well and maintain a high precision in classification of every class present. A common technique to create increased diversity in the data is data augmentation. Data augmentation can for example be stretching of images to give the objects a slightly different shape or filters can be applied.

3.9 Training

When training a CNN, considerations must be taken. One of the central challenges when training a neural network is overfitting. One of the main strengths of neural networks roots in the universal approximation theorem. No matter what a function f(x) is, there is guaranteed to be a neural network so that for every possible input, the value f(x), or at least a close approximation, is output from the network [24]. If a neural network is trained for too long, the model is prone to overfit the data. That is, approximating the function describing the training data too closely. This leads to bad generalization, usually resulting in reduced performance. Consider the data presented in Figure 3.16. The goal is to separate the blue and red dots, representing 2 different classes. While the green line will separate the two classes with 100% accuracy during training, the better model would likely be the black line. This is because the training set is only examples of real life instances

of objects and in most use cases, the model will encounter variations of the classes trained on.

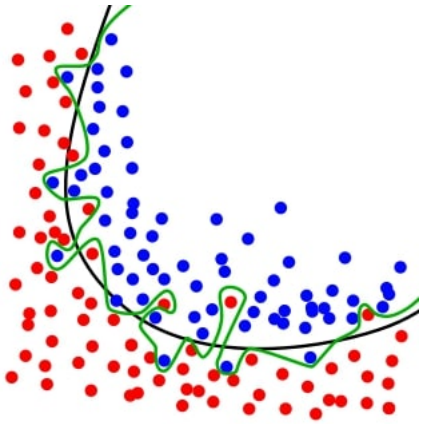


Figure 3.16: The green line represents an overfitted model, while the black line represents a model with good generalization
Originally published in [34]

One method to avoid overfitting would be early stopping [25]. During training, the accuracy of the model tends to continue to increase indefinitely until it converges to near 100% accuracy. However, this is exactly why the data is split up into three separate sets as discussed in section 3.8. The validation set will validate each set of weights generated by the training. Due to overfitting, the validation set accuracy will begin to decrease, i.e. the model is beginning to generalize increasingly poorly as seen in Figure 3.17. This is when to stop training. The best performing model will likely come from the set of weights where the validation set scores the best. Stopping too early will lead to an underfitted model. This may be visualized as trying to separate the dots in Figure 3.16 with a linear model. If the model is underfitting the data, adding more training data will not help. A more complex model, better features or more training are needed. Another method for constraining overfitting is utilizing dropout layers [24]. This is layers that force some of the neurons to deactivate during a particular forward or backward pass. Dropout is usually performed in fully connected layers, where some neurons may develop a co-dependence. Deactivating some of them during training at random leads to higher influence of each remaining neuron in addition to forcing the network to explore new ways of lowering losses. This way, overfitting is reduced.

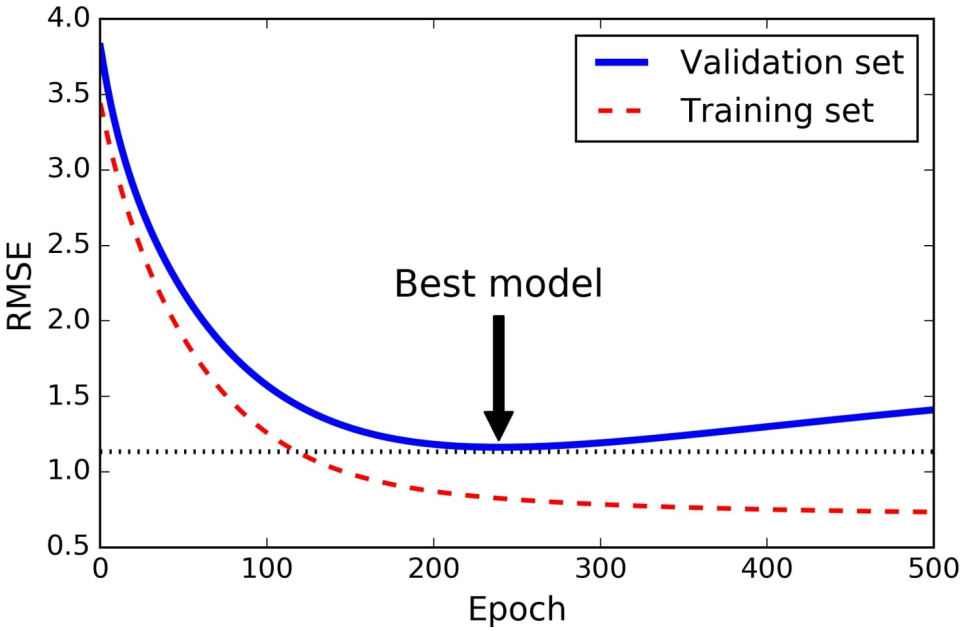


Figure 3.17: The validation set gives us a good idea of when to stop training to avoid overfitting. Root mean square error (RMSE) is a metric describing the error. One epoch is one full iteration of the training set. The numbers on the axis are case dependent.
Originally published in [35]

Another consideration to take when training a CNN is the problem with vanishing gradients [24]. At each backpropagation, each weight are updated proportionally to the partial derivative of the error function with respect to the current weight in each iteration of training. Sometimes, the gradient becomes vanishingly small, and the weight is hindered in learning.

3.10 Image processing

Image processing is a powerful tool for various tasks concerning both geometric calibration of IR cameras and computer vision. Thresholding and edge-map extraction are two methods which have their own strengths and limitations, and will be presented.

3.10.1 Thresholding

Thresholding is a method of image segmentation typically used in grayscale images. In the simplest implementation, the output is a binary image representing the segmentation. In this implementation, each pixel of an image is compared with an intensity threshold

T. For thresholding to work as intended, the image should contain two classes of pixels following a bi-modal histogram, a histogram with only two distinct peaks. An example of a bi-modal histogram can be seen in Figure 3.18.

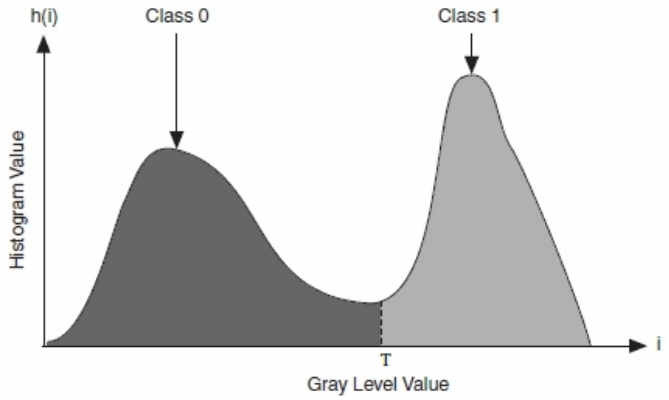


Figure 3.18: A bi-modal histogram allowing for good separability of classes in thresholding. Originally published in [36]

A binary thresholding implementation changing each pixel to either black or white in an 8 bit grayscale image can be seen in (3.22).

$$g(x,y) = \begin{cases} 255\text{ , if } \quad f(x,y) \geq T \\ 0\text{ , if } \quad f(x,y) < T \end{cases} \tag{3.22}$$

where x and y are pixel coordinates, $g(x,y)$ is the thresholded image, $f(x,y)$ is the input image, and T is the intensity threshold. Determining T can be done manually, or techniques like Otsu's method may be used [37]. This algorithm assumes that the image contains two classes of pixels following a bi-modal histogram. An optimal threshold is selected by a discriminant criterion to maximize the separability of the classes. The steps in Otsu's method are presented below. Let the pixels of a given image be presented in L gray levels $[1,2,...,L]$. The number of pixels at level i is denoted by n_i and the total number of pixels by $N = n_1 + n_2 + ... + n_L$. The gray-level histogram is regarded as a probability distribution:

$$p_i = \frac{n_i}{N}, \quad \text{where} \qquad p_i \geq 0, \sum_{i=1}^{L} p_i = 1 \tag{3.23}$$

The pixels are dichotomized into two classes C_0 and C_1 by a threshold level T where C_0 is the set of pixels with levels $[1,...,T]$ and C_1 is the set of pixels with levels $[T+1,...,L]$.

Then the probabilities of class occurence are given by

$$\omega_0 = P(C_0) = \sum_{i=1}^{T} p_i = \omega(T) \tag{3.24}$$

$$\omega_1 = P(C_1) = \sum_{i=T+1}^{L} p_i = 1 - \omega(T) \tag{3.25}$$

and class mean levels are given by

$$\mu_0 = \sum_{i=1}^{T} iP(i|C_0) = \sum_{i=1}^{T} i\frac{p_i}{\omega_0} = \frac{\mu(T)}{\omega(T)} \tag{3.26}$$

$$\mu_1 = \sum_{i=T+1}^{L} iP(i|C_1) = \sum_{i=T+1}^{L} i\frac{p_i}{\omega_1} = \frac{\mu_\tau - \mu(T)}{1 - \omega(T)} \tag{3.27}$$

where

$$\omega(T) = \sum_{i=1}^{T} p_i$$

$$\mu(T) = \sum_{i=1}^{T} ip_i$$

is the zeroth and first order cumulative moments of the histogram up to the intensity threshold T and

$$\mu_{tot} = \mu(L) = \sum_{i=1}^{L} ip_i$$

is the mean level of the input picture. The class variances are given by

$$\sigma_0^2 = \sum_{i=1}^{T} (i - \mu_0)^2 P(i|C_0) = \sum_{i=1}^{T} (i - \mu_0)^2 \frac{p_i}{\omega_0} \tag{3.28}$$

$$\sigma_1^2 = \sum_{i=T+1}^{L} (i - \mu_1)^2 P(i|C_1) = \sum_{i=T+1}^{L} (i - \mu_1)^2 \frac{p_i}{\omega_1} \tag{3.29}$$

In order to evaluate the goodness of the intensity threshold at level T in regards to class separability, the following discriminant criterion is introduced

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \quad \kappa = \frac{\sigma_{tot}^2}{\sigma_W^2}, \quad \eta = \frac{\sigma_B^2}{\sigma_{tot}^2} \tag{3.30}$$

where

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2$$
$$\sigma_B^2 = \omega_0(\mu_0 - \mu_{tot})^2 + \omega_1(\mu_1 - \mu_{tot})^2$$
$$\sigma_{tot}^2 = \sum_{i=1}^{L} (i - \mu_{tot})^2 p_i$$

σ_W^2 are within-class variance, σ_B^2 is between-class variance and σ_{tot}^2 is total variance. Then, this is reduced to an optimization problem to search for the intensity threshold T that maximizes one of the object functions in (3.30). The following relation always holds

$$\sigma_W^2 + \sigma_B^2 = \sigma_{tot}^2$$

As the within- and between-class variance both are functions of intensity threshold T while total variance is independent of T together with between-class variance being first-order statistics, η is chosen to evaluate the separability of the classes at intensity threshold T. The optimal threshold T^* that maximizes η is selected in the following sequential search using (3.24)-(3.27)

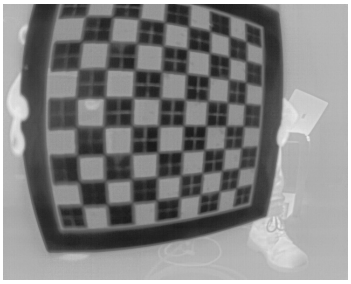
$$\eta(T) = \frac{\sigma_B^2(T)}{\sigma_{tot}^2} \tag{3.31}$$

$$\sigma_B^2(T) = \frac{(\mu_{tot}\omega(T) - \mu(T))^2}{\omega(T)(1-\omega(T))} \tag{3.32}$$

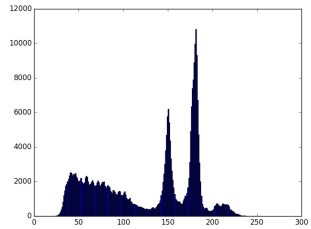
with the optimal threshold T^* being

$$\sigma_B^2(T^*) = \max_{1 \leq T \leq L} \sigma_B^2(T) \tag{3.33}$$

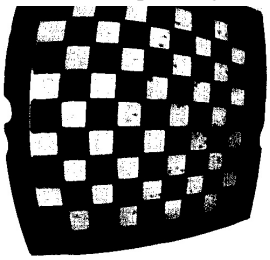
Examples of thresholding, both determining T manually and using Otsu's method, can be seen in Figure 3.19. From the histogram 3.19b of input image 3.19a, it is seen that it is not perfectly bi-modal. However, Otsu's method still works as intended as seen in 3.19d. The same result can of course be obtained by manually determining intensity threshold T, however, this may be time-consuming if multiple images are to be binarized correctly. The binarized image in 3.19c was obtained after multiple guesses of T. The intensity threshold from Otsu's method was $T = 119$, while the manually set intensity threshold is $T = 145$.



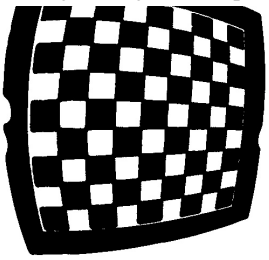
(a) Input image



(b) Image histogram of input image



(c) Manually guessing intensity threshold, $T = 145$



(d) Choosing T using Otsu's method, resulting intensity threshold $T = 119$

Figure 3.19: Examples of thresholding where intensity threshold T is determined manually and using Otsu's method

3.10.2 Edge map

An edge map is based on gradient-based edge detection where image gradients are computed by convolution. Canny edge detection is a common edge detection algorithm, proposed by John F. Canny in [38]. It is a multi-stage algorithm with the first stage being noise reduction [39]. Because edges in images may be weakened by noise, edge detection is susceptible to image noise. Therefore, a Gaussian convolutional filter is applied. The standard deviation σ of the filter can be set high to detect more gradual edges, or low to detect sharp edges. The next stage is finding the intensity gradient of the image. The smoothed image is applied a Sobel kernel horizontally and vertically, obtaining the first derivative G_x and G_y in both directions. From the two resulting images, the edge gradient and direction for each pixel is computed.

$$G_{xy} = \sqrt{G_x^2 + G_y^2} \tag{3.34}$$

$$\Theta = \arctan \frac{G_y}{G_x} \tag{3.35}$$

Note that the gradient direction is perpendicular to edges and is rounded to one of four angles; vertical, horizontal and both diagonal directions.

The next step is non-maximum suppression, a full scan of the image is performed to remove unwanted pixels not constituting an edge. The neighborhood of every pixel is considered to verify if it is a local maximum in the direction of the gradient. This is visualized in Figure 3.20

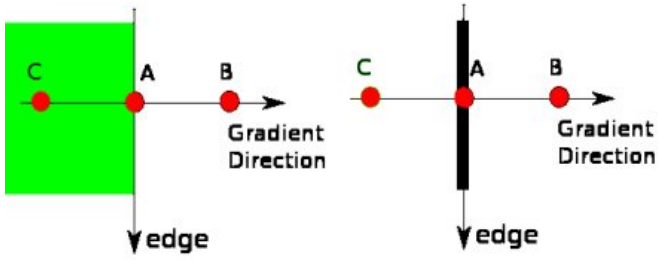


Figure 3.20: Non-maximum suppression visualized. Originally published in [39].

Point A is on an edge in the image between point C and B, which represent high and low intensity pixels respectively creating a gradient normal to the edge. When A is considered a local maximum, it is considered for the next stage; Hysteresis thresholding. Otherwise, it is suppressed and set to zero, i.e. not considered an edge. In the final stage, hysteresis thresholding, all edges from the previous stage are revisited and and classified using two threshold values; a minimum value T_{min} and a maximum value T_{max}. If an edge has a greater intensity gradient than the maximum value, it is considered a sure-edge, while edges with smaller gradient intensities than the minimum value are considered non-edges and are discarded. Edges falling in between the minimum and maximum value are classified as edges or non-edges depending on their connectivity. If an edge falling in between the thresholds is connected to a pixel part of a sure-edge, it is classified as an edge. Otherwise it is discarded. The two thresholds are therefore important to choose correctly depending on the image to get a satisfying result. Examples of different input parameters with identical input picture can be seen in Figure 3.21.

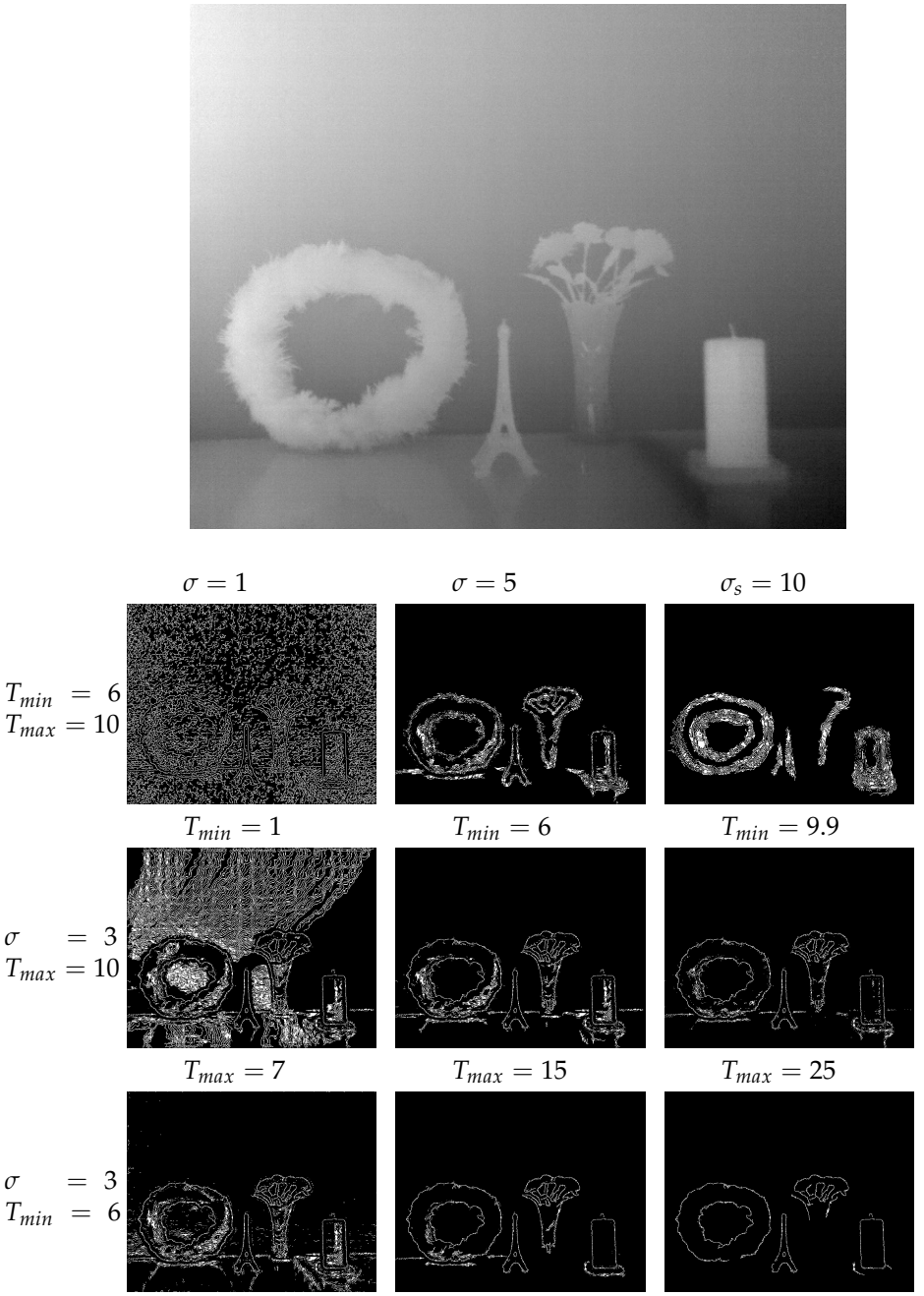


Figure 3.21: Examples of edge maps using different parameters for σ, T_{min} and T_{max}. The effects of increasing one parameter at a time may be seen in rows 1, 2 and 3 respectively. The input image is shown at the top.

Part II

Hardware, calibration and system design

Chapter 4

Sensors and system

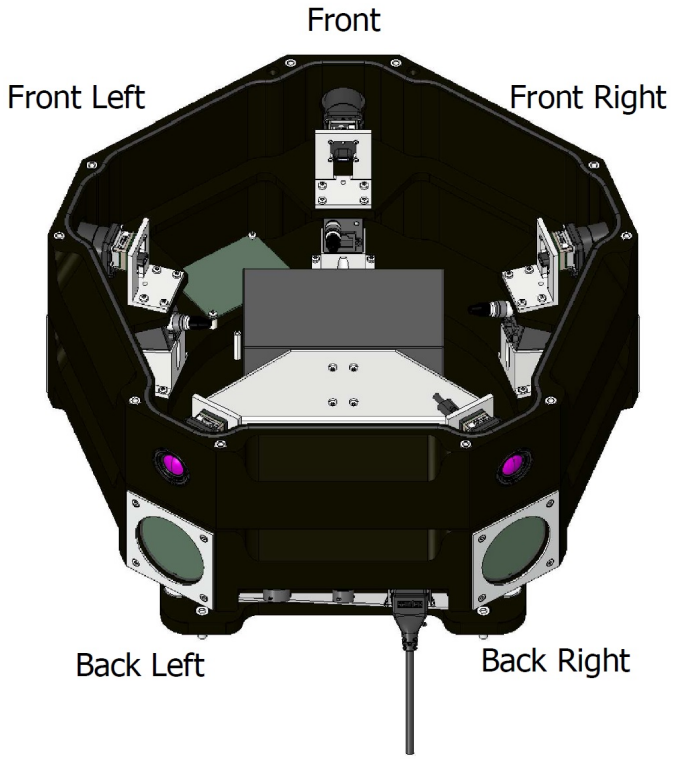


Figure 4.1: 360° sensor rig with 5 FLIR BlackFly 2 EO cameras and 5 FLIR Boson 640 IR cameras. The upper row of cameras are IR cameras and the lower row are EO cameras. Originally published in [40].

4.1 Sensor Rig Layout

Included in the sensor rig are ten cameras, five IR cameras and five EO cameras. They are all connected to a NVIDIA TX2 carrier board. The angle between every neighbouring camera is 72°, such that five cameras together evenly covers 360°. The infrared cameras are interfaced through Video4Linux and configured with FLIRs Boson API [41]. The IR cameras are connected via USB Type-c. The data rate per IR camera at 9 frames per second is 30 Mbit/s. As the IR cameras are interfaced with USB 2.0, with around 480 Mbit/s theoretical and around 330 Mbit/s practical maximum data rate, all 5 IR cameras

may transfer at full speed. The EO cameras have a maximum data rate of 380 Mbit/s at 22 FPS. The maximum total throughput between onboard TX2 and external computer is 1 Gbit/s which must be used for all 10 cameras. This means that unless processing is being done onboard the TX2, then either the frame rate must be restricted, or some form of compression must be applied if all cameras are used [40, 42].

The IR cameras in the sensor rig are FLIR Boson 640 cameras with specifications as shown in table 4.1.

IR Camera	**FLIR Boson 640**
Lens type	Kowa LM6JC
Focal length	4.9 mm
Sensor size	$\frac{2}{3}''$
Image resolution	640 x 512
HFOV according to supplier	95°
Interface	USB 2.0
Max framerate	9 fps
Maximum data rate	30 Mbit/s at 9 fps

Table 4.1: Infrared camera parameters

4.2 GNSS

The GNSS onboard MilliAmpere is a Hemisphere VS330 GNSS receiver. It transmits information to the OBC and receives RTK data via the Satel VHF receiver. This configuration results in 10 mm + 1 ppm horizontal accuracy and 20 mm + 2 ppm vertical accuracy [43]. Its heading accuracy depends on the separation distance of the two antennas, the higher the separation distance, the more accurate the heading accuracy. Hemisphere specify heading accuracy for separation distance up to 5.0m, which is 0.02°. On MilliAmpere, the antennas are separated by almost 2.0 m, resulting in a heading accuracy close to 0.05°.

4.3 IMU

The IMU onboard MilliAmpere is a Xsens MTi-20 with integration level VRU [44]. It has a specified static and dynamic precision in roll and pitch of 0.2° and 0.5° respectively and a resolution of $0.25mG$ with a specified latency of $< 2ms$.

4.4 Coordinate Transformations

To obtain the pose of each IR camera, the coordinates of MilliAmpere have to be transformed. The coordinate transformation to each camera from the center of MilliAmpere is

described in equation (4.1).

$$
\begin{aligned}
{}^{Cam}_{C_O}\boldsymbol{T} &= R_z R_y R_x t_x t_z R_{z_{cam}} t_x \\
&= \begin{bmatrix} c_\psi & -s_\psi & 0 & 0 \\ s_\psi & c_\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\theta & 0 & c_\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\phi & -s_\phi & 0 \\ 0 & s_\phi & c_\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\sigma & -s_\sigma & 0 & 0 \\ s_\sigma & c_\sigma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & A \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_\psi c_\theta c_\sigma + s_\sigma(s_\theta c_\psi s_\phi - s_\psi c_\phi) & c_\sigma(s_\theta c_\psi s_\phi - s_\psi c_\phi) - c_\psi c_\theta s_\sigma \\ c_\theta s_\psi c_\sigma + s_\sigma(s_\theta s_\psi s_\phi + c_\psi c_\phi) & s_\sigma(s_\theta s_\psi s_\phi + c_\psi c_\phi) - c_\theta s_\psi s_\sigma \\ c_\theta s_\phi s_\sigma - s_\theta c_\sigma & c_\theta s_\phi c_\sigma - s_\theta s_\sigma \\ 0 & 0 \end{bmatrix} \\
&\quad \begin{bmatrix} s_\psi s_\phi + s_\theta c_\psi c_\phi & c_\psi c_\theta c_\sigma A + s_\sigma(s_\theta c_\psi s_\phi - s_\psi c_\phi)A + s_\psi s_\theta z + s_\theta c_\theta c_\sigma z + c_\psi c_\theta x \\ -c_\psi s_\phi + s_\theta s_\psi c_\phi & c_\theta s_\psi c_\sigma A + s_\sigma(s_\theta s_\psi s_\phi + c_\psi c_\phi)A + s_\theta s_\psi c_\phi z - c_\psi s_\phi z + c_\theta s_\psi x \\ c_\theta c_\phi & c_\theta s_\phi s_\sigma A - s_\theta c_\sigma A + c_\theta c_\phi z - s_\psi x \\ 0 & 1 \end{bmatrix}
\end{aligned}
\tag{4.1}
$$

where ϕ, θ and ψ denotes the angle of rotation around the x, y and z axis respectively, which is the three rotational degrees of freedom of MilliAmpere. The angles ϕ, θ and ψ corresponds to roll, pitch and yaw respectively. x and z is the translation along the x and z axis to reach the coordinate system at the center of the sensor rig. No translation along y is necessary. Translation along the x axis may be equal to zero depending on the final placement of the platform, but is included in the equation in case it is not. σ denotes the constant rotation about the z axis for each camera, e.g. $\sigma_F = 0^\circ$ and $\sigma_{FR} = 72^\circ$. Finally, the arm $A = 0.2m$ is the translation necessary to translate the coordinate system along the x axis from the center of the sensor rig, to reach each camera. ${}^{Cam}_{C_O}\boldsymbol{T}$ describes each camera's pose according to MilliAmperes pose. This is important to know to correctly identify the position with respect to MilliAmpere of a detected object.

4.5 Robot Operating System

The implementation of the system is done through Robot Operating System, or ROS, an open source framework for writing robot software [45]. It is a collection of tools, libraries and conventions that aims to simplify the task of creating complex and robust robot behavior across a wide variety of platforms.

ROS consists of several parts:

- A set of drivers allowing for data sampling from sensors and issuing commands to motors and other actuators.
- A large collection of fundamental robotics algorithms
- Computational infrastructure that allows moving data around and connect the various components in a system
- Tools for visualizing the state of the system, debugging and recording sensor data and other data.
- A wiki documenting many of the aspects of the framework.

The following paragraphs describe philosophical aspects of ROS:

- *Peer to peer*: ROS systems consist of numerous small programs connecting to one another and continuously exchange messages. There are no central routing system, which means the messages travel directly from one program to another.
- *Tools based*: Many small generic programs may result in complex software systems. ROS does not have a canonical IDE. Tasks such as navigating the source code tree, visualizing system interconnections, graphically plotting data stream etc. are performed by separate programs. This encourages creation of new improved implementations as they can be exchanged for implementations better suited for a particular task domain.
- *Multilingual*: ROS has chosen a multilingual approach, allowing for multiple scripting languages operating together in a single system. ROS software modules can be written in any language for which a client library has been written. The libraries communicate with one another by following a convention that describes how a message has been "flattened" or "serialized" before being transmitted over the network.
- *Thin*: ROS conventions encourages contributors to create standalone libraries and wrap those libraries so they can communicate with other ROS modules.
- *Free and open source*: Both noncommercial and commercial use of ROS is allowed. ROS passes data using IPC, which means systems built using ROS can have fine-grained licensing of their various components.

A simple ROS network can be seen in Figure 4.2. At startup, nodes register with roscore, and then query roscore to find other nodes and data streams by name. The roscore knows what messages every node provides and which it subscribes to. Roscore provides the nodes with adresses of relevant message producers and consumers. The talker and listener node periodically make calls to the roscore while exchanging peer-to-peer messages directly themselves.

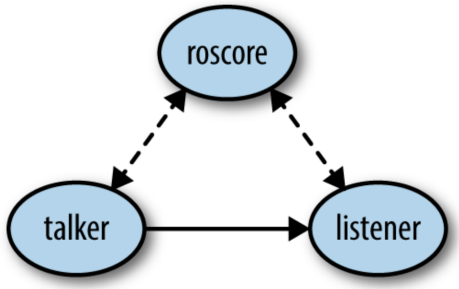


Figure 4.2: Simple ROS network with two nodes and the roscore. Originally published in [45].

Chapter 5

Calibration of IR cameras

The following chapter presents the production method of two calibration fields together with the physical principles on which they rely to create gradients in the LWIR spectrum. The respective calibration procedures used are also presented.

5.1 IR camera calibration

While tried and true calibration methods for EO-cameras exists, IR cameras lacks standardized methods for geometric calibration. Usually, a checkered pattern is printed on a sheet of paper and pictured with the camera which is to be calibrated from several different angles. Due to the properties of infrared radiation however, it is harder to create the gradients needed for corner detection on a two dimensional plane. To overcome this challenge, emissivity differences in materials are utilized. The goal is to make a checkered calibration field allowing standard calibration methods for EO-cameras to be used for calibration of IR-cameras.

5.1.1 Calibration fields

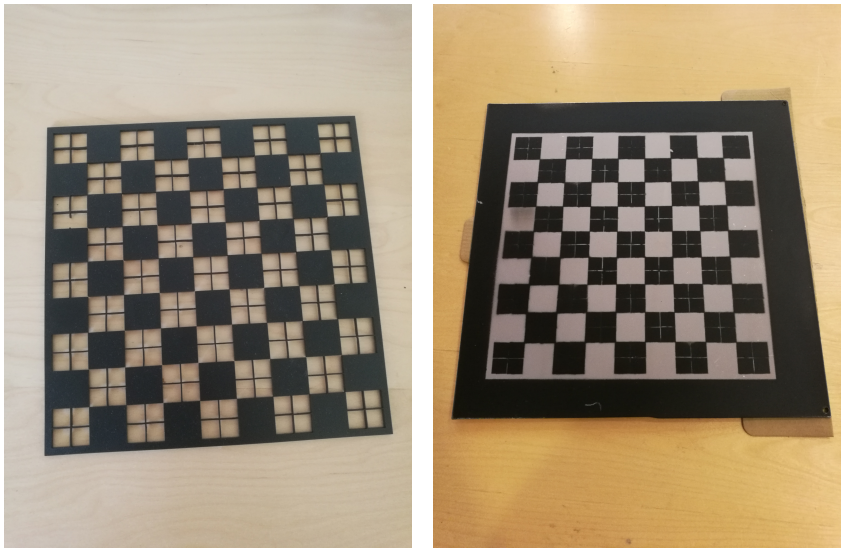
Based on the findings in [7], passive, not actively heated, calibration fields were found to generate higher gradients in the infrared spectrum than active, or heated, calibration fields due to heat spread in the materials and to the surrounding air. The findings in [46] however, showed that the active calibration field used produced good results. To compare and explore strengths and weaknesses in calibration methods using active and passive calibration fields, two methods are proposed. The method using passive calibration field is based on Leslie's cube [2] as well as the work done in [7] where low emissivity materials were used and [6] where the authors cooled the calibration field to create gradients. Despite

the authors of [6] achieving excellent gradients, they did not account for images captured at sharper angles, which because of the three dimensional nature of the calibration field resulted in vanishing gradients. The active calibration field is based on the method used in [46], but with holes with smaller radius as proposed by one of the authors as well as some quality of life changes.

5.1.2 Production and design of passive calibration field

To allow an IR-camera to see the checkered pattern, the difference in emissivity of the materials used should be as large as possible. While highly polished aluminium has a decently low emissivity of 0.039-0.057, nickel-plated (electroplated) copper boasts an even lower emissivity of 0.03 [47]. This means that a nickel-plated copper plate will act almost as a perfect mirror in the infrared spectrum. A gilded copper plate would be marginally better with respect to emissivity. Because of the price however, nickel-plated copper was the material of choice.

The second material needed to create contrast should be as close to a theoretical black body with respect to emissivity as possible. Because of the low cost, practicality and high emissivity of about 0.96, matte black spray paint was the material of choice. To enable a sprayed checkered pattern, a plate of acrylic was laser cut as a more accurate alternative to manually masking the nickel-plated copper plate with tape. Acrylic was the material of choice as it is easily available, reasonably cheap and the fact that it is resilient against expanding when exposed to heat. Because of the nature of the checkered pattern however, none of the pieces are connected. To overcome this, thin crosses was left between each square to hold it all together. The acrylic mask and finished calibration field can be seen i figure 5.1.



(a) Laser cut acrylic mask **(b)** Passive calibration field

Figure 5.1: The mask used for spray painting a checkered pattern along with the finsihed calibration field

5.1.3 Production and design of active calibration field

A wooden plate was coated with Carbo e-Therm ACR120-200A.01, a conductive paint which heats up when exposed to voltage. To create sufficient gradients, a plate was 3D-printed with holes to expose the hot coating. The 3D-printed plate isolates from heat reasonably well allowing for contrasts based on difference in temperature rather than emissivity.

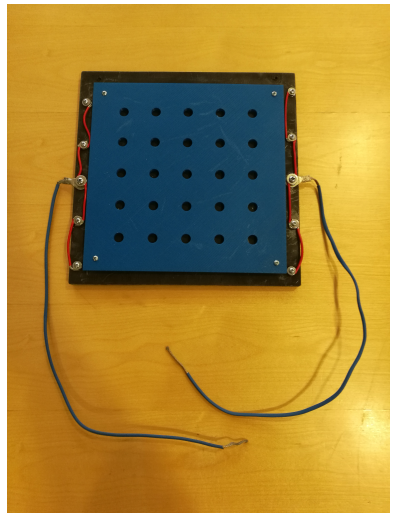
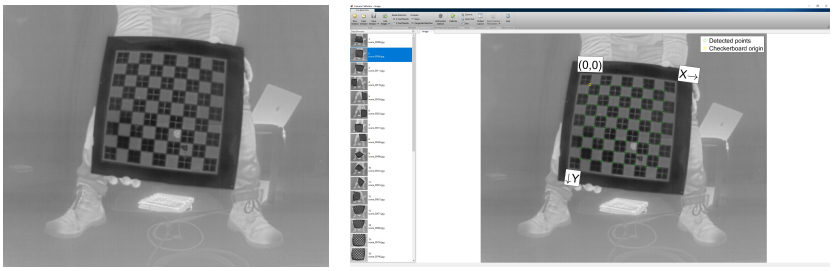


Figure 5.2: Active calibration field. A wooden plate coated with thermally conductive paint with a 3D printed plate attached.

5.1.4 Calibration using passive calibration field

To create gradients between the high emissive paint and low emissive nickel plated copper, the calibration field was cooled in a freezer. Technically, the calibration field could be heated up as well, as long as there is a difference in temperature between it and the surroundings. This would possibly create a shining effect however, which may reduce gradients between the materials, and make corner detection less accurate. The infrared radiation of the surroundings is reflected in the nickel plated copper while the high emissive paint emits radiation according to its own temperature. To gather the images needed for calibration, the field was filmed from different positions and angles, and relevant frames were extracted as .jpg files. The calibration tool used was Matlab camera calibration app [21]. Most of the frames were accepted, and corners were reliably detected correctly. An example frame and corner detection on the same frame can be seen in Figure 5.3.



(a) Frame from calibration set **(b)** Corner detection

Figure 5.3: An example of a frame captured with FLIR Boson 640 and corner detection of the same frame in the Matlab calibration app.

Although frames of the calibration field captured at sharp angles ($\theta \geq 45°$) relative to the camera plane is not advised, and not included in the calibration procedure used, some examples can be seen in Figure 5.5 showing the robustness of corner detection using this method. This is partly due to the calibration field having all its features in the same 2-dimensional plane. Corner detection also worked well for varying distances between the camera and the calibration field.

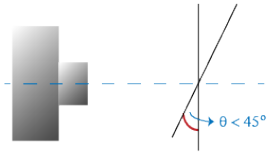


Figure 5.4: Camera angle. Originally published in [21].

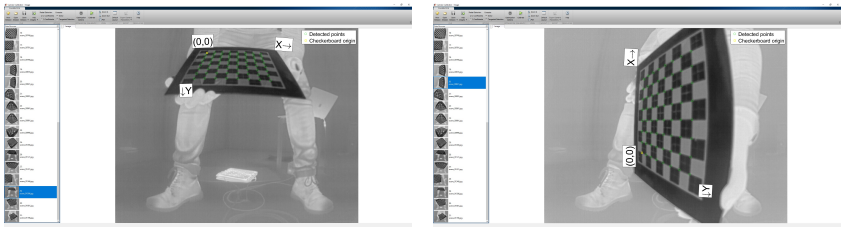
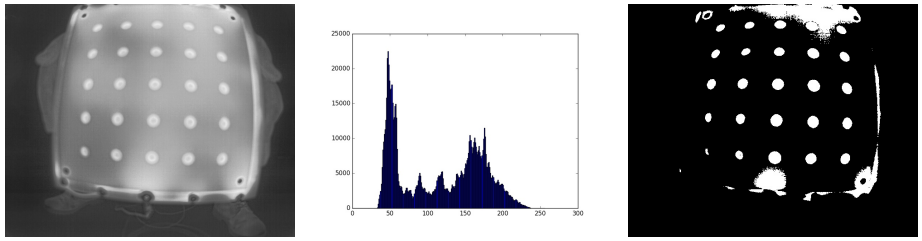


Figure 5.5: Correctly detected corners of the calibration field at sharp angles relative to the camera plane

5.1.5 Calibration using active calibration field

The active calibration field was applied $12V$, allowing it to heat up sufficiently. Frames were gathered in the same way as the passive calibration field. As Matlabs camera calibration tool does not support circle detection, OpenCV calibration tool [48] was used. Circle detection was done using the findCirclesGrid() function. This function did not manage to detect circles correctly directly on the extracted frames, so thresholding was attempted. This was not feasible due to non-bi-modal histograms, which can be seen in Figure 5.6. As there are not just two distinct tops, the histogram is not bi-modal and thresholding may

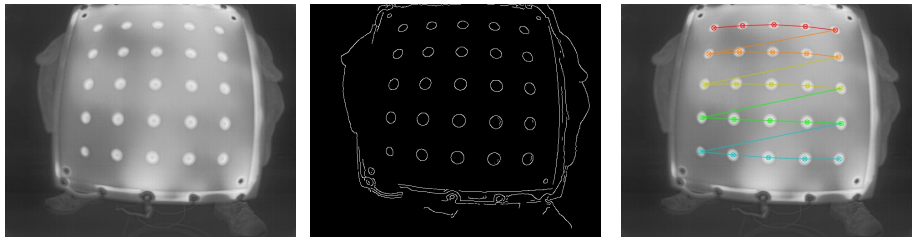
not work as intended. This is due to uneven heat signature where the calibration field were supposed to have a uniform heat signature significantly lower than in the holes.



(a) Original frame **(b)** Image histogram **(c)** Threshholding

Figure 5.6: The original frame, its histogram and resulting binary image segmented using thresholding

Extracting the edge-map however, enabled correct circle detection. An example can be seen in Figure 5.7 where the original frame is the same as in 5.6a. Extracting the edges and creating an edge-map does not require the image histogram to be bi-modal as the operation uses local gradients rather than a global intensity threshold.



(a) Original frame **(b)** Edge map **(c)** Circle detection

Figure 5.7: An example frame of the active calibration field, its edge-map and circle detection based on the edge map 5.7b projected onto the original frame 5.7a

This method was however less robust in respect to point detection, especially when the calibration field was in the edges of the frames where the radial distortion is highest. This is a problem because it is advised to include frames with the calibration field positioned near the edges for calibration. Additionally, this method was not robust to changes along the optical axis of the camera or changes in angles relative to the camera plane. Thus, most of the pictures accepted for calibration were at similar distance along the optical axis, near the center of the frame and normal to the camera plane.

Chapter 6

ROS pipeline

The full pipeline from image capturing to position estimation is implemented in ROS. The pipeline is distributed over three nodes to obtain modularity. The following sections will present each nodes function, and the reasoning behind it. An illustration of the system can be seen in Figure 6.1. The existing ROS network implemented on MilliAmpere can be seen in Appendix B.

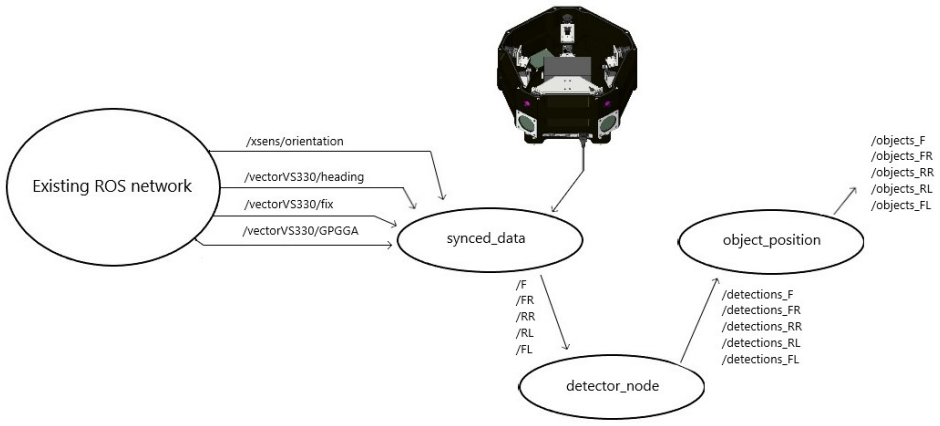


Figure 6.1: An illustration of the different nodes in the ROS network

6.1 Node 1 - Data sampling

The function of node 1, synced_data, is to grab a frame from the live stream of the IR cameras using GStreamer [49] together with data from GPS and IMU to enable synchronization of the image frame with global position, heading, roll and pitch of the vessel. This is important to enable accurate calculation of each cameras pose in the world and body frame. If the synchronization is off, a camera's actual pose may deviate from the calculated pose. Therefore, calculation of detected objects position in respect to the camera, explained in section 6.3, may not coincide with the objects actual position in respect to the camera.

As of time of writing, FLIR Boson 640 does not support external frame synchronization or a time stamp feature. Due to this, time delays have to be estimated, and buffered together with data from GPS and IMU. Fortunately, FLIR specifies the delay for the Boson 640 camera, which is 18-37ms [50] depending on camera settings. Additionally, the images are undistorted using the intrinsic parameters and distortion coefficients found in chapter 5. The node subscribes to topics from other, preexisting nodes in the ROS-network. The node publishes the undistorted frames, along with camera position, camera orientation and gps-time in a Detection2D message. It publishes five topics, one for each IR camera.

6.2 Node 2 - Object detection

With modularity and the rapid advancement in computer vision in mind, object detection is done in a separate node- detector_node. This enables easy replacement with better performing detectors or trained models in the future. The node uses an SSD detector described in 3.7 together with mobilenet, a lightweight model which may enable real time object detection without GPU accelerated computation with the trade off being performance in respect to accuracy.

A Tensorflow implementation based on [51] is used and a mobilenet model was trained using a custom classes with the dataset made in [1]. Despite [1] having shown that training a model with a dataset containing histogram equalized and bilaterally filtered IR images along with gray-scale images in the visible spectrum may increase performance, this method was not used. This is due to the computational expensiveness of the image processing in question, which needs to be applied to every input frame in real time for the detector to take advantage of this training method. As there is five IR cameras which needs to run in parallel with object detection performed at a sufficient frame rate, in addition to five EO cameras, the model was trained using normalized IR-images, which require no real time computationally expensive image processing.

This node subscribes to the topics published from node 1. When a detection is ready, it will publish the coordinates for the bounding boxes, predicted class, confidence rate, camera orientation, camera position in a NED frame and which camera the frame was captured from. Publishing the image itself is not necessary and would apply an unnecessary load to

the system. The data is published in a Detection2Darray message. Once again, it publishes one topic per IR camera.

6.3 Node 3 - Calculation of object position

Node 3 subscribes to the topics published in node 2. It uses the received data to calculate a body-ray to the detected objects position with respect to the camera. This is done using

$$r = \kappa^{-1} \begin{bmatrix} x & y & 1 \end{bmatrix}^T \tag{6.1}$$

where κ is the camera matrix containing the intrinsic parameters and x and y are the pixel coordinates of the center pixel of the bottom of the bounding boxes published in node 2. The bottom is chosen as this will be the the world point closest to the camera, and may be used to estimate distance with the mono camera setup. r is a body ray from the optical center of the camera to the normalized image plane. From this, the angle between the cameras optical axis and the detected object can be calculated. If distance to the detected object is known, a scalar s may be multiplied with r to enable calculation of object position in the camera frame, and therefore the NED frame. The node publishes one topic for each camera in a Detection2Darray.

Part III

Experiments, results and discussion

Chapter 7

Conducted experiments

Several experiments were conducted to test the both individual components together with the whole system. Descriptions of the experiments and the motivation behind them will be presented in the following chapter.

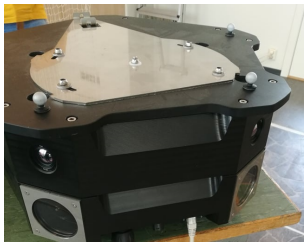
Figure 7.1: Target vessel Telemetron to the left and MilliAmpere to the right

7.1 Camera calibration

Firstly, the calibration accuracy of each calibration method have to be determined. To quantify the precision of the camera calibration, "slangelabben" at NTNU were used. This is a controlled environment equipped with a motion capture system using 16 cameras, reporting the position and orientation of objects equipped with special markers. With known camera pose in the same coordinate system, this enables accurate world point to pixel projection which enables calculation of pixel error between projected pixel and the pixel where the target is positioned in the image frame. Both the camera and target needs to be marked with at least 3 markers to accurately extract pose. The target will be imaged with several different poses in respect to the camera so that it appears in multiple positions in the image.

Figure 7.2: "Slangelabben"

Markers were placed on the sensor rig with one marker above the tested camera's microbolometer sensor as shown in Figure 7.3a. The target used for imaging was practically invisible in the infrared spectrum, thus, the target was heated with a hairdryer. An example can be seen in Figure 7.3, where the markers are clearly visible with an EO-camera and only the heated marker can be seen in the image from the LWIR sensitive camera.



(a) Camera markers

(b) Target markers

(c) Uncalibrated image in LWIR spectrum with one of the markers heated.

Figure 7.3: Comparison of the same target in visible and LWIR spectrum

Using a program called Motive:Tracker, the position of every marker can be extracted, and multiple points can create a rigid body, which enables orientation extraction. A screenshot from the program is presented in Figure 7.4. The green rigid body consisting of 3 markers represents the camera. The position of the camera is extracted as the marker above the microbolometer, with the estimated height difference of 0.06 m subtracted. The blue rigid body represents the target. Technically, only one point was needed. Due to some objects appearing as points however, it was easier to identify in the program after merging the three markers to a rigid body.

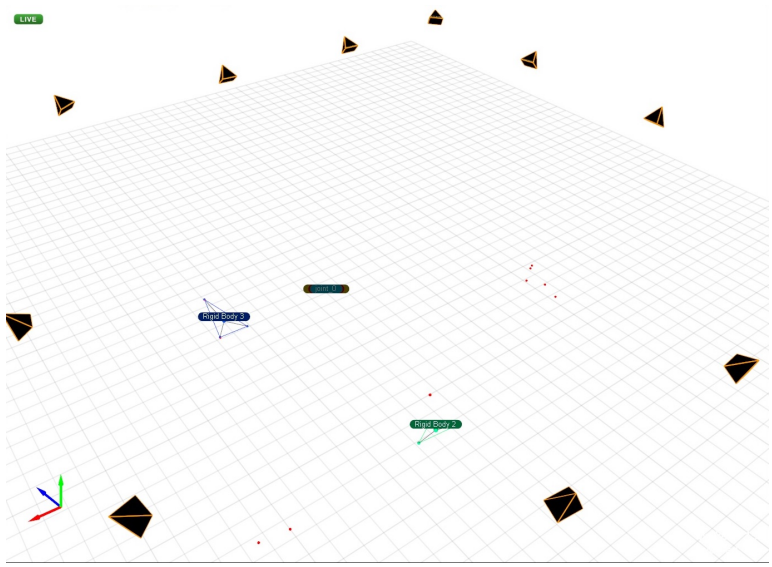


Figure 7.4: A screenshot from Motive:Tracker

7.2 Brattøra experiments

To test the system in practice, several experiments were conducted. Experiments were conducted in Brattørabassenget in Trondheim. The sensor rig was mounted on top of MilliAmpere. The target vessel was Telemetron, the red boat in the picture in Figure 7.1. Both vessels has known longitude and latitude coordinates together with GPS time. This combined with MilliAmpere having known attitude and camera position enables several important experiments.

For the sake of the experiments, the target vessel will be manually pointed out rather than having a detector make bounding boxes to increase accuracy. This is because it is important to know where the GPS antenna is on the target vessel, such that it can be pointed out in the images so it corresponds with the logged longitude and latitude coordinates on the vessel.

By utilizing equations (3.10)-(3.11), an object in the 3D NED frame can be projected onto the 2D virtual image plane. In the following experiments, the goal is to identify what causes potential deviations between the projected 2D virtual image coordinates and the actual image coordinates where the target is. All topics published by the ROS network of MilliAmpere was recorded using rosbags. Each following experiment were conducted twice. The experiments differ by the intrinsic matrix and distortion coefficients used for image undistortion and perspective projection. The results from the experiment in 7.1, i.e. which calibration method yields the better result, decides which datasets is to be analyzed in the following experiments.

7.2.1 Experiment 1: Stationary imaging

To test the two calibration methods in practice, both vessels were stationary to negate potential desynchronization in time stamps. Frames of Telemetron were captured both in the center of the image to get a reference and in the very edges where the images originally suffers the most from radial distortion. This experiment was conducted with intrinsic matrices and distortion coefficients from both cameras. The goal of this experiment is to quantify errors due to distortion and transformation from NED frame to the camera.

7.2.2 Experiment 2: Data synchronization

To test data synchronization, Telemetron was again stationary. MilliAmpere, while maintaining near constant NED position, was spinning around its own axis with approximately constant angular velocity. This enables comparison of ground truth data, i.e. ground truth longitude and latitude, and GPS time for both vessels with where the target vessel appears to be in the image frame. The purpose of this experiment is to quantify the potential latency between the moment the image was captured and when it was sampled by the ROS network.

7.2.3 Experiment 3: Fast changing camera pose

To stress test the system, both in terms of data synchronization, coordinate transformation and camera calibration, fast changes in roll, pitch and heading were generated by maneuvering and rocking MilliAmpere. The quick changes in attitude should reveal any desynchronization present in the data. The rocking of MilliAmpere makes all six degrees of freedom come into play in the transformation from NED to each camera, which will reveal potential inaccuracies. Due to the pose of the cameras when MilliAmpere is exposed to high values in roll and pitch, images with Telemetron in closer to the corners of the image frame were captured.

Chapter 8

Results and discussion

In the following chapter, results from the two calibration methods along with results from experiments conducted in Brattørabassenget and object detection will be presented and discussed.

8.1 Camera calibration

The results from the calibration of the IR cameras using the passive and active calibration fields are discussed. This includes distortion, intrinsic parameters, reprojection error and pixel errors. All parameters can be seen in appendix A.

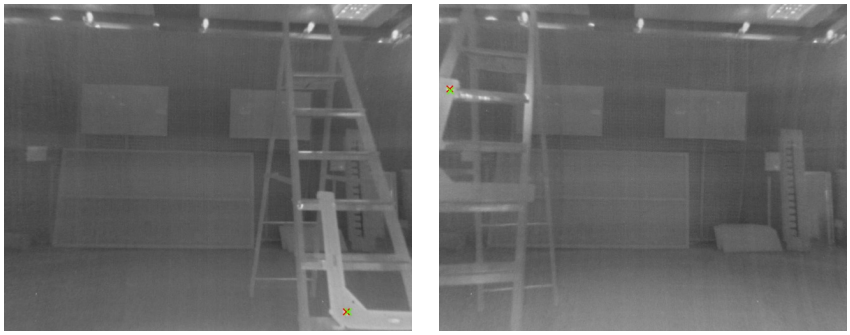
8.1.1 "Slangelabben"

The pixel accuracy of the rear right camera is validated on an image set with 10 different target positions. Most of the targets were positioned near the edges of the image frame. The images are undistorted using both calibration methods, resulting in two sets of images. Thus, the results are based on the same original images for a fair comparison. The images presented in the following section have a red and a green cross. The red cross represents the pixel position of the seen target in the image, while the green cross represents the projected three-dimensional target coordinates projected onto the image.

Passive calibration field

The method utilizing the passive calibration field yielded an average pixel error of $\overline{\Delta x} = \overline{\Delta y} = 1\ px$ in both x and y direction. The average error in pixel length was 1.91 px. The

images with the largest errors in x and y are presented in Figure 8.1. The horizontal error in 8.1a is $\Delta x = -3$ while the vertical error in 8.1b is $\Delta y = -3$. In both images, the target was positioned near the corners of the image.



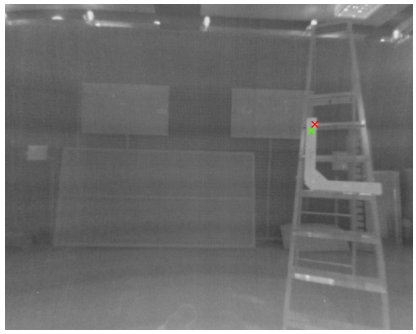
(a) Largest horizontal error in the image set of $\Delta x = -3$

(b) Largest vertical error in the image set of $\Delta y = -3$

Figure 8.1: The image frames with largest errors in both directions using intrinsic parameters and distortion coefficients from passive calibration.

Active calibration field

The method utilizing the active calibration field yielded an average pixel error of $\overline{\Delta x} = 2\ px$ horizontally and $\overline{\Delta y} = 9\ px$ vertically. The average error in pixel length was $10.20\ px$. The images with the largest errors in x and y are presented in Figure 8.2. The horizontal error in 8.1a is $\Delta x = 6$ while the vertical error in 8.1b is $\Delta y = -14$. Notice that while in 8.1b, the target is close to the upper edge of the image frame, while in 8.1a, the target is closer to the middle, where the distortion originally is moderate. This may be due to inaccurate principal point, which differs greatly from the estimated principal point from the calibration method utilizing the passive calibration field. Using the active calibration field, the principal point is calculated to be $(u_0, v_0) = (317.7474, 260.9770)$ while it from the passive calibration field is $(u_0, v_0) = (319.8949, 251.5129)$. The deviance in the vertical component of the principal point contributes to the vertical error in both images in Figure 8.2.



(a) Largest horizontal error in the image set of $\Delta x = 6$

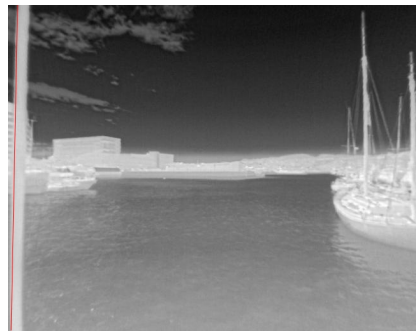
(b) Largest vertical error in the image set of $\Delta y = -14$

Figure 8.2: The image frames with largest errors in both directions using intrinsic parameters and distortion coefficients from active calibration.

8.1.2 Distortion

A comparison between the two calibration routines may be seen in Figure 8.3, where two images have been taken with the rear right camera. Image 8.3a are undistorted with the intrinsic parameters and distortion coefficients from the calibration method using the active calibration field while image 8.3b used results from the method utilizing the passive calibration field. Consider the antenna to the very left in the images. This antenna is straight, and should appear straight in the image if the camera is perfectly calibrated. In image 8.3a however, it can be seen that the camera still inhibits some distortion after calibration. Comparing it to image 8.3b, the antenna appears almost perfectly straight as the left edge is almost parallel to the straight red line.

(a) Image from rear right camera calibrated using active calibration field



(b) Image from rear right camera calibrated using passive calibration field

Figure 8.3: A comparison between images captured with the same camera, but calibrated using different methods

8.1.3 Reprojection Error

Due to the distortion itself, the calibration method using the active calibration field struggled with circle detection, resulting in an image set lacking accepted images with the calibration field positioned near edges and corners of the image frame. While this have affected the distortion coefficients and intrinsic parameters, it has directly affected the reprojection error in a certain way. This can be seen when comparing the two calibration methods. The average reprojection error using the passive calibration field is 0.2403 while the average reprojection error using the active calibration field is 0.03288. The method using the active calibration inhibits 13.68% overall reprojection error of the reprojection error using the passive calibration field. While a lower reprojection error may seem like a property belonging to the superior method, this is not always the case. The reprojection error is calculated from the same set of images used for calibration, thus, if the there are few images with the calibration field positioned in the image frame where the radial distortion is highest, the radial distortion coefficients may be inaccurate, and still provide low reprojection error. This is exemplified by the distortion map in 8.4, where the pixel shift is visualized. If every detected point is in the middle of the image frame, almost no pixel shifting is necessary, and the reprojection error will consequently be low. Despite the reprojection error during calibration being significantly lower for the active calibration field than for the passive calibration field, the method using the passive calibration field together with Matlab calibration tool yielded the better result.

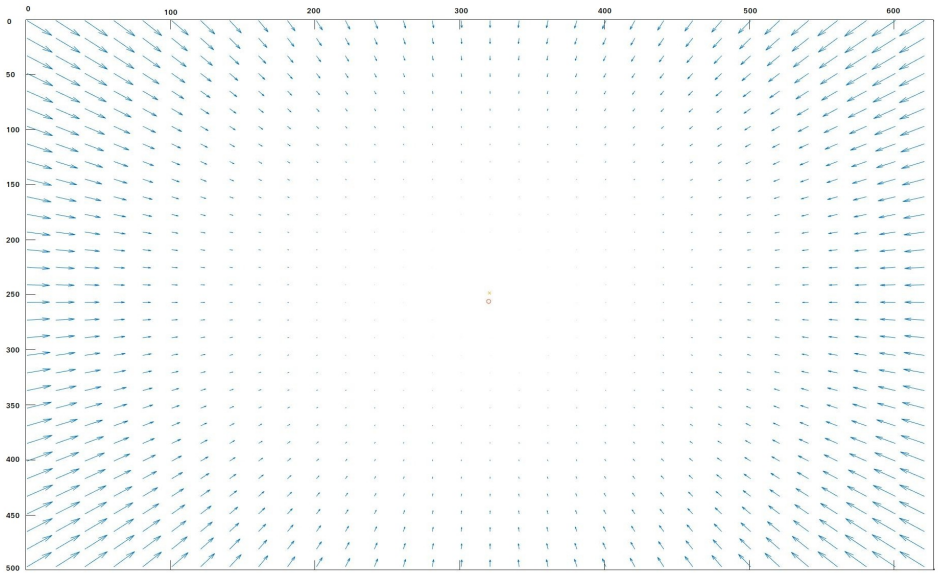


Figure 8.4: Distortion map for front facing IR camera using the passive calibration field. The red circle marks the center of the image, while the yellow cross marks the calculated principal point. The further away from the principal point, the higher the effect of radial distortion. The corners suffers the most.

8.1.4 Discussion

Accurate calibration is important in this use case. Objects close to MilliAmpere will appear in the lower edge of the image frame where the pre-undistorted image suffers from radial distortion. Objects close to MilliAmpere may be the most critical objects to classify and locate correctly. The same objects may also be the hardest for the detector to detect as the trained models learned features may not coincide with the potentially distorted features present in the image. If the IR cameras inhibits sufficient distortion, it may also result in incorrect body rays from MilliAmpere to an object, potentially causing dangerous situations.

With an average error of 2.32 px, the calibration method utilizing the passive calibration field yielded more precise calibration than the method using the active calibration field, which had an average error of 9.91 px. The uncalibrated camera had an average error of $\overline{\Delta x} = 14\,px$ and $\overline{\Delta y} = 11\,px$ in x and y direction respectively and a total average error of 21.23 px in pixel length. The points were projected onto the distorted image using intrinsic parameters from the method using the passive calibration field as this yielded results closer to ground truth. While active calibration had low errors in x, the error in y was substantial, and close to the vertical pixel error of the uncalibrated camera. Using the camera calibrated using the passive calibration field should in a worst case scenario inhibit a margin of error of 0.26° horizontally and 0.33° vertically based on the results from 8.1.1.

8.2 Brattøra experiments

From the results in 8.1, the dataset using parameters from the passive calibration field is analyzed. When calculating the error between the actual pixel coordinates of Telemetron and the pixel coordinates obtained from perspective projection, a pixel needs to be selected as the position of Telemetron as it will occupy multiple pixels. Telemetrons longitude and latitude is centered at the GPS-antenna at the rear end illustrated in figure 8.5.

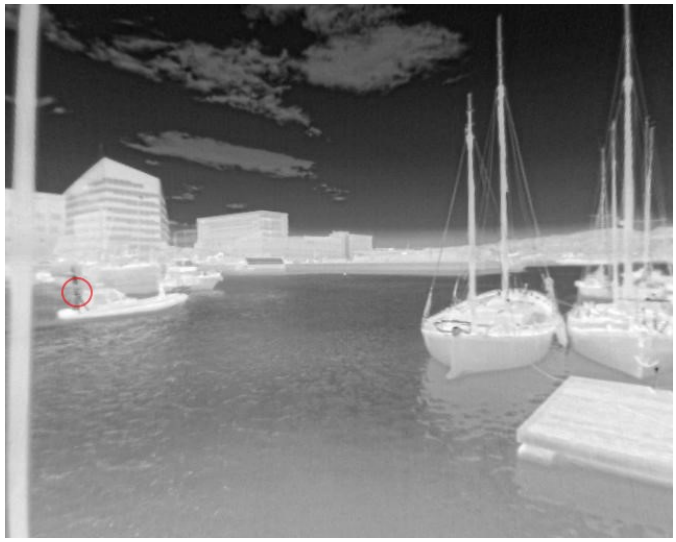


Figure 8.5: Telemetrons GPS-antenna marked by the red circle

This coordinate is transformed into the same NED frame as MilliAmpere, illustrated in figure 8.6. The height of the antenna is estimated to be $0.5m$ above the deck of MilliAmpere, which is $z = 0$ in the NED frame. The pixel selected is an attempt at aiming at the GPS-antenna. This will be an uncertainty in this experiment as it is hard to point out the right pixel, especially at distance. Another uncertainty may be the accuracy of the zeroing of MilliAmpere's orientation. The biggest uncertainty however would be desynchronization between measurements. As the GPS connection to the sensor rig was not ready, image timestamps with GPS time were not available.

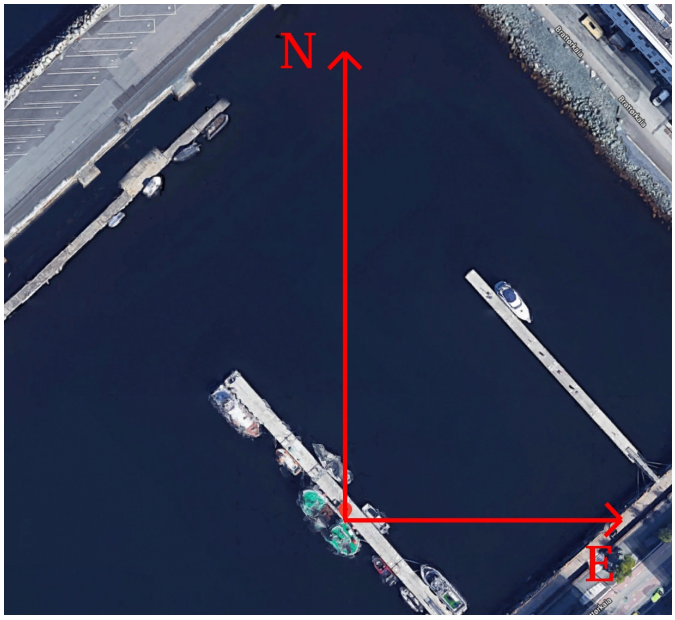


Figure 8.6: The NED frame in which MilliAmpere, its cameras and Telemetron coordinates are represented. Origo is located at the pier as marked.

The following images presenting results from the experiments will contain a set of crosses, one red and one green. The red cross represents the estimated location of Telemetrons GPS antenna in the image, while the green cross represents the three dimensional NED coordinate of Telemetron projected onto the image. The image coordinate system has its origin located in the top left corner as illustrated in Figure 8.4

8.2.1 Experiment 1: Stationary imaging

Keeping both vessels stationary for this experiment was challenging due to high wind speed. Thus, some unwanted changes in orientation were introduced along with positional changes. Due to none constant camera orientation and position, potential desynchronization of sensor data, especially the delay present from image grabbing may affect the results. From inspection of the data gathered, the yaw and position of the cameras were fairly constant. Based on this observation, the horizontal pixel error between projection and image should be weighted heaviest, as roll and pitch which may contribute to vertical error were harder to counteract.

A selection from the results using the distortion coefficients and the intrinsic matrix found by using the passive calibration field is presented below. From the example in figure 8.7, the observed pixel value was $(x, y) = (312, 281)$ whereas the projected pixel was $(x, y) = (308, 244)$. A vertical pixel error of 37 px was one of the highest in this experiment and corresponds to an error in angles of $5.21°$, or 4.69 m at the current distance. This is because

of the fairly high camera pitch rate due to wind and waves. This may imply that there is a latency in some of the data.

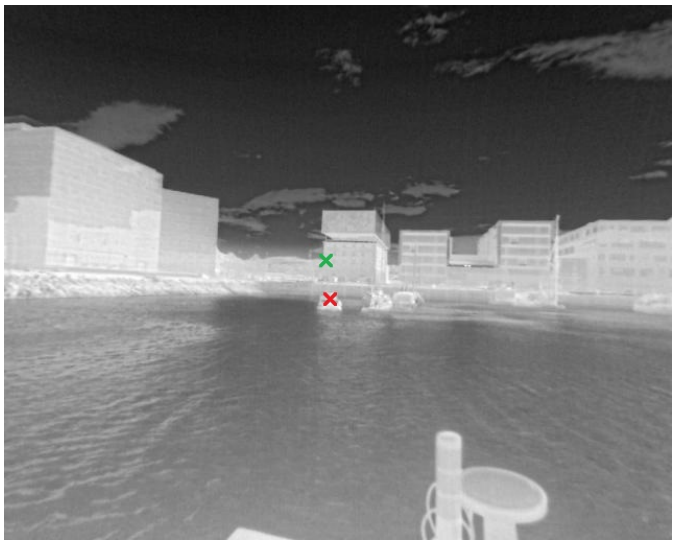
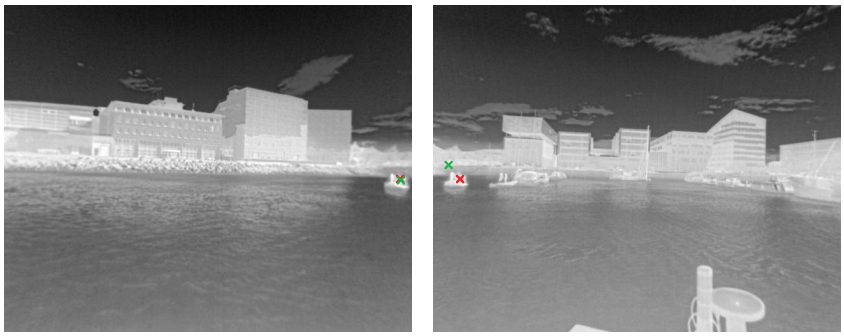


Figure 8.7: Telemetron captured in the center of the image frame. This minimizes the effects of distortion and camera roll and establishes a baseline for the experiment.

The examples in 8.8 are captured at the same point in time, with two different cameras. The horizontal and vertical pixel error in 8.8a is 1 px and 2 px respectively, while the error in 8.8b is 18 px and 20 px.



(a) Image captured by the front camera **(b)** Image captured by the front right camera

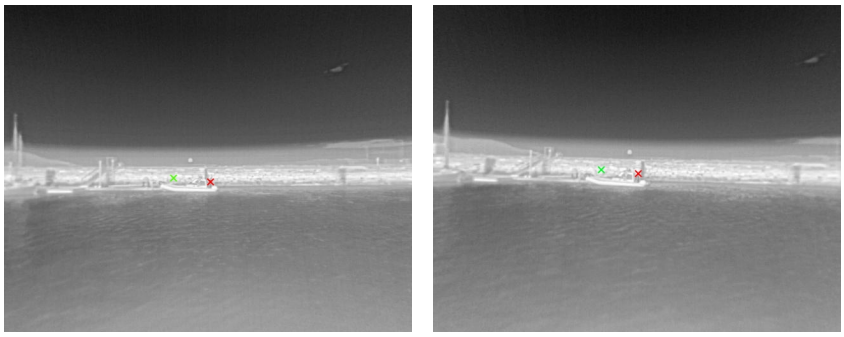
Figure 8.8: Front and front right facing cameras capturing an image of Telemetron at the same time.

The difference between the errors in 8.8 may be due to sequential image sampling from the cameras, the limited frame rate of the cameras or a combination. Any of the scenarios may

lead to one of the frames being up to date, while the other frame is outdated with respect to camera pose and target position. As the frames are captured with different cameras, inaccurate calibration also may affect the error. Based on the results from 7.1 however, this alone should not affect the pixel error this much.

8.2.2 Experiment 2: Data synchronization

Due to conservation of angular momentum, the same principle used in gyroscopes, the roll and pitch of each IR camera remained low. This can be seen in Figure 8.9 where the vertical pixel error is low. The horizontal pixel errors between the projected and selected pixels is close to constant, as the yaw rate is close to constant. This error is however not unsignificant. This speaks to a desynchronization between the images and the rest of the data. By backtracking through the measurements of camera pose and target position, this delay may be estimated.



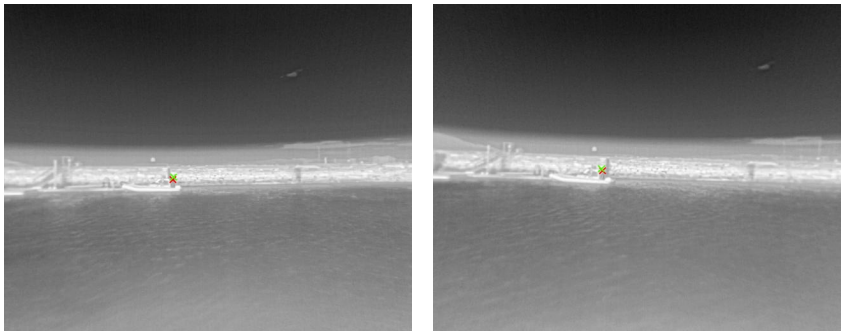
(a) Frame from rear left facing camera during rotation

(b) Frame from front facing camera during rotation

Figure 8.9: The red crosses is where the antenna of Telemetron is positioned in the image frame and the green crosses marks the pixel from the projection

Consider the image frame from the rear left camera in 8.9a. The vertical pixel error between the projected and seen pixel is 6. By calculating the body ray from the camera to both pixels using (6.1), the angle between them can be calculated. The vertical error between the two body rays in degrees is 0.85°. Considering the actual distance between the target and camera, this translates to a vertical error of $0.57\,m$. The horizontal error of 58 pixels, or 8.18° results in a difference of $5.55\,m$ in the camera coordinate system. Rotating with an angular velocity of $0.176\,rad/s = 10.084\,deg/s$, the delay between when the image was taken and when it arrived as data may be estimated. Assuming constant angular velocity, the camera will use $0.81\,s$ rotating 8.18°. Backtracking the gathered data and extracting the image from $0.81\,s$ earlier, the error is reduced to 1 pixel horizontally and 4 pixels vertically, corresponding to an error of 0.14° and 0.42° respectively. An illustration can be seen in Figure 8.10a. In the camera coordinate frame, this gives an horizontal error of $0.1\,m$ and a vertical error of $0.29\,m$.

In the case of 8.9b, the results are similar. Both errors are the same in pixels, and therefore angles. The distance between camera and target is 3.68 m shorter. Therefore, the errors in vertical and horizontal in meters are now 0.85 m and 5.03 m respectively. The angular yaw velocity was in this case $0.128\,rad/s = 7.334\,deg/s$. With this angular velocity, again assumed constant, the camera uses 1.12 s rotating 8.18°. Backtracking the data and extracting the image 1.12 s younger, the horizontal and vertical pixel error is 2 and 3 pixels respectively, translating to 0.28° horizontal and 0.42° vertical body ray error. In the camera coordinate frame, this gives an offset of 0.17 m horizontally and 0.26 m vertically. An illustration may be seen in Figure 8.10b.



(a) Frame from rear left facing camera during rotation corrected for time delay

(b) Frame from front facing camera during rotation corrected for time delay

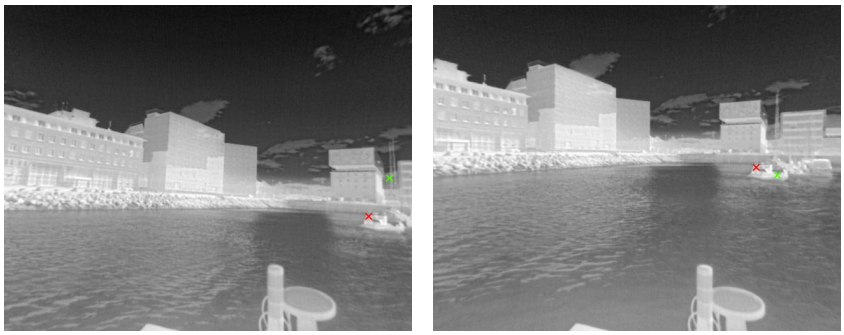
Figure 8.10: The red crosses is where the antenna of Telemetron is positioned in the image frame and the green crosses marks the pixel from the projection.

Some error is still present, this may be due to small inaccuracies in transformations from MilliAmpere to each IR camera or the sample rate of the data, especially the images. The camera runs at 9 FPS, allowing for up to 0.11 s old images. During the experiments, the RTK-GPS was in floating mode, allowing for lower accuracy than stated in section 4.2.

Buffering data and waiting a set amount of time before using it for calculations with an image may reduce the error. It will however not work perfectly due to a variable time-delay between image capturing and the image data being registered. To solve the problem, relevant data should be actively synchronized with the camera. This requires the external triggering feature not yet released by FLIR for the Boson 640 camera. When released and implemented, the only latency between the cameras and NVIDIA TX2 should be close to the specified delay mentioned in section 6.1. As the latency varied with 0.31 s between only two measurements and was up to 1.12 s, there is another source of latency between the cameras and the OBC. This may be gstreamer, the SSH connection between the OBC and sensor rig, or a combination. When the GPS connection to the sensor rig is connected, the image should be synchronized with GPS data and sent as a combined message. When registered at the OBC, node 1 should grab data from a buffer time stamped with gps-time to ensure proper synchronization. Then, all data will be synchronized, and the age of the image can be accurately quantified for use in collision avoidance.

8.2.3 Experiment 3: Fast changing camera pose

In this experiment it is expected to have large and variable errors between seen and projected point due to the non constant rate of roll, pitch and yaw. The results from the reprojection before correcting for latency can be seen in Figure 8.11. From 8.11a, the horizontal and vertical pixel error of 33 and 60 pixels corresponding to 3.18° and 7.07° respectively is fairly high. At the distance between target and camera, this leads to an error of $3.19\,m$ horizontally and $5.81\,m$ vertically in the camera coordinate system. Depending on the method used for distance estimation, this high of a vertical error may render the calculations infeasible. Comparing the image with 8.11b, it can be seen that the vertical error of 12 pixels corresponding to 1.53° is significantly lower. This is likely due to a coincidence where the image latency closely matches the time it takes for the camera to reach a similar pitch value as an earlier value again.

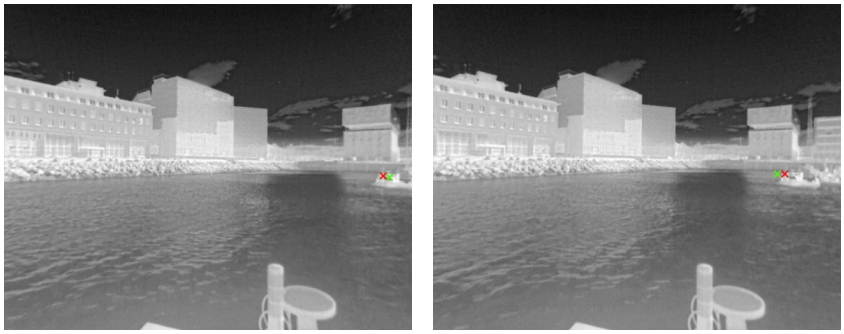


(a) Frame from front right facing camera during fast changes in attitude

(b) Frame from front right facing camera during fast changes in attitude

Figure 8.11: The red crosses is where the antenna of Telemetron is positioned in the image frame and the green crosses marks the pixel from the projection

Assuming constant rate in the 3 rotational degrees of freedom will not be a feasible method for estimating latency due to the variable rate of roll, pitch and yaw. Backtracking the images until the best fit between projected points in Figure 8.11 and the new seen point may give an indication of the image latency as the system seems accurate when corrected for latency based on the results from the previous experiment. The result can be seen in Figure 8.12. The image in 8.12a is $0.9\,s$ younger than 8.11a. Pixel error is 7 and 2 pixels horizontally and vertically, making the errors in angles 0.97° and 0.19°. In camera coordinates, this is an error of $0.95\,m$ and $0.23\,m$ respectively. The image in 8.12b is $1.75\,s$ younger than 8.11b. The horizontal pixel error is 12 pixels, corresponding to a body ray error of 1.28° and $1.08\,m$ error in the camera coordinate frame. In this case, there were no error in the vertical direction. Due to the pose of the target vessel in respect to the camera, there marked pixel inhibits uncertainty as it is not directly seen in the images.



(a) Frame from front right facing camera during fast changes in attitude corrected for latency

(b) Frame from front right facing camera during fast changes in attitude corrected for latency

Figure 8.12: The red crosses is where the antenna of Telemetron is positioned in the image frame and the green crosses marks the pixel from the projection

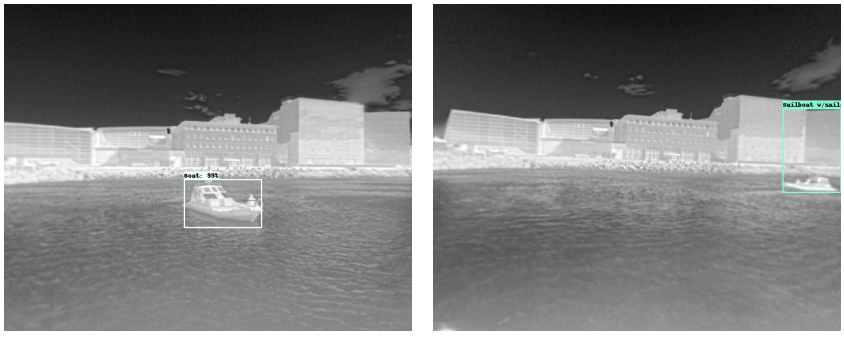
Finding the best fit manually by backtracking in the image data is only a valid method under the assumption that the image latency is the only desynchronized data and all other parameters affecting the projection are correct. From the results when the system was corrected for image latency in 8.2.2, roll and pitch were close to zero at all times, and have not been thoroughly tested prior to this experiment. Therefore, this method may be invalid. An image latency of up to $1.75\,s$ is substantial. With estimated image latency of $0.81\,s$ and $1.12\,s$ in the previous experiment, this may indicate that manually backtracking until the best fit is found is an invalid method. However, the variable image latency may be due to a non optimal implementation of sequential image publishing, which has since been resolved. Occasionally, a set of frames would not be published when sampled before another frame was registered. As $1.75\,s$ is about double the latency estimated from the other examples, this seems to be what happened. If the images are synchronized with the rest of the data, the errors are acceptable. However, with the setup described in 8.2.2, the system output may still be outdated. A solution may be implementing a separate ROS node locally on the NVIDIA TX2 rather than using gstreamer possibly resulting in less latency.

Due to the fast changing attitude, not only image latency, but restricted frame rate may affect the results. For example, before a new frame is sampled in 8.11a, MilliAmperes roll changes by $\Delta\phi_{MA} = 1.48^\circ$. Using equation 4.1, this alone translates to a change in front right camera attitude of $\Delta\phi_{FR} = 0.46^\circ$ and $\Delta\theta_{FR} = 1.41^\circ$. This is another example speaking to the importance of accurate data synchronization when rapid changes in camera pose are present.

8.3 Detection

The dataset used for training may not result in optimal performance due to sparse diversity between classes and similar images within some classes. For the purpose used in [1], where every trained model were trained and tested on the same data with the exception being data augmentation, the data set served its purpose. With a new data set containing IR data set in maritime environments is to be created shortly after this thesis is due, the existing data set was not expanded or modified. Therefore, metrics as mAP and precision recall analysis will not be performed as it may not accurately reflect expected performance in a live system. Based on the analysis done in [1], which used the same detector, but a model typically slightly more accurate, a mAP of close to 86.04% could be expected. Instead, with the system developed, it has been made easy to implement a new model trained with new data set for better performance. Other models as VGG-16 or another detector may also be implemented with minimal changes if desired.

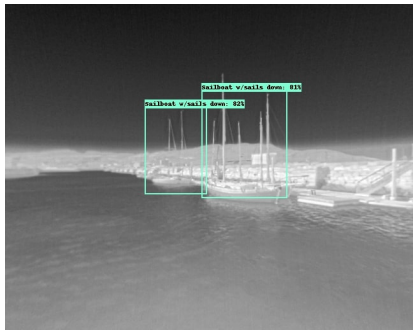
On the data from the experiments, the detector usually detected Telemetron which has characteristics of a typical boat, on distances within about 20 m from the cameras. For reference, Telemetron is 17.8 m away from the camera in 8.13a. On longer distances, the detector usually either misclassified it as "Sailboat w/sails down", or the confidence rate was too low, $< 50\%$, to count as a detection. Examples can be seen in Figure 8.13.



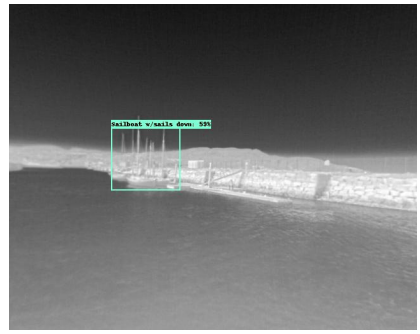
(a) Correct classification of boat. **(b)** Boat classified as "Sailboat w/sails down"

Figure 8.13: Extracted frames of boats from real time object detection.

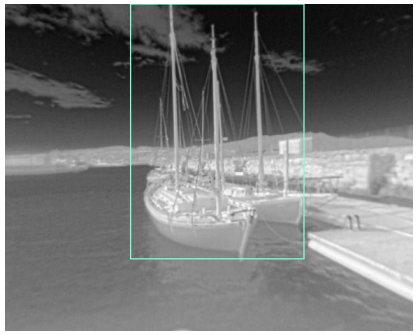
The detector occasionally detects sailboats with sails down. When seen from an angle, or at longer distances, the detection rate is good. When seen directly from the front or back at short distance however, the detector struggled to correctly classify them. These results suggest lack of representation of the class in question at sharp angles and shorter distances. Example detections can be seen in Figure 8.14.



(a) Correct classification of two sailboats with sails down.



(b) Correct classification sailboat with sails down at distance.

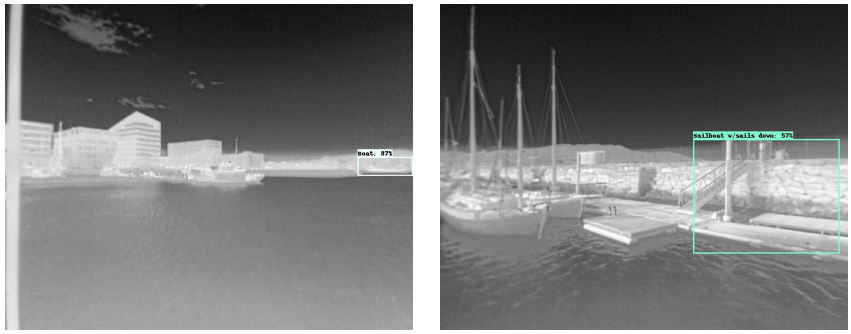


(c) Correct classification of one sailboat with sails down. The bounding box is however not precise.

(d) Confidence rate too low to count as a classification.

Figure 8.14: Extracted frames of sailboats with sails down from real time object detection.

The trained model made some false detections where none of the classes was present. In 8.15b, the detector classifies part of a pier with a pole as a sailboat with sails down. With some similar features, as the pole and a mast, this is again suggests sparse training data of sailboats with sails down.

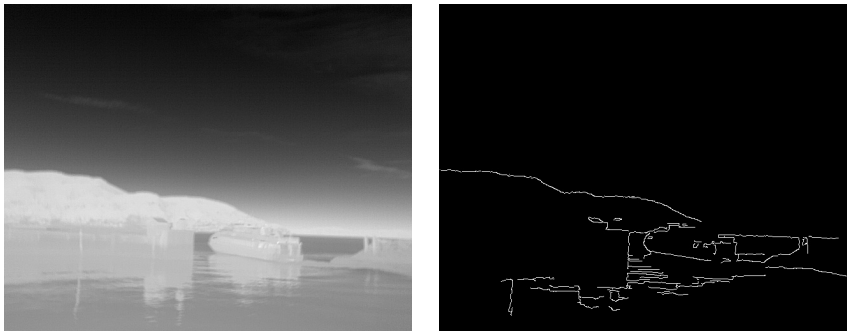


(a) False detection of boat when no boat is present in the bounding box.

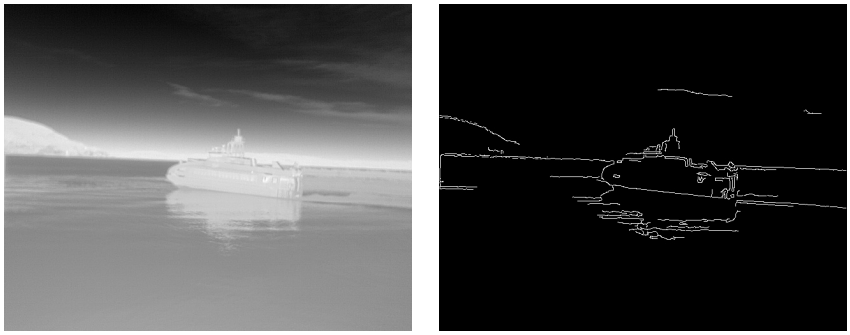
(b) False detection of sailboat with sails down

Figure 8.15: False positives.

Where objects were misclassified, the background usually was cluttered. Background clutter may be a weakness in object detection in infrared data as the target objects usually have a similar temperature to the background, and therefore a similar infrared signature. As there is only one channel, pixel intensity, instead of three channels used in object detection with RGB images, the detector have to classify objects with less information. This results in weak edge-gradients making it challenging to discern the target object from the background. An example can be seen in Figure 8.16, where the edge map has been extracted from two images of the same ferry with different backgrounds. In 8.16b, the silhouette of the ferry is broken due to background clutter. Comparing it to 8.16d, the full silhouette of ferry is intact. For the detector, this may result in false, or no detection.



(a) Ferry with background clutter **(b)** Edge map

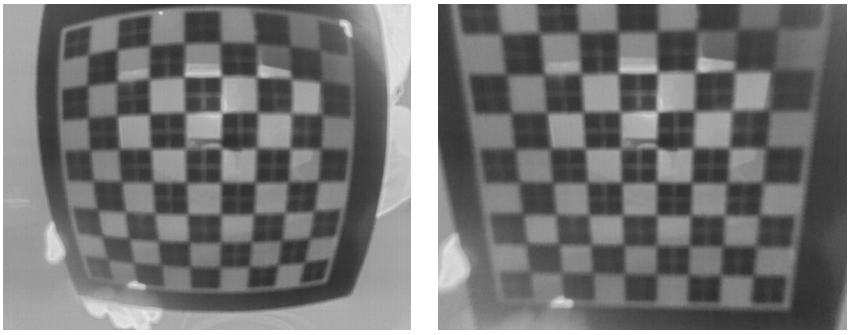


(c) Ferry with less background clutter **(d)** Edge map

Figure 8.16: Edge maps of ferry with and without background clutter. 8.16b is the edge map of 8.16a, and 8.16d is the edge map of 8.16c. The same parameters were used.

8.4 Effective field of view

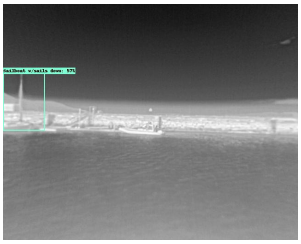
The specified FOV per camera is 95°. Due to calibration however, some FOV is lost per camera. This can be seen in Figure 8.17, where most of the calibration field is present in the image pre-undistortion and the edges suddenly appear out of frame post-undistorion.



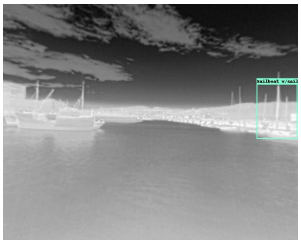
(a) Pre-undistortion **(b)** Post-undistorion

Figure 8.17: A distorted frame and an undistorted frame. The FOV has shrunk due to undistortion.

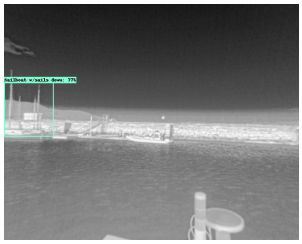
While this still should cover all 360°, the effective FOV is less due to the detector operating better when not just part of an object is seen. 8.18 shows an example of effective FOV of the current model. Consider the frame from the front facing camera in 8.18a. In this case, the tip of the detected sailboat has an x value of 63 px in the frame. In the frame from the same camera in 8.18b, the tip of the sailboat has an x value of about 563 px. In this case, there are 500 "useful" pixels with respect to width, rather than 640. This translates to 63.56° effective FOV for the front facing camera rather than 76.83° edge-to-edge post calibration FOV or 95° specified FOV. Fortunately, the same sailboat is detected in the frame in 8.18c which is captured from the front right facing camera at the same time as the frame in 8.18b. The whole sailboat still cannot be seen in two frames at the same time, which may result in effective blind spots where the detector cannot detect objects.



(a) First frame the detector managed to classify the sailboat while rotating. Captured with front facing camera.



(b) Last frame the detector managed to classify the sailboat while rotating. Captured with front facing camera.



(c) Frame from front right facing camera at the same point in time as 8.18b

Figure 8.18: Images illustrating the effective FOV of IR layer of the sensor rig.

Objects at shorter distances, which are arguably the most critical objects to detect, the effective blind spots may be larger. Consider Figure 8.19, where at distance, there are no blind spots, but moving the object close renders it invisible for the cameras.

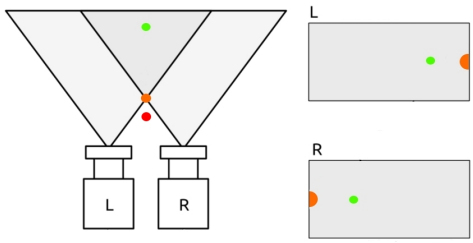


Figure 8.19: The green object is fully seen by both cameras, the orange object is partly seen by both cameras and the red object is not seen by any of the cameras.

Adding together the post calibration FOV of each camera, the full FOV of the IR layer of the sensor rig is 382.22° with overlapping FOV for each neighbouring camera. Because the cameras are configured in a circular orientation, the FOV overlaps at a longer distance away from the cameras compared to Figure 8.19, and the overlap increases more gradually with distance. Therefore, frame stitching might be a viable option to increase detection performance when objects, especially longer objects, are "in between" cameras. Again considering 8.18b and 8.18c, the full sailboat cannot be seen in any of the frames. As there are shared points between the frames however, image stitching could produce a representation of the full sailboat at the current distance.

Chapter 9

Conclusion

Two methods for geometric calibration of LWIR sensitive cameras have been proposed. Radial distortion was still present after calibration when utilizing the active calibration field due to poor circle detection performance. The method utilizing the passive calibration field compatible with Matlab's camera calibration app provided accurate calibration, both in terms of distortion coefficients for undistorting images and intrinsic parameters for undistortion and projection. From the experiment testing calibration precision, the average error in pixel length between ground truth and seen target is 1.91 px.

A fully functional real time multi class object detection pipeline is proposed. The pipeline is created using ROS and is tailored to MilliAmpere. Normalized rays of camera-frame coordinates pointing to detected objects are published on the network for use in collision avoidance. While some data desynchronization is present due to image latency, a method is proposed to resolve data desynchronization utilizing the GPS-connection to the sensor rig.

Several datasets in the form of rosbags have been created. All topics published from MilliAmpere together with LWIR data from the sensor rig and time stamped Telemetron position are recorded for future research and development.

9.1 Further work

A working system for real time object detection with a camera model translating pixels to rays of camera-frame coordinates using a new calibration method for IR cameras have been designed, but there is still room for further research and improvement.

9.1.1 Calibration of infrared cameras

As a cheaper and easier available alternative, aluminium should replace nickel-plated copper to test if it will yield sufficient gradients despite slightly higher emissivity. Because the checkered pattern consists of individual pieces, small gaps between the corners of the laser cut acrylic plate allowed spray paint to slightly blur the corners. To overcome this, single pieces of low and high emissivity materials should be milled and puzzled together in a checkered pattern in an immersed milled square, e.g., highly polished and spray-painted aluminum. This may yield a more accurate calibration field allowing more accurate calibration of the camera. The camera calibration routine should also be tested with 3 radial distortion coefficients.

9.1.2 Data synchronization

The current system suffers from some data desynchronization due to image latency. The IR images from the sensor rig should be time stamped with GPS-time locally before being sent to the OBC. The relevant data should be buffered and stamped with GPS time as well to allow synchronized data for accurate rays to the detected objects. Additionally, time stamping the data will also quantify the age of the data, which may be useful information in collision avoidance.

9.1.3 Object detection

While a model has been trained, the performance is likely not optimal due the existing dataset lacking representation of some classes, while other classes lack variety. A new model should be trained and implemented on a substantially larger dataset. Plans exist in the Autoferry project to gather such a data set over the summer 2019. As objects may not be fully seen by any individual camera when they are "in between" cameras depending on the distance and size of the object, image stitching may increase detection performance in such scenarios.

Appendix A

The distortion coefficients, intrinsic parameters and reprojection errors from both calibration methods are presented in appendix A.

Front IR camera

-	-	Passive calibration field	Active calibration field
Rad	$\begin{bmatrix} k_1 & k_2 \end{bmatrix}$	$\begin{bmatrix} -0.3527 & 0.1081 \end{bmatrix}$	$\begin{bmatrix} -0.3680 & 0.1148 \end{bmatrix}$
Tan	$\begin{bmatrix} p_1 & p_2 \end{bmatrix}$	$\begin{bmatrix} 7.5873e-04 & -9.9092e-04 \end{bmatrix}$	$\begin{bmatrix} 1.3074e-03 & 6.9619e-04 \end{bmatrix}$
K	$\begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 403.5068 & 0 & 320.6654 \\ 0 & 403.5167 & 248.2110 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 404.7162 & 0 & 317.6144 \\ 0 & 404.6830 & 246.7288 \\ 0 & 0 & 1 \end{bmatrix}$
err	-	0.2620	0.0310

Front Right IR camera

-	-	Passive calibration field	Active calibration field
Rad	$\begin{bmatrix} k_1 & k_2 \end{bmatrix}$	$\begin{bmatrix} -0.3633 & 0.1160 \end{bmatrix}$	$\begin{bmatrix} -0.3651 & 0.1208 \end{bmatrix}$
Tan	$\begin{bmatrix} p_1 & p_2 \end{bmatrix}$	$\begin{bmatrix} -6.5629e-04 & -8.3030e-04 \end{bmatrix}$	$\begin{bmatrix} 4.1724e-04 & -6.7855e-04 \end{bmatrix}$
K	$\begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 406.4848 & 0 & 315.7331 \\ 0 & 406.3131 & 249.5483 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 401.2662 & 0 & 318.3653 \\ 0 & 400.2694 & 249.6304 \\ 0 & 0 & 1 \end{bmatrix}$
err	-	0.2081	0.0348

Rear Right IR camera

-	-	Passive calibration field	Active calibration field
Rad	$\begin{bmatrix} k_1 & k_2 \end{bmatrix}$	$\begin{bmatrix} -0.3582 & 0.1159 \end{bmatrix}$	$\begin{bmatrix} -0.3805 & 0.1509 \end{bmatrix}$
Tan	$\begin{bmatrix} p_1 & p_2 \end{bmatrix}$	$\begin{bmatrix} 0.0013 & 2.2999e-04 \end{bmatrix}$	$\begin{bmatrix} -3.0338e-03 & 7.7260e-05 \end{bmatrix}$
K	$\begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 403.9308 & 0 & 319.8940 \\ 0 & 404.0541 & 251.5129 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 400.0112 & 0 & 317.7474 \\ 0 & 399.8288 & 260.9770 \\ 0 & 0 & 1 \end{bmatrix}$
err	-	0.2467	0.0330

Rear Left IR camera

-	-	Passive calibration field	Active calibration field
Rad	$\begin{bmatrix} k_1 & k_2 \end{bmatrix}$	$\begin{bmatrix} -0.3773 & 0.1527 \end{bmatrix}$	$\begin{bmatrix} -0.4052 & 0.2192 \end{bmatrix}$
Tan	$\begin{bmatrix} p_1 & p_2 \end{bmatrix}$	$\begin{bmatrix} 4.4258e-04 & 0.0026 \end{bmatrix}$	$\begin{bmatrix} 8.1041e-04 & 2.0027e-03 \end{bmatrix}$
K	$\begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 404.5471 & 0 & 309.1175 \\ 0 & 404.9005 & 259.6884 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 403.7564 & 0 & 312.4088 \\ 0 & 404.1155 & 255.5701 \\ 0 & 0 & 1 \end{bmatrix}$
err	-	0.2446	0.0272

Front Left IR camera

-	-	Passive calibration field	Active calibration field
Rad	$\begin{bmatrix} k_1 & k_2 \end{bmatrix}$	$\begin{bmatrix} -0.3601 & 0.1206 \end{bmatrix}$	$\begin{bmatrix} -0.3872 & 0.1553 \end{bmatrix}$
Tan	$\begin{bmatrix} p_1 & p_2 \end{bmatrix}$	$\begin{bmatrix} 2.1968e-04 & 0.0015 \end{bmatrix}$	$\begin{bmatrix} 5.8773e-04 & 2.8712e-04 \end{bmatrix}$
K	$\begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 404.5495 & 0 & 320.3003 \\ 0 & 404.5017 & 255.8629 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 402.2561 & 0 & 319.1244 \\ 0 & 402.3386 & 250.9100 \\ 0 & 0 & 1 \end{bmatrix}$
err	-	0.2401	0.0384

Appendix B

MilliAmpere

To launch the ROS network onboard MilliAmpere with the proposed object detection pipeline using LWIR sensitive cameras, follow the steps below:

1. Establish a 5 SSH connections to the sensor rig with $ssh ubuntu@192.168.0.150
2. Start camera stream for each camera using one of the following commands in each terminal:
   - $gst-launch-1.0 v4l2src device=/dev/video-FLIR-F ! videoconvert ! omxh264enc insert-sps-pps=true ! rtph264pay ! udpsink host=192.168.0.255 port=20 sync=false
   - $gst-launch-1.0 v4l2src device=/dev/video-FLIR-FR ! videoconvert ! omxh264enc insert-sps-pps=true ! rtph264pay ! udpsink host=192.168.0.255 port=21 sync=false
   - $gst-launch-1.0 v4l2src device=/dev/video-FLIR-RR ! videoconvert ! omxh264enc insert-sps-pps=true ! rtph264pay ! udpsink host=192.168.0.255 port=22 sync=false
   - $gst-launch-1.0 v4l2src device=/dev/video-FLIR-RL ! videoconvert ! omxh264enc insert-sps-pps=true ! rtph264pay ! udpsink host=192.168.0.255 port=23 sync=false
   - $gst-launch-1.0 v4l2src device=/dev/video-FLIR-FL ! videoconvert ! omxh264enc insert-sps-pps=true ! rtph264pay ! udpsink host=192.168.0.255 port=24 sync=false
3. Open a terminal in the root environment with $sudo su
4. If displaying images with bounding boxes is desired, uncomment the lines with .imshow(...) in detect_ros.py
5. In the ROS workspace, launch the ROS network using $roslaunch milli.launch

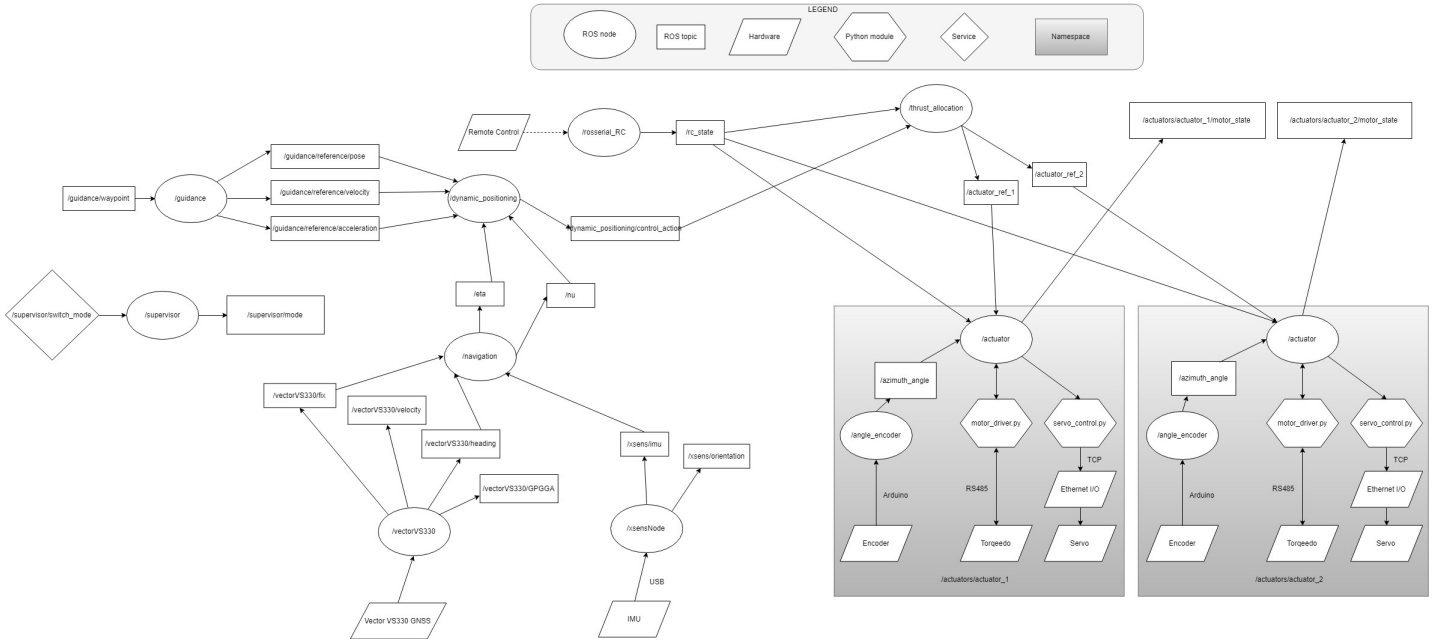


Figure B.1: Existing ROS network on MilliAmpere, note that this is subject to change.

Bibliography

[1] Eskil Hatling Hølland. Object detection of maritime vessels in lwir using convolutional neural networks. Technical report, NTNU, 2018.

[2] Emil Melander, Jesper Haglund, Matthias Weiszflog, and Staffan Andersson. To see the invisible: open-ended university thermodynamics labs with infrared cameras. Technical report, Uppsala universitet, November 2015.

[3] Mica R. Endsley. *Designing for Situation Awareness An Approach to User-Centered Design, Second Edition*. Boca Raton: CRC Press, 2004.

[4] Øystein Kaarstad Helgesen. Sensor fusion for detection and tracking of maritime vessels. Master's thesis, NTNU, January 2019.

[5] NTNU. Autoferry. `https://www.ntnu.edu/autoferry/about`.

[6] F. S. Leira, K. Trnka, T. I. Fossen, and T. A. Johansen. A ligth-weight thermal camera payload with georeferencing capabilities for small fixed-wing uavs. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 485–494, June 2015.

[7] W. Hartmann, S. Tilch, H. Eisenbeiss, and K. Schindler. Determination of the uav position by automatic processing of thermal images. 2012.

[8] Susana Lagüela, Higinio Gonzalez, Julia Armesto, and Pedro Arias. Calibration and verification of thermographic cameras for geometric measurements. *Infrared Physics Technology - INFRARED PHYS TECHNOL*, 54:92–99, March 2011.

[9] Gordon Petrie and G Buyuksalih. Geometric and radiometric calibration of frame-type infra-red imagers. *Publications Institute of Photogrammetry Engineering Surveying, University of Hannover*, 18, September 1999.

[10] Thomas Luhmann, Julia Ohm, Johannes Piechel, and Thorsten Roelfs. Geometric calibration of thermographic cameras. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38:411–416, 2010.

[11] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek. Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):1993–2016, Aug 2017.

[12] Young-Sik Shin and Ayoung Kim. Sparse depth enhanced direct thermal-infrared SLAM beyond the visible spectrum. *CoRR*, abs/1902.10892, 2019.

[13] Amanda Berg. *Detection and Tracking in Thermal Infrared Imagery*. PhD thesis, Linköping University, 2016.

[14] Emissivity in the infrared. https://www.optotherm.com/emiss-physics.htm.

[15] B. Oswald-Tranta. Motion deblurring of infrared images. pages 783 – 787, 2019.

[16] Kenji Hata and Silvio Savarese. Cs231a course notes 1: Camera models. https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf.

[17] Illinois college of engineering. Geometry and camera models. https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2002%20-%20Projective%20Geometry%20and%20Camera%20Models%20-%20Vision_Spring2011.pdf, 2011.

[18] M. Forsyth and J. Ponce. *Computer Vision, A Modern Approach*. Pearson, 2012.

[19] Visesh Chari and Ashok Veeraraghavan. *Lens Distortion, Radial Distortion*, pages 443–445. Springer US, Boston, MA, 2014.

[20] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, June 1997.

[21] https://se.mathworks.com/help/vision/ug/single-camera-calibrator-app.html.

[22] Marco Leonardi. Ttk25 - lecture 8 - supervised learning, 2018.

[23] Convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/. (Accessed on 24/10/2018).

[24] Michael A. Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com, 2015. (Accessed on 12/10/2018).

[25] Pooling layer. https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/pooling_layer.html.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[27] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[28] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[29] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[30] Eddie Forson. Understanding ssd multibox real time object detection in deep learning. https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab.

[31] Wei et al. Liu. SSD: Single shot multibox detector, 2016.

[32] Dumitru Erhan Dragomir Anguelov Sergey Ioffe Christian Szegedy, Scott Reed. Scalable, high-quality object detection. https://arxiv.org/abs/1412.1441, December 2014. (Accessed on 03/10/2018).

[33] Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Andrew Rabinovich Christian Szegedy, Wei Liu. Going deeper with convolutions. https://arxiv.org/abs/1409.4842, September 2014. (Accessed on 03/10/2018).

[34] Overfitting in machine learning: What it is and how to prevent it. https://elitedatascience.com/overfitting-in-machine-learning.

[35] Aurélien Géron. Hands-on machine learning with scikit-learn and tensorflow. https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282, March 2017. (Accessed on 12/10/2018).

[36] NI. Thresholding. http://zone.ni.com/reference/en-XX/help/372916T-01/nivisionconcepts/thresholding/.

[37] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[38] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[39] Canny edge detection. https://docs.opencv.org/master/da/d22/tutorial_py_canny.html.

[40] Maritime Robotics. Technical documentation. 2019.

[41] https://github.com/FLIR/BosonLinuxSDK.

[42] Olav Gjestad. Schematics ntnu360. 2019.

[43] Hemisphere. Vector vs330 gnss receiver data sheet. August 2017.

[44] Xsens. Mti 10-series. 2017.

[45] B. Gerkey M. Quigley and W. D. Smart. *Programming Robots with ROS*. O'Reilly, 2015.

[46] H. Helgesen et. al. Real-time georeferencing of thermal images using small fixed-wing uavs in maritime environments. Technical report, NTNU, January 2019.

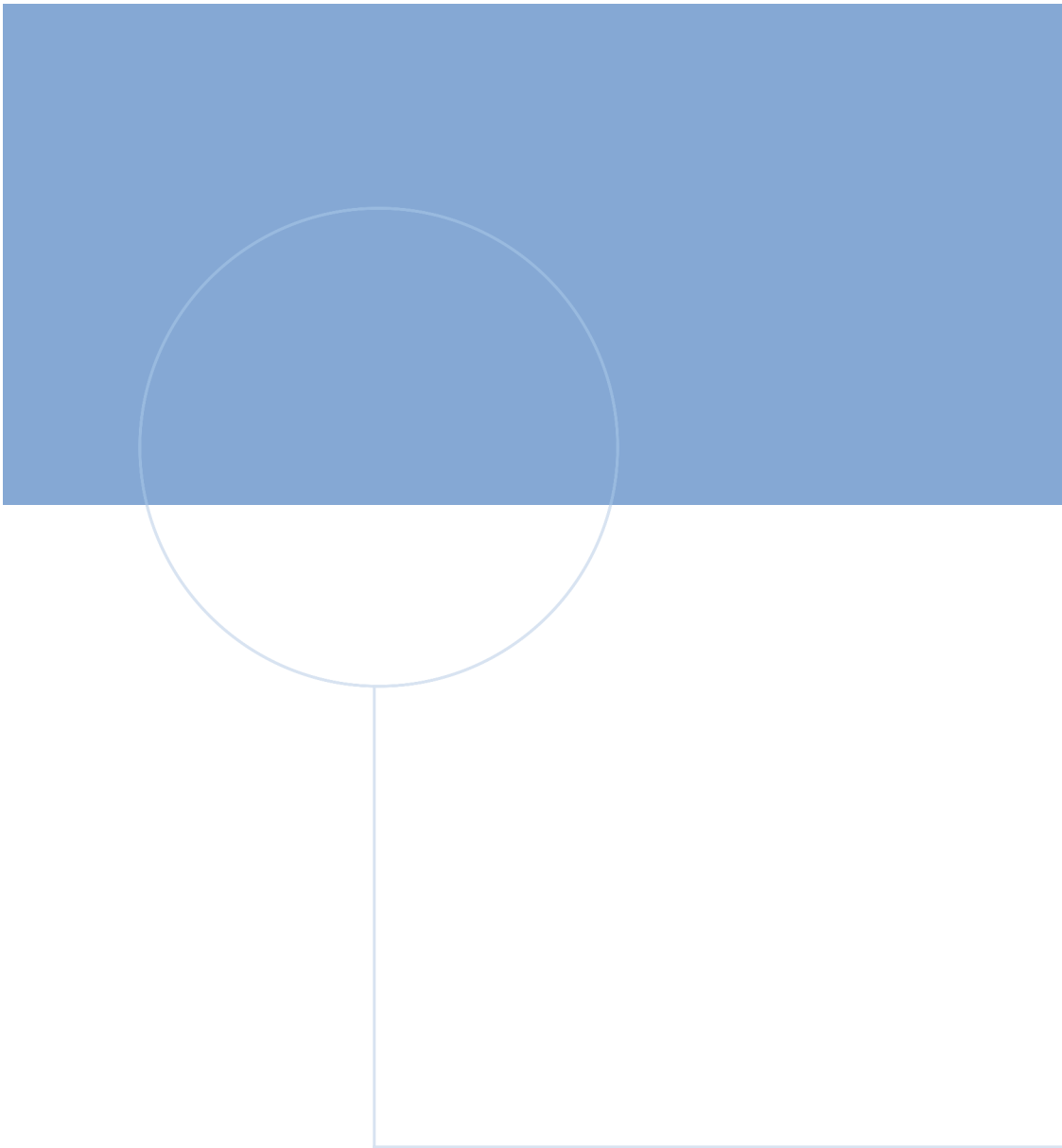
[47] Engineering Toolbox. Emissivity coefficients materials. https://www.engineeringtoolbox.com/emissivity-coefficients-d_447.html, 2003.

[48] G. Bradski. Dr. dobb's journal of software tools. 2000.

[49] https://gstreamer.freedesktop.org/documentation/.

[50] FLIR. Flir boson engineering datasheet <9hz configuration. August 2018.

[51] Rohit Salem. Tensorflow object detector. https://github.com/osrf/tensorflow_object_detector?fbclid=IwAR05YBhxkQ_1uXxVWjW_O0MknqD7xo7xrEAHwklbAXibAE5j_zfXhLDDIi8.



NTNU
Norwegian University of
Science and Technology