Bachelor thesis

in the degree program

Maritime Technologies (MAR)

Integration of High-resolution Embedded Camera for Maritime Object Georeferencing

Submitted by

Yanneck Deichmann

Matr. Nr.: 32942

on April 29, 2024

at the HS Bremerhaven

University supervisor: Prof. Dr. Axel Bochert

Company: German Aerospace Center Institute for the Protection of Maritime Infrastructures

Technical supervisor: Borja Jesus Carrillo Perez

Affidavit

I, Yanneck Deichmann, declare that I have authored this thesis independently, that I have not used other than the declared sources, and that I have explicitly marked all material which has been quoted either literally or by content from the sources used.

Bremerhaven, the April 29, 2024

Yanneck Deichmann

Digital signiert von Yanneck Deichmann
DN: OU=MI-SIT, O=DLR, CN=Yanneck Deichmann, E=yanneck.deichmann@dlr.de
Ort:
Datum: 2024.04.29 15:39:25+02'00'
Foxit PDF Reader Version: 2024.1.0

Yanneck Deichmann

Acknowledement

Dear readers, thank you to all who have contributed to the realisation of this bachelor thesis on developing a photosensitive embedded system that enables image transformation with the help of sensor data from a smartphone. Special thanks go to Prof. Dr. Axel Bochert, my supervisor at the Hochschule Bremerhaven. His expert knowledge, constructive feedback, and support have been instrumental in enabling me to complete this thesis.

Also, a special thanks goes to Borja Jesus Carrillo Perez, my supervisor at the German Aerospace Institute, for the protection of Maritime infrastructure. His practical experience and his dedication have been an indescribable help to me in writing this thesis.

A special thanks go to the DLR Institute for the Protection of Maritime Infrastructures because they made it possible for me to do my bachelor's thesis with their company.

My thanks go to everyone who supported me in these difficult moments.

Last, I want to thank my parents, my beloved partner, and the DLR staff. Your moral support, encouragement and encouraging words constantly strengthen me when things get complicated.

Thank you for everything.

Abstract

In maritime security contexts, it is essential to have an accurate and timely georeferencing system for maritime objects such as ships and vessels. This is crucial for enhancing situational awareness and ensuring efficient monitoring. Existing systems are often static, and when they are moved, they lose the reference for image transformation. To address this issue, this thesis proposes the integration of a high-resolution camera into a mobile housing unit with an embedded system for image processing and a smartphone for complementary data acquisition, which allows ship georeferencing in a mobile manner.

The ship recognition software used in this system is YOLOv8, and the Android smartphone collects sensor data. This configuration facilitates accurate conversion of pixel coordinates of the recognized ships into geographic locations using raycasting as georeferencing method, which enables display of latitudes and longitudes of ships on a map. The experimental results validate that the detection and georeferencing of ships is performed with an error of 16±7 meters using the integrated system. These results are comparable to those of static camera systems, however adding the potential of mobile embedded systems to further enhance maritime monitoring capabilities. This innovative solution is a step forward in improving the accuracy and efficiency of maritime monitoring systems

Table of Contents

Acronyms

R	Rotation Matrix.
T	Translation Vector.
CMOS	Complementary Metal Oxide Semiconductor.
CNNs	Convolutional Neural Networks.
CPU	Central Processing Unit.
DLR	German Aerospace Center.
FoV	Field of View.
FPGAs	Field Programmable Gate Arrays.
GB	Gigabyte.
GPS	Global Positioning System.
GPUs	Graphics Processing Units.
hAcc	Horizontal Accuracy.
lat	Latitude.
lon	Longitude.
MI	Institute for the Protection of Maritime Infrastructures.

PCIe	Peripheral Component Interconnect Express.
USB	Universal Serial Bus.
WGS84	World Geodetic System 1984.
YOLOv8	You Only Look Once, Version 8.

1. Introduction

This thesis was developed in collaboration with the Institute for the Protection of Maritime Infrastructures (MI) of the German Aerospace Center (DLR) in cooperation with the Methods and Processing group. Focusing on the processing and analysis of sensor and instrument data, this group explores maritime regions to detect and assess critical infrastructures both above and below the waterline [15]. The foundation of this thesis lies in the recognition of the maritime situation for the improvement of situational awareness. Throughout this thesis, the resources, materials, and premises of the institution have been utilized.

1.1. Motivation

In the context of this Bachelor's thesis, the importance of ship detection and georeferencing in improving maritime situational awareness is highlighted. Maritime Situational Awareness involves understanding and monitoring maritime activities to mitigate security risks, protect the environment, and optimize navigation [24]. By accurately detecting vessels from optical cameras and precisely referencing their geographical positions, the risk of collisions can be reduced, the utilization of maritime resources can be optimized, and the maritime industry's strict safety and security protocols can be maintained [25].

The choice of topic for this bachelor thesis stems from the interest in developing maritime systems with security relevance. the integration of an embedded system and a high-resolution camera that aimed to detect ships and convert the pixel of the detected ships into real-world coordinates. In previous works [5], ship recognition and georeferencing were accomplished using image transformation. However, when the camera was moved, the calibration was no longer valid, therefore losing the geographic references. Moreover, the work in [5] uses low-resolution cameras. The use of a high-resolution camera and an embedded system that integrates sensor data to dynamically georeference ships would address the gap presented. The high-resolution of the high-resolution camera enhances the system's capability for object recognition and georeferencing, and using an embedded system allows the processing of data locally, enhancing speed and security [6].

Another innovative aspect is the utilization of data from sensors on an Android smartphone for transformation and calibration. Furthermore, the use of external sensors to calibrate the system for georeferencing, allows the system's mobility and sets it apart from the stationary system of [5] and a lightweight and compact size facilitate easy transport. This mobility serves several purposes: it allows for cost-effective deployment of the camera, and it enables temporary monitoring, during the construction of maritime infrastructures, during natural disasters or accidents. Additionally, it significantly enhances maritime security by enabling real-time monitoring, thereby aiding in the prevention of unauthorized or suspicious activities and facilitating prompt responses to potential threats or emergencies.

1.2. Goals of the Thesis

This thesis is centered around the development of a embedded vision system with a high-resolution industrial camera designed to recognize ships in images and georeference them to obtain their Latitude (lat) and Longitude (lon) coordinates relative to the geographic coordinate system. The completion of this thesis contributes to the field of maritime computer vision and provides insights into the realm of maritime security. The main goals of this bachelor thesis are outlined as follows:

- Integration of hardware components into the system, encompassing a high-resolution camera, an embedded device, a smartphone, a lens, a Peripheral Component Interconnect Express (PCIe) and a hard drive.
- Calibration of the system to ensure optimal performance.
- Integration of essential software for ship detection and georeferencing of captured high-resolution images.

- Experimental evaluation of the accuracy of the employed georeferencing method within this thesis, to assess the performance of the entire pipeline.

1.3. Structure of the Thesis

To ensure a thorough and engaging understanding of the objectives put forward in this thesis, it has been thoughtfully divided into five distinct and interconnected parts. Each part has been structured to provide a detailed analysis of the research objectives, building on the previous section to create a cohesive and compelling narrative. This division of the thesis into distinct parts allows for a clear and logical presentation of the research findings, making it easier for the reader to follow and comprehend.

Chapter 2 delves into the theoretical foundation, discussing embedded vision for maritime awareness, background on object detection with a focus on the maritime domain, and image georeferencing for maritime applications.

Chapter 3 focuses on the description of the system, beginning with a summary of the system pipeline using a diagram and elaborating on each pipeline component. This includes the 50-megapixel high-resolution camera, its housing, the lens, the Jetson AGX Xavier as embedded system, and a PCIe to connect the camera with the system. it will also be discussed how metadata required for the system are acquired by reading sensors from a smartphone, comprising longitude, latitude, and orientation angles crucial for image georeferencing.

A machine-learning object detector, specifically Ultralytics' You Only Look Once, Version 8 (YOLOv8), is employed to detect ships in the images. The chapter concludes with the utilization of the raycasting method for ship georeferencing.

Chapter 4 focuses on system calibration, involving the calculation, determination, and calibration of individual parameters of the smartphone sensors. Ensuring the smartphone's optimal positioning for accurate data determination and signal strength is paramount.

Chapter 5, discusses the implementation, execution of experiments, test setup, and presentation of results.

Finally, the conclusion chapter evaluates the measurements and provides recommendations for future work on the project.

2. Theory

This chapter aims to present the fundamental concepts crucial for comprehending this thesis. Firstly, it offers an overview of embedded vision's application in the maritime domain. Second, it presents the process of object recognition and subsequent georeferencing from images. Additionally, it presents mathematical principles necessary for converting pixel coordinates into world coordinates.

2.1. Embedded Vision for Maritime Awareness

Given the complexity of the maritime environment, which includes various ships, boats, and other maritime operations, maritime awareness is essential for minimizing collisions, illegal activities, environmental impacts, and threats to maritime security [24]. A comprehensive understanding of the marine environment is necessary to implement preventive measures. Hence, embedded vision has been utilized in maritime surveillance, revolutionizing the implementation of advanced image processing in embedded systems [24]. Embedded vision systems facilitate real-time processing, automatic pattern recognition, and object tracking. The efficiency of embedded vision in maritime surveillance has been bolstered through specialized hardware components such as powerful Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs). This hardware enables parallel processing of large volumes of visual data and helps optimize the performance of surveillance systems [23]. Integrating high-resolution cameras and sensors forms the cornerstone of embedded vision in maritime surveillance. These modern imaging devices facilitate real-time capture of detailed visual information about ships, boats, and other maritime objects [6]. The application of specialized image processing algorithms is critical for extracting relevant information from the captured visual data. Algorithms for object detection, pattern recognition, and tracking contribute to secure vessel identification and precise monitoring of their movements [9]. Another rationale for using embedded systems in computer vision setups is data security. Transferring data to a server risks cyberattacks, such as privacy breaches [34]. Moreover, data privacy laws can be better respected when the system processes confidential data such as people present on the scene, without the risk of data transfer to a server or cloud [17]. This thesis utilizes embedded systems to perform ship detection and georeferencing from high-resolution images.

2.2. Object Recognition in the Maritime Domain

Object recognition in the maritime domain has its foundation in image processing, with potential applications ranging from monitoring ship traffic to detecting possible collisions at sea [30]. This powerful tool enables the recognition and highlighting of ships in images and videos, as illustrated in Figure 2.1 [28]. Combining this with machine learning capabilities allows the system to automatically reference important information. Automatic recognition is essential to provide the system with objects and corresponding object classes for object georeferencing [40]. Artificial neural networks, such as Convolutional Neural Networks (CNNs), are highly effective for image and object recognition as they can directly learn features from raw data [33], [40]. To enable this, characteristic features of the objects to be recognized, including shape, color, or texture, are first extracted by the CNNs. Once these features are extracted, they are compared with features in various object classes. Upon detection of an object, its position is localized in the image, usually in the form of bounding boxes around the detected object. Finally, each recognized object within the image is matched with object classes [41]. In this thesis, the class "boat" is utilized; this is a pre-defined object class, and the YOLOv8 [36] software is responsible for object recognition in the camera system used in this thesis. Details and special features of YOLOv8 are described in more detail in Chapter 3.5.

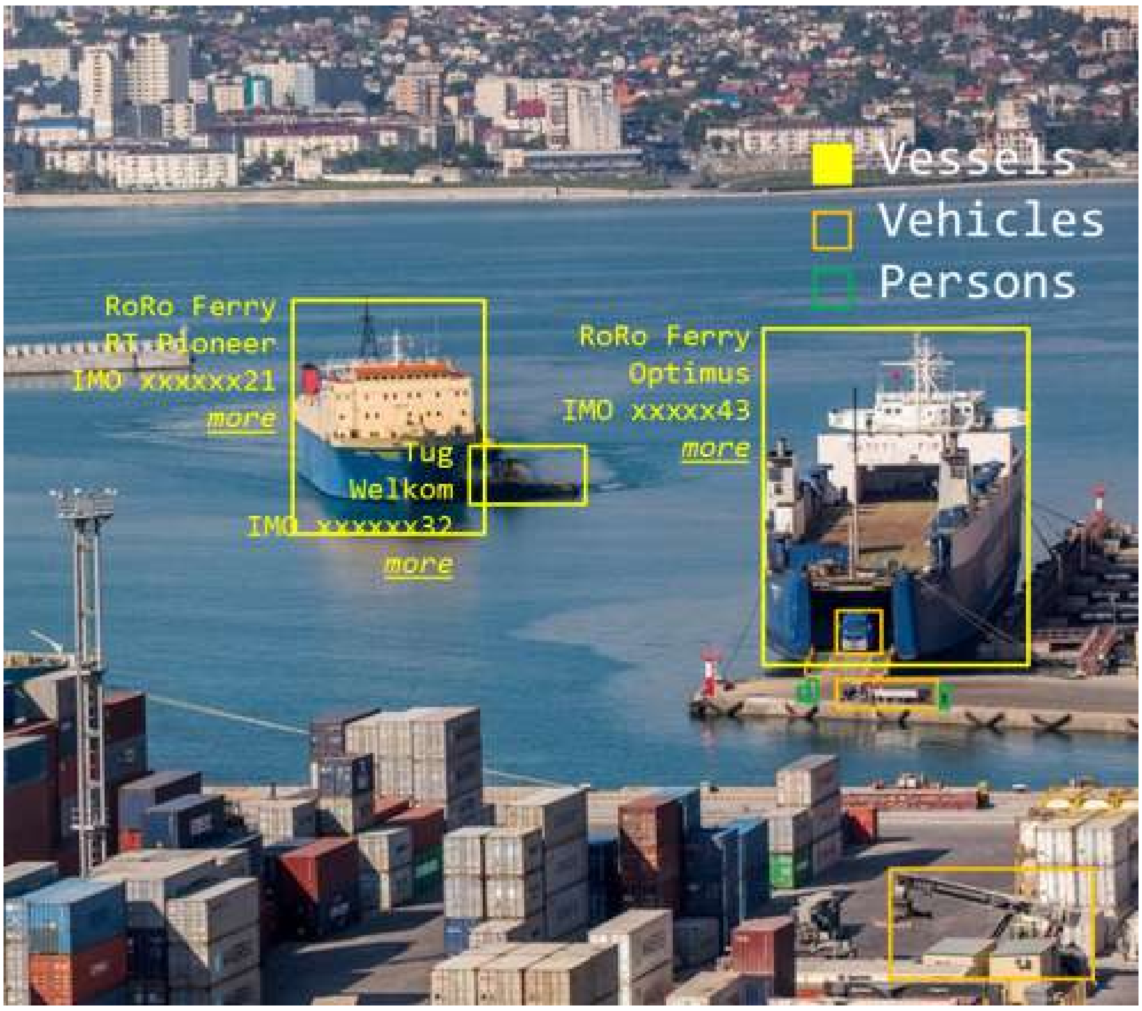


Figure 2.1.: Mock-up example of how ships and other relevant elements at the maritime infrastructure are detected and classified [15].

2.3. Image Georeferencing for Maritime Applications

In this thesis, image georeferencing is employed as one of the tasks, particularly to automatically assign the latitude and longitude of the ships detected in the high-resolution images. Accurate information about the positioning of ships relative to the camera system is indispensable for effective monitoring reference. Optimized monitoring of ships necessitates real-time detection, enabling tracking of relevant variables in maritime situations [5]. Additionally, it facilitates the detection of geographical positions within critical infrastructures. Similar systems have been utilized in previous works at DLR to detect and reference ships [6]. However, the camera systems used in prior research were static, which makes the system unusable in mobile setups. This thesis addresses this issue by developing a system capable of producing accurate ship georeferences even after changes in location or realignment. Therefore, the collection of sensor metadata is crucial for referencing and image transformation in subsequent stages.

2.3.1. Image Transformation

Image transformation, in the realm of computer vision, constitutes a fundamental process for manipulating and adjusting visual data captured by cameras, allowing to be presented in a different space. This transformation procedure is essential for converting images to meet specific criteria or applications. Image transformation involves a series of mathematical operations applied to the raw visual input, resulting in a modified representation that aligns with the desired objectives [37]. This transformation finds application in various contexts, making it a key element in image processing, where it models precise spatial relationships between objects and the camera [35]. Additionally, image transformation plays a crucial role in specific object georeferencing. Adjusting perspectives and orientations enables advanced algorithms to accurately identify objects in a scene and display them on a map [22]. Furthermore, in surveillance systems, camera transformation also plays a central role in accurately monitoring objects, improving situational awareness.

2.3.1.1. Extrinsic And Intrinsic Parameters

Extrinsic and intrinsic parameters are essential terms for image transformation and must first be explained. The extrinsic parameters describe the spatial relationship between the camera and the external world. Moreover they provide information about how the camera was positioned and oriented in three-dimensional space [39]. The intrinsic parameters are the properties of the camera itself and are characteristic values for each camera and lens.

The first component of the extrinsic parameters is represented by Translation Vector (T), denoting the camera's position in the world coordinate system [4]. It comprises three values, which are as follows:

$$T = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix} \tag{2.1}$$

T_x, T_y, and T_z indicate displacement along the X, Y, and Z axes. For example, if T_x = 2 meters, $T_y = 1$ meter, and $T_z = 3$ meters, the camera was 2 meters to the right, 1 meter up, and 3 meters forward from the reference point. The second component of the extrinsic parameters is denoted by Rotation Matrix (R), which describes the camera's orientation. It consists of nine elements representing the rotational transformation around the three axes (pitch ϕ, yaw θ, and roll ψ) as shown in: 2.2.

$$R = \begin{bmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} \\ \theta_{yx} & \theta_{yy} & \theta_{yz} \\ \psi_{zx} & \psi_{zy} & \psi_{zz} \end{bmatrix} \tag{2.2}$$

Each element in the matrix contributes to the camera's rotation in the respective direction. Combining the translation vector and rotation matrix forms the extrinsic matrix (often denoted as $[R \mid T]$), ultimately defining the camera's extrinsic parameters, as shown in 2.3 [42]. Understanding extrinsic parameters and how to use them was crucial for various applications where accuracy between the camera and the environment is essential.

$$[R|T] = \begin{bmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} & T_x \\ \theta_{yx} & \theta_{yy} & \theta_{yz} & T_y \\ \psi_{zx} & \psi_{zy} & \psi_{zz} & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

This matrix shown in (2.3) represents the combined rotation matrix and translation vector. Typically, this is a 3x4 matrix, but in computer vision and graphics, a fourth row with [0 0 0 1] is often added to extend the matrix to homogeneous coordinates [11]. This is done because, in computer graphics and computer vision, homogeneous coordinates are often used to describe transformations in a uniform framework, which is also why this approach enables seamless integration of camera extrusion and intrinsic parameters for image transformation and 3D visualisation [3].

In regard to the intrinsic parameters, a significant parameter is the focal length (f_x,

f_y) which is a crucial parameter representing the distance from the camera's lens to the image sensor. It determines the camera's ability to focus and affects the scale of objects in the image [42]. A longer focal length results in magnification and a narrower Field of View (FoV), while a shorter one provides a broader field. The focal length is typically expressed in pixels. This is determined by the lens used. The second intrinsic parameter is the sensor diagonal size (c_x, c_y) in millimeters, of the camera itself. This indicates the length of the diagonal of the image sensor used in the camera. A larger diagonal means higher image quality, thanks to better light sensitivity. These two parameters are used to calculate the camera's FoV, an essential intrinsic value for subsequent image georeferencing [26].

Both the focal length and the sensor diagonal size are included in the intrinsic matrix K, as can be seen in 2.4. This matrix is a fundamental component of camera calibration and geometry. It describes the intrinsic parameters of a camera that relate to the optical properties of the camera itself, regardless of its position or orientation in the scene.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.4. Mathematical Conversion of Pixel to World Coordinates

Converting pixel coordinates into geographical coordinates requires calibration and knowledge of the intrinsic and extrinsic camera parameters. The focal length is the principal point of the intrinsic parameters, and from the extrinsic parameters, the rotation matrix (R) and translation vector (T) are needed. This individual formulas and their derivations will be explained in more detail in the Methods Chapter 5; in this chapter, the aim is to understand the function of the formulas in question. The first step is to find the perspective division; when viewing a scene through a camera, the three-dimensional scene is projected onto a two-dimensional image plane [22]. This process can be described by perspective division, makes objects farther away appear smaller in the image. In georeferencing, the perspective division is used to bring the projected image coordinates into a standardised form, which can then be converted into 2D image coordinate space, for example, to create camera surveillance or a cartographic representation. This is achieved using the following Matrix:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.5}$$

After the perspective division, the normalized image coordinates (X', Y', Z') are obtained. To convert the image coordinates into normalised camera coordinates, the inverse of the intrinsic matrix K^{-1} was used. The inverse intrinsic matrix was applied to the normalized image coordinates by multiplying

$$\begin{bmatrix} X'' \\ Y'' \\ Z\text{”} \end{bmatrix} = K^{-1} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \tag{2.6}$$

The results (X'', Y'', Z'') are now normalized camera coordinates. In the next step, these normalized camera coordinates are further transformed to obtain the global 3D coordinates of the point. This step is essential to determine the spatial position of the point concerning the camera, which is done with the help of the inverse of the rotation matrix (R^{-1}). The purpose of this inverse matrix is to transform the normalized camera coordinates again, but this time in the global space. The rotation matrix R describes the rotation of the camera in 3D space. This matrix's inverse (R^{-1}) transforms the normalized camera coordinates into the global space:

$$\begin{bmatrix} X''' \\ Y''' \\ Z'\text{”} \end{bmatrix} = R^{-1} \begin{bmatrix} X'' \\ Y'' \\ Z\text{”} \end{bmatrix} \tag{2.7}$$

After obtaining the coordinates (X''', Y''', Z''') in the global space through the inverse of the rotation matrix, the next step involves reverting the translation. The translation vector (T) represents the camera's position in the 3D-space. To obtain the precise global 3D coordinates of the point, the subtraction of the translation vector is performed:

$$\begin{bmatrix} X_{global} \\ Y_{global} \\ Z_{global} \end{bmatrix} = \begin{bmatrix} X''' \\ Y''' \\ Z''' \end{bmatrix} - T \tag{2.8}$$

Here, X_{global}, Y_{global}, and Z_{global} are the final global coordinates of the point in 3D space. This step accounts for the camera's translation from the origin. It ensures that the coordinates are placed in the global coordinate system rather than being relative to the camera position. For example, in latitudes and longitudes, the origin is located at (0°, 0°) where the prime meridian and the equator intersect. This process, known as camera transformation, is crucial for mapping pixel coordinates to real-world spatial coordinates, enabling applications such as object localization in computer vision and photogrammetry [22]. The last step in this transformation involves converting these coordinates into geographic coordinates, precisely latitude (lat) and longitude (lon). Let us denote the final global coordinates again as $Xglobal$, $Yglobal$, and $Zglobal$, and the resulting geographic coordinates as (lat, lon). The conversion look like this:

$$\text{lat} = \arctan\left(\frac{Z_{global}}{\sqrt{X_{global}^2 + Y_{global}^2}}\right) \tag{2.9}$$

$$\text{lon} = \text{arctan2}\left(Y_{global}, X_{global}\right) \tag{2.10}$$

In this case, arctan was the arctangent function, and arctan2 was a modified arctangent function that considers the signs of both its arguments, providing a full range of angles. These formulas give the latitude and longitude of the point in the global space. The resulting coordinates (lat, lon) represent the point's geographic location on the earth's surface. In this thesis, the raycasting (explained in section 3.6) leverages the theoretical concepts presented in this section to bring pixels to real world coordinates.

3. System Description

In an internship prior to this work, the basic structure of the housing in which the camera and the associated components are located was created prior to the start of this thesis. Several essential criteria had to be taken into account when selecting the housing. These criteria were: the lightest possible housing construction, waterproof, resistant, and suitable for outdoor environments, with enough space for the individual components. The housing was developed to be easily integrated with the embedded system and the industrial camera. It should also be possible to machine the housing easily. For these reasons, the choice fell on the aluminium housing GA model 9119.210 from Rittal GmbH & Co. KG [31] see Figure A.3. To organize the planned position structure within the housing as space-savingly as possible, the necessary brackets and supports were designed with Solidworks and produced with a 3D printer. A circular cut-out was also made in the housing to insert a window to enable the camera to capture images. In addition, an opening for the cable feed was cut into the inside of the housing on the opposite side. Also, a holder for a silica package was attached to the inside of the removable lid of the housing together with them. This was to counteract excessive humidity, as this would damage the electronics inside the housing. In this thesis, one of the main tasks was to integrate the described hardware components and software to create a working pipeline for the system.

This chapter examines the intricacies of the pipeline that underpins this thesis, including its structure and individual components. It also explains how the individual components are linked and work together to create a coherent and effective pipeline.

3.1. Diagram that summarises the pipeline

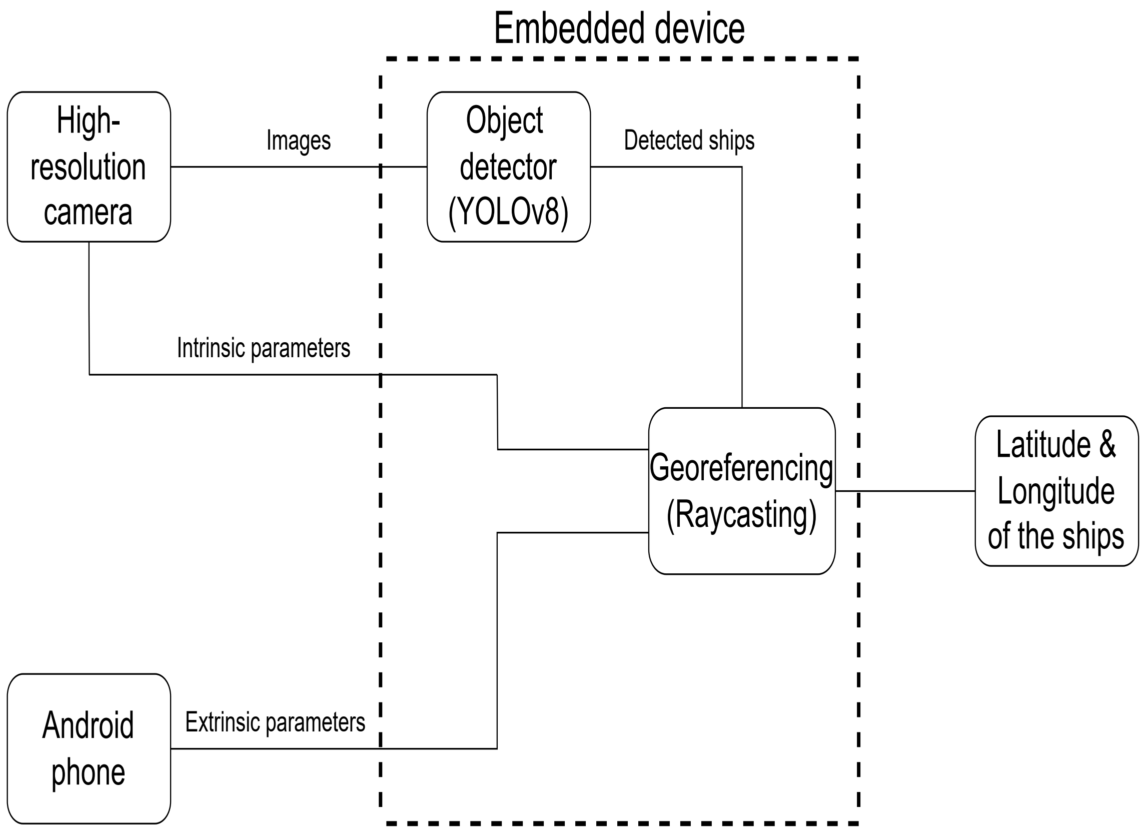


Figure 3.1.: Pipeline of the System

Figure 3.1 shows the diagram of the pipeline that summarizes the integration. The pipeline consists on the use of a high-resolution camera for image capture, along with intrinsic parameters. An android phone is used to obtain the extrinsic parameters. The images and the parameters are used for ship detection and georeferencing, which will provide the latitude and longitude of the detected ships. The individual components will be explained in more detail. First, the camera and the lens are presented. This is followed by a description of the embedded device used. Afterwards, a description of the extrinsic parameters collected with the help of an Android phone is provided. The subsequent module is the object detection algorithm YOLOv8 and how it is employed for ship detection. The explanation concludes with a description of the raycasting principle used for ship georeferencing.

3.2. High Resolution Camera

The first step in the pipeline is the collection of high-resolution images. For this purpose, the 50-megapixel camera CB500CG-CM from Ximea was used in this thesis [38]. This is a state-of-the-art industrial camera characterized by its high resolution and impressive performance, see in the Appendix A.2. This high resolution is achieved by the built-in Complementary Metal Oxide Semiconductor (CMOS) image sensor, which enables the camera to create images with up to 50 megapixels. Previous works used low-resolution cameras for georeferencing [5]. To the best of our knowledge, this work is the first one using this 50Mpix industrial camera for georeferencing purposes in the context of maritime awareness. The CB500CG-CM is a color camera, which means that it can capture and reproduce the various of 50 megapixel at 16 bit resolution, allowing 65534 possible colors. Also, with its fast frame rate thanks to the PCIe connection, the camera can capture 30 images per second, which is crucial for monitoring mobile infrastructures A.2. Another advantage of this camera is its compact design,which enables it to be mounted in our system. The PCIe connection is essential here, as the camera was connected to the embedded device via this connection, which transmitted the image data at high speed of $1GB/s$. The EF24-70mm f/2.8L II USM lens, see in Appendix A.1 from Canon was also attached to the camera itself, which can offer a wide range of possibilities thanks to its focal length of 24-70mm. Thanks to the maximum aperture of f/2.8, the lens can provide good image quality even in limited lighting conditions. The images acquired by this integration of camera and lens are then sent to the embedded device, and the intrinsic camera parameters 5 used in the pipeline.

3.3. Embedded Device

In this section, the embedded device used, its features, and their utilization in this thesis are described. An embedded device is a hardware component inside a more complex system. Unlike conventional computers designed for various general purposes, embedded devices are specialized for specific functions or applications. Due to their lower computing capabilities. Examples include controlling sensors to collect data or monitoring processes within the system [32]. The compact size of the device is also advantageous as it dose not occupy excessive space within the system, which was particularly crucial for more compact setups like the system of this thesis. As embedded devices are increasingly employed in critical environments such as monitoring or security systems, they must possess high reliability and robustness despite their compact design. Most embedded devices have various communication interfaces to receive data from other software within the system or to transmit data themselves. In this thesis, the embedded device served as the central interface between the various components, where captured images and data from the smartphone were used for ship detection and georeferencing using Python scripts, and the interaction of various software components. The embedded device utilized in this thesis was the Jetson AGX Xavier from NVIDIA [8]. The Jetson AGX Xavier is specifically tailored for computer vision applications, providing high computing power for neural networks, machine learning, and computer vision, despite its small size. The built-in powerful Central Processing Unit (CPU) and GPUs make the Xavier an useful embedded device for applications requiring collection and processing of sensor data. A 500 Gigabyte (GB) hard drive connected to the Jetson via Universal Serial Bus (USB) was used to store the images and the transformed georeferenced data. The system runs on Linux, specifically on Ubuntu 18.04 LTS, to enable communication with the rest of the software modules terminal is used.

3.4. Extrinsic Parameter Extraction using an Android Phone

Collecting extrinsic parameters is a crucial step in georeferencing images, typically involving specialized equipment. However, in this thesis, a unique approach was taken: an Android smartphone was used as opposed to using dedicated sensors. This offers several advantages, particularly in the project's prototype phase. Modern smartphones are equipped with various detection sensors, from gyroscopes to altimeters and compasses, which are essential for this thesis. Various localization and navigation options, including the Global Positioning System (GPS), used in this thesis to determine the phone's coordinates, are also provided. Moreover, phones provide the opportunity to collect data on the rotation axes, the system's altitude in relation to the environment. The smartphone's internet connectivity was another significant advantage, allowing for real-time data transfer from the system to a situational awareness room. A USB interface connects the phone and the embedded system, to transfer sensor data to the embedded device, and power from the device to the phone. This setup minimizes the system's power requirements and ensures its portability. The phone was positioned on the right outer wall of the housing to avoid interference with the reception as it will be shown in the calibration Chapter 4. The Ulephone module Power Armor 13 [13] was chosen for the system due to its relevant features and capabilities:

1. The necessary sensors like the compass, gyroscope, altimeter, and GPS reception to measure the parameters are incorporated.
2. The smartphone possesses a waterproof and dust-proof casing, ideal for outdoor use.

The Android interface was chosen for its easy access to the system's developer settings, simplifying the collection of necessary sensor data. The interface is accessed from the system using the terminal. The embedded device can address all the required sensors simultaneously. The collected extrinsic parameters include the lat and lon coordinates, which were determined using GPS. The rotation axes of the smartphone, known as yaw, pitch, and roll, are also determined using the smartphone. These parameters provide crucial information on the camera's behavior in space, essential for georeferencing the captured images. The system's altitude, indicating its height, is another collected extrinsic parameter. A Python script reads these parameter values from the phone's system, see Appendix A.2. The functionality of this script is detailed in the results of Chapter 5. These extrinsic parameters are then sent to the embedded device for georeferencing, following the same process as the intrinsic parameters, as it can be seen in diagram 3.1.

3.5. Ship Detection using YOLOv8

In order to georeference ships in images, they must first be recognized within the images. An object detection tool was utilized in this work, as introduced in Chapter 2.2. This section will concentrate on the tool employed in this thesis, the software YOLOv8 from Ultralytics [36]. YOLO is a deep-learning based object recognition architecture in computer vision. It was initially introduced in 2016 and was notably characterized by its efficiency in real-time recognition [36]. Unlike other approaches to object recognition, which require multiple steps to recognize objects, YOLO accomplishes object recognition in a single step, hence the name You Only Look Once. In contrast to other object recognition approaches, which focus on specific regions or sections, YOLO simultaneously examines the entire image. This enables YOLO to develop a comprehensive understanding of the image context and utilize this knowledge when recognizing objects. An example of YOLO's capability to recognize objects even in complex scenarios or environments, is depicted in Figure 3.2.

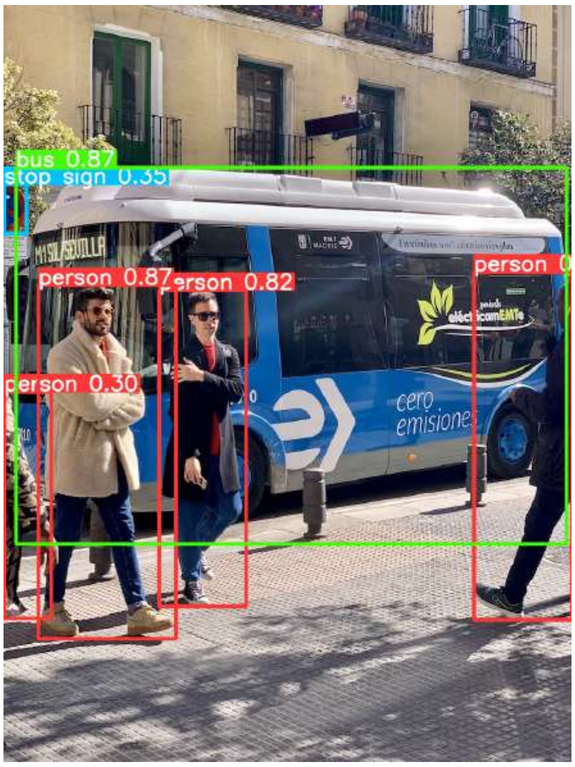


Figure 3.2.: Detection of different objects with the help of YOLO [19]

Even if objects are partially obscured, lighting conditions are poor, or the background varies greatly, YOLO can leverage the context of the entire image to accurately identify objects. This capability proves particularly useful for images of ships on the water [6]. YOLO can recognize connections and relationships between different objects, structures, and background elements by considering the entire image. This aids the model in localizing and classifying objects more accurately as it utilizes context to avoid potential misinterpretations, as shown in diagram 3.3. Due to this efficiency, the YOLO architecture is frequently employed in areas where systems must be able to act in real time [36]. These include the automotive industry in the field of autonomous driving, as well as surveillance systems and shipping. The diagram 3.3 shows the individual steps of object recognition by YOLOv8 and how they build on each other. In this thesis, a pretrained version of YOLOv8 was used; for this reason, the test dataset could directly be used to recognise the ships on high-resolution images, without the need of training.

A Python script was utilized to perform YOLOv8 in the embedded device. YOLOv8 has already been employed in previous work at DLR [6] and has therefore been adapted to the recognition of ships, which proved to be immensely beneficial for this thesis. After the ships in the images are recognized and highlighted with bounding boxes, the ships on high-resolution images.

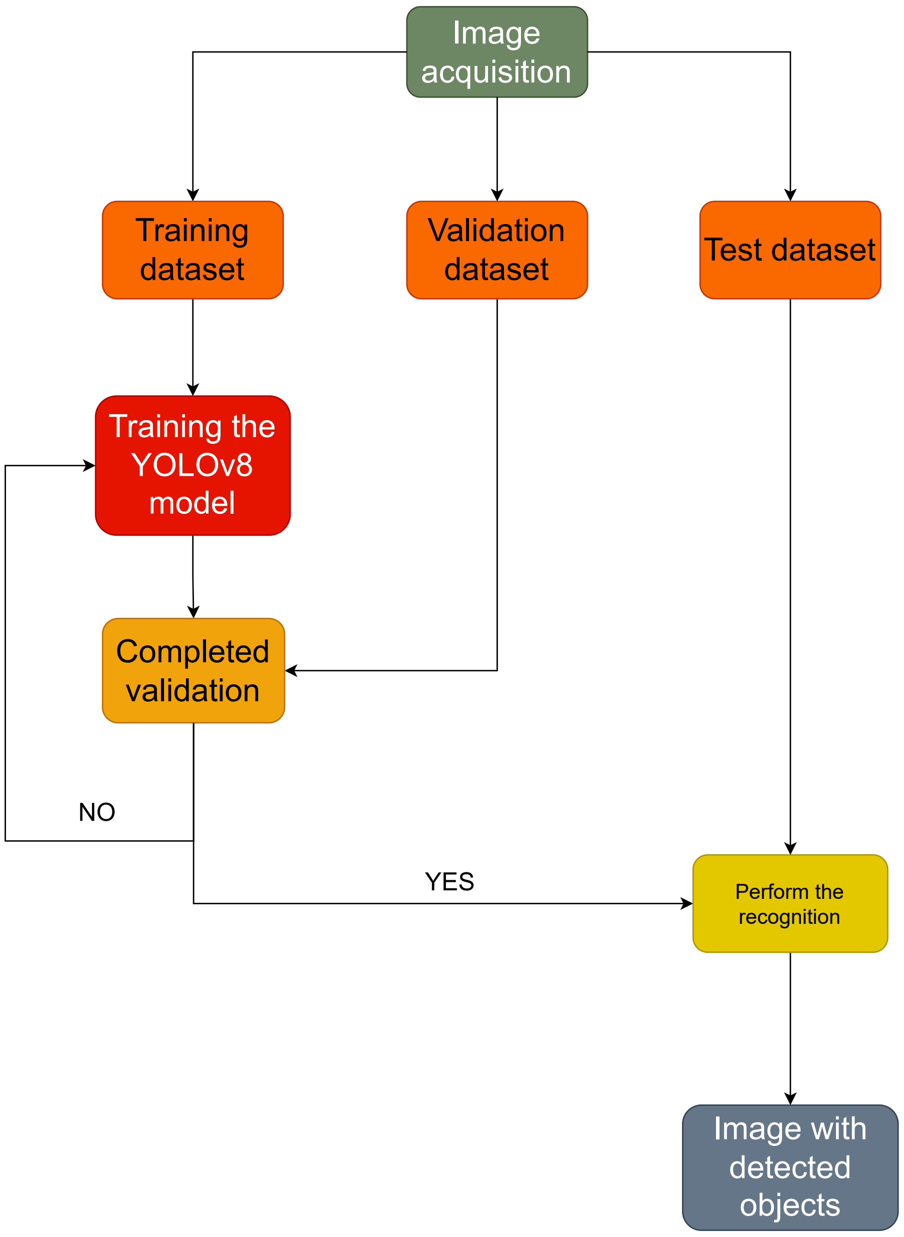


Figure 3.3.: The object recognition diagram of YOLOv8

3.6. Ship Georeferencing using Raycasting

The final step in the process was georeferencing the images. This step integrates all the preceding steps, including the intrinsic parameters of the camera and lens, extrinsic parameters collected with the assistance of an Android smartphone, and the information about the ships recognized using YOLOv8 as can be seen in Diagram 3.1. The objective was to georeference all visible ships in the images. Raycasting is the practical tool to implement the theoretical concepts presented in Chapter 2, providing details on how it is used in the actual georeferencing process within this thesis. The results in Chapter 5 provide a more detailed explanation of the corresponding Python script A.4. The advancement of computer graphics and image synthesis in the 1960s and 1970s significantly contributed to the emergence of the raycasting method [7]. This method found applications in various fields, including computer graphics, robotics, video game development, and physical simulations [29]. This thesis uses the raycasting method to determine the actual geographical coordinates of ships detected by YOLOv8 relative to the camera. In Figure 3.4 the principle of raycasting can be seen.

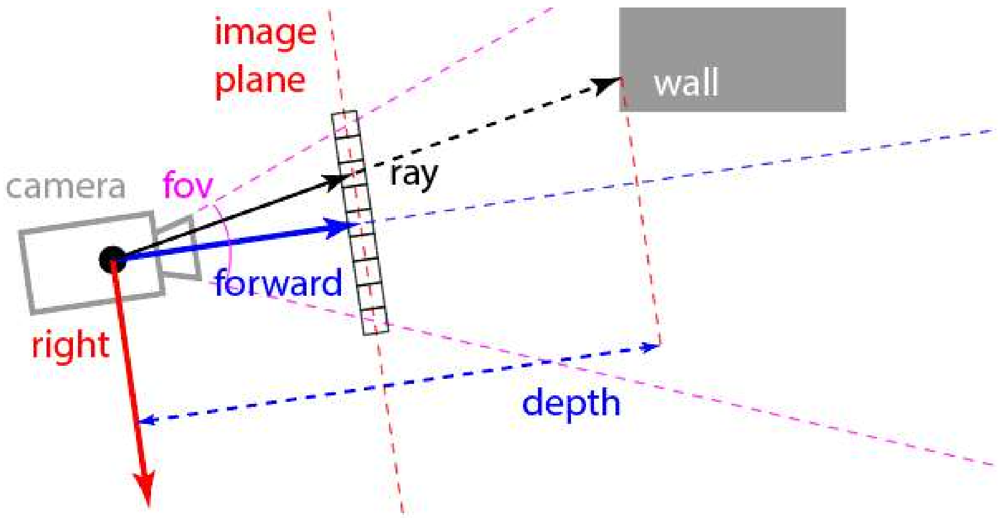


Figure 3.4.: The principle of raycasting [12].

The method involves establishing a starting point for the raycasting process for the corresponding pixel of each ship recognized and marked by YOLOv8. To achieve this, a ray, aligned with the position and viewing direction of the camera, is projected into the centre of the plane to be captured at the beginning of the raycasting method. This point then reflects the central pixel in the image. From this pixel, a ray is then created for each pixel within the image as seen in Figure 3.5. For each ship, the bottom line of the bounding boxes was selected as the georeferencing point. Precisely determining the starting point for raycasting is crucial to ensure accurate conversion of the bottom-centre pixel coordinates into actual geographic coordinates. The generated ray interacts with the objects or features in the image, including the detected ships. When a ray intersects with an object in the scene, an intersection point is created between the ray and the object hit. In this case, the centre of the bottom line of the bounding boxes in the image is considered the endpoint of the ray. These intersection points represent the 3D positions of the points in the scene seen by the corresponding pixel in the image. Once the 3D positions of the intersection points have been determined, the pixel coordinates of this point can be converted into geographical coordinates. To enable this, the extrinsic and intrinsic parameters of the camera are required [29]. Once this process is complete, the pixel coordinates of the referenced object are output in the converted geographical coordinates as shown in Chapter 4.

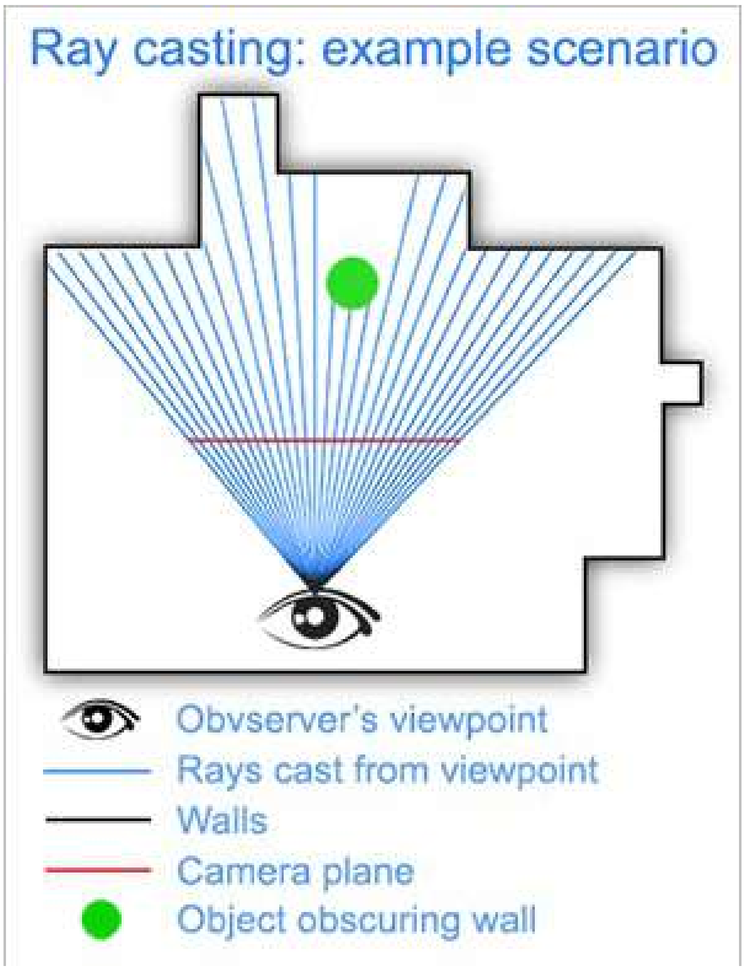


Figure 3.5.: Visualisation of the rays hitting a object [20].

4. System Calibration

This chapter explains the methodology and results for determining the position of the Android smartphone, on the housing, encompassing GPS, rotation angles, and altitude, through experimental investigations. The calibration of the system will allow to obtain accurate extrinsic parameters for ship georeferencing using the high-resolution images.

4.1. Positioning of the Phone for GPS Location

During both this thesis and the preceding hardware development phase, careful consideration was given to the optimal placement of the phone to obtain the extrinsic parameters of the system. Initially, the idea was to position it inside the housing alongside all other components. However, this approach faced significant drawbacks. The aluminium layers of the housing could obstruct the GPS signal and the internet connection. Additionally, the touchscreen function can not be used inside the housing, rendering all measurements unreliable and invalid [27].

Attaching the smartphone to the exterior of the system exposes it to environmental factors which could introduce deviations in the measured parameters, especially the GPS. Thus, it was necessary to ensure that the smartphone was mounted in a manner that minimized the impact of such effects. Therefore, before determining the mounting method, the correct placement of the phone needed to be calculated.

Potential positions for the phone included the top, left and right sides, as well as the back of the housing, see Figure 4.1. To align the positions on each side with that of the camera, the distance from the camera to each side was measured and projected accordingly. An experiment was conducted outdoors to identify the optimal option

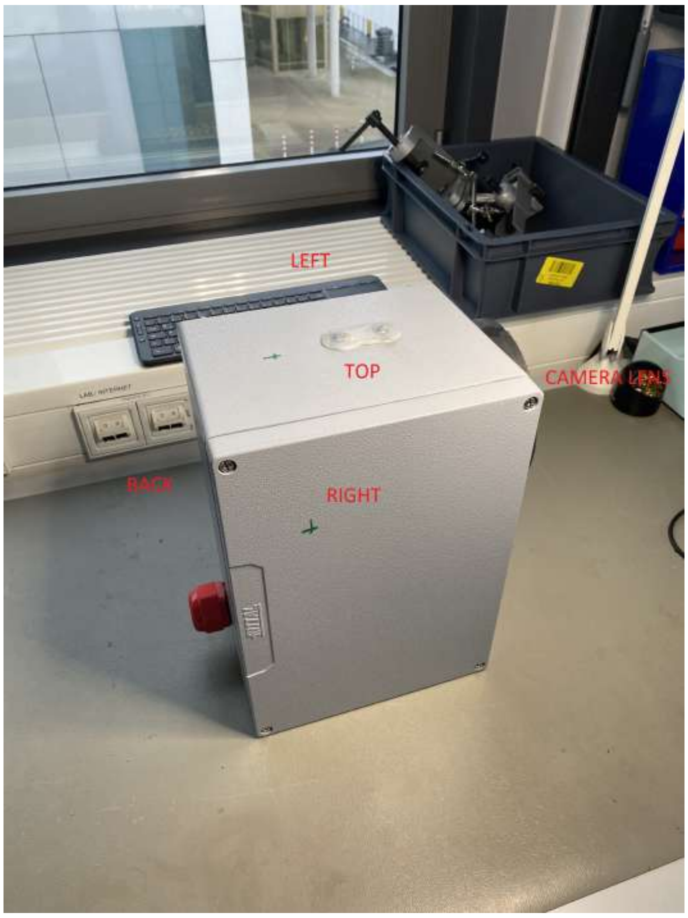


Figure 4.1.: Showcase of the housing, with markings for the possible positions for attaching the smartphone

among the four. Ten different positions were marked, and ten measurements were taken for each side, yielding 40 measurements in total. For each of the four sides, the average value and standard deviation for the latitude and longitude were calculated, as shown in equation 4.1 and 4.2 respectively.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4.1}$$

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{4.2}$$

To establish a reference for the values, a handheld GPS device (eTrex 32x from Garmin [16]) was utilized and also placed on the four dedicated sides that can be seen in Figure 4.1. For this calculation. The table 4.3 below compares the results of each respective side with the average value of the eTrex 32x 4.4, aiming to identify the best possible position. Here, the sixth decimal place indicates an accuracy in the centimeters range.

Position	Average Latitude	Average Longitude	Standard deviation Latitude	Standard deviation Longitude
Top side	53.521767°	8.583583°	1.75×10^{-5} °	1.83×10^{-5} °
Right side	53.521767°	8.583600°	1.21×10^{-5} °	1.17×10^{-5} °
Left side	53.521783°	8.583583°	1.92×10^{-5} °	1.89×10^{-5} °
Back side	53.521783°	8.583600°	1.76×10^{-5} °	1.72×10^{-5} °

(4.3)

Handheld GPS	Average Latitude	Average Longitude
eTrex 32x	53.521767°	8.583600°

(4.4)

Two significant conclusions can be drawn from the results of this experiment. Firstly, the position with the lowest standard deviation is the right side of the case, which holds for both the latitude and longitude values. Secondly, it is evident that the GPS coordinates collected by the Android smartphone are, on average, the uncertainty in the order of centimetres to those obtained with the handheld GPS device. A change only occurs after the sixth decimal place; before that, the values of the handheld and the phone are identical and lie at 53.521767° for the latitude and 8.583600° for the longitude. This underscores the validity and reliability of using the smartphone to measure the parameters.

Now that the optimal position for the smartphone has been identified, it can be securely mounted there to minimize the impact of external factors on the collected parameters. A bracket was designed, see Figure 4.2 and fabricated using a 3D printer to fix it to the right housing wall. The smartphone is securely held in place within the bracket, allowing for continued access to the display and the USB connection.

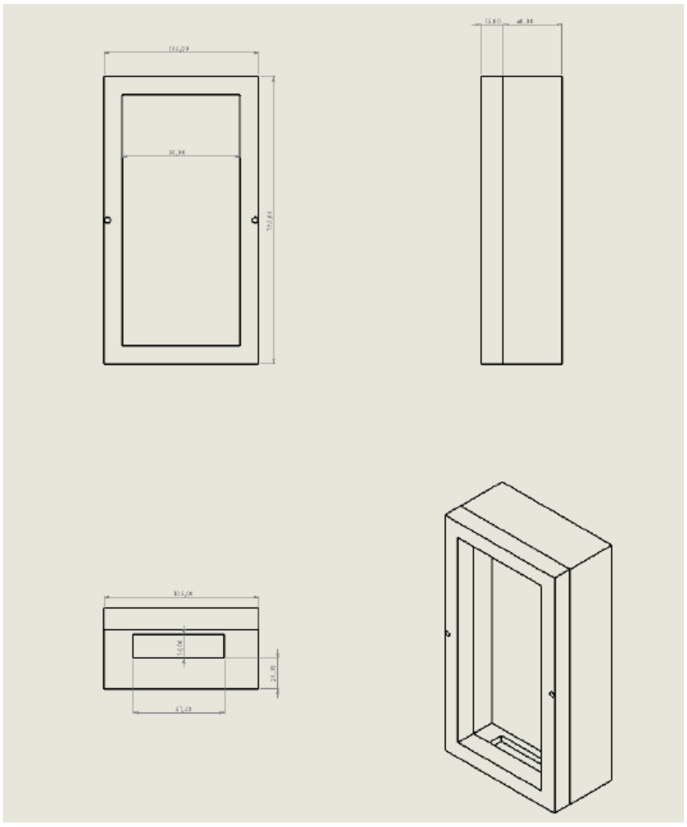


Figure 4.2.: Image of the holding bracket for the phone with the dimensions

4.2. Angles and Altitude calibration

With the Android phone now situated on the right side of the housing, it becomes crucial to calibrate its orientation including the compass and altitude of the system. To mitigate any potential deviations to ensure optimal extrinsic parameter obtention.

4.2.1. Calibration of the Angles

The calibration of the angles is an essential step because the system orientation must be chosen with respect to respect the earth's coordinates. To ensure alignment between the phone's three rotation angles (pitch ϕ, yaw θ, and roll ψ) and the camera, it is essential to account for any offsets introduced by the phone's positioning relative to the camera. During the hardware design and assembly of the system preceding this Bachelor thesis, the camera's position was adjusted to form angles of 0° relative to the housing walls. Therefore, when the phone is affixed to the right side, as depicted in Figure 4.3, the offset angles between the phone's position and the camera can be projected.

To obtain reliable measurements of the phone angles with respect to the camera, the smartphone is placed in its final position on the right side of the housing. Ten measurements are taken at this position, and then the average value and the standard deviation for each of the three rotation angles are are computed. The results can be seen in Table 4.5.

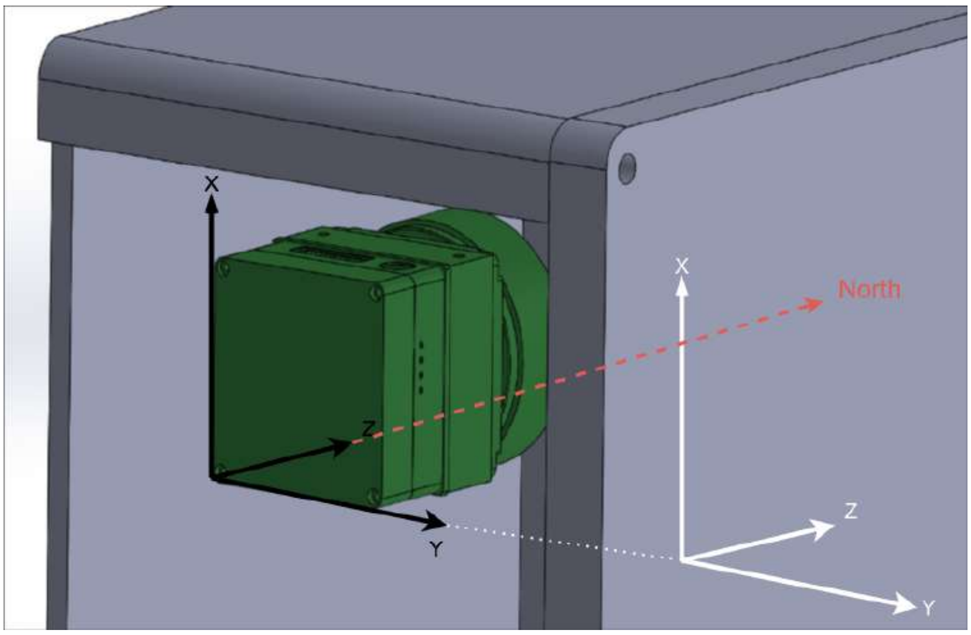


Figure 4.3.: Angle projection of the camera onto the housing sides

It's important to note that the housing is is calibrated while being oriented towards the north (z), such that the compass of the phone and the camera are calibrated at

0°, where:

- Roll (ψ) is the angle between the axes X and Y.
- Pitch (ϕ) is the angle between the axes X and Z.
- Yaw (θ) is the angle between the axes Y and Z.

Angels	Average	Standard deviation
Pitch ϕ	-1,10°	0,26°
Yaw(north) θ	0,57°	1,88°
Roll ψ	88,69°	0,04°

(4.5)

The values indicate that the assumption of 90° for Roll (ψ) shows an average value of $88.69\pm0,04$. The Pitch (ϕ) and Yaw (θ) show a offset of $-1,1\pm0,3$ and $0,6\pm1,9$, respectively. The calibration values for ϕ, θ and ψ are used in the georeferencing process to compensate the angular offset between the values measured by the phone and the actual camera angular values.

4.2.2. Offset and Calculation of the Altitude

In addition to the extrinsic parameters, the android phone was also used to measure the altitude of the system, the altitude is given in the formula 2.1 as T_y. This is crucial for the georeferencing in the Chapter 5 parameter, as it requires information about the height difference between the system and the water on which the ship is located. It was noticeable, during the calibration, that the phone measurement for the altitude did not match the values of the area of the world the phone was placed in. For example, in Bremerhaven, the expected altitude was 2 meters. However, the phone would provide approximately 42 meters when measuring on the ground in different parts of the city. This disparity arises because the smartphone employs the World Geodetic System 1984 (WGS84) to determine altitude. The WGS84 is a global reference system for surveying and mapping [10], with a crucial component being the geoid, which represents the theoretical shape of the Earth's surface considering gravity [21]. However, the geoid is imperfect, featuring irregularities due to mass distribution within the Earth, as depicted in Figure 4.4. Consequently, height measurements are expressed relative to the geoid, with positive altitudes denoting positions above the geoid and negative altitudes indicating positions below it [10].

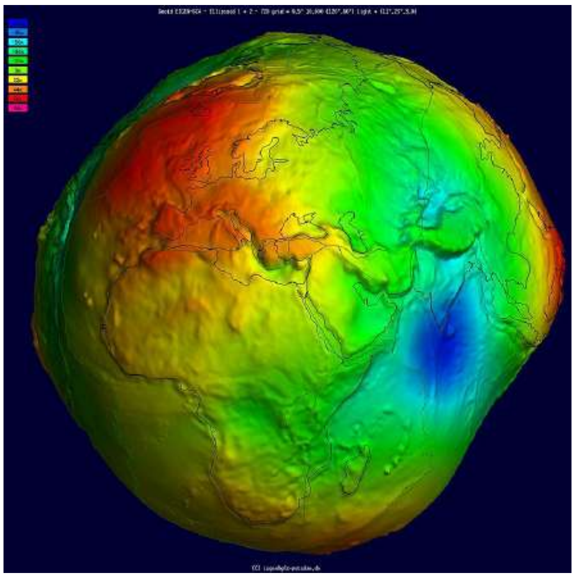


Figure 4.4.: 3D module of earth's geoid [18]

Figure 4.5 illustrates a map depicting the variations in altitude attributed to the geoid's influence. This visualization aided in identifying an offset of approximately 40 meters. Subsequently, after deducting the 40 meters inferred from the smartphone measurements, the resulting altitude value also aligned with the 2 meters altitude determined by other systems.

In order to empirically validate this assumption, another four sets of 10 altitude measurements were conducted, encompassing four distinct elevation levels for comparative analysis. These 40 measurements were executed on the institutes rooftop, 2nd floor, ground level, and water level in the port basin Fischereihafen. For each position, the average value and standard deviation were computed from the ten acquired results per set, as depicted in Table 4.6. The two highlighted lines represent the pivotal values for subsequent georeferencing.

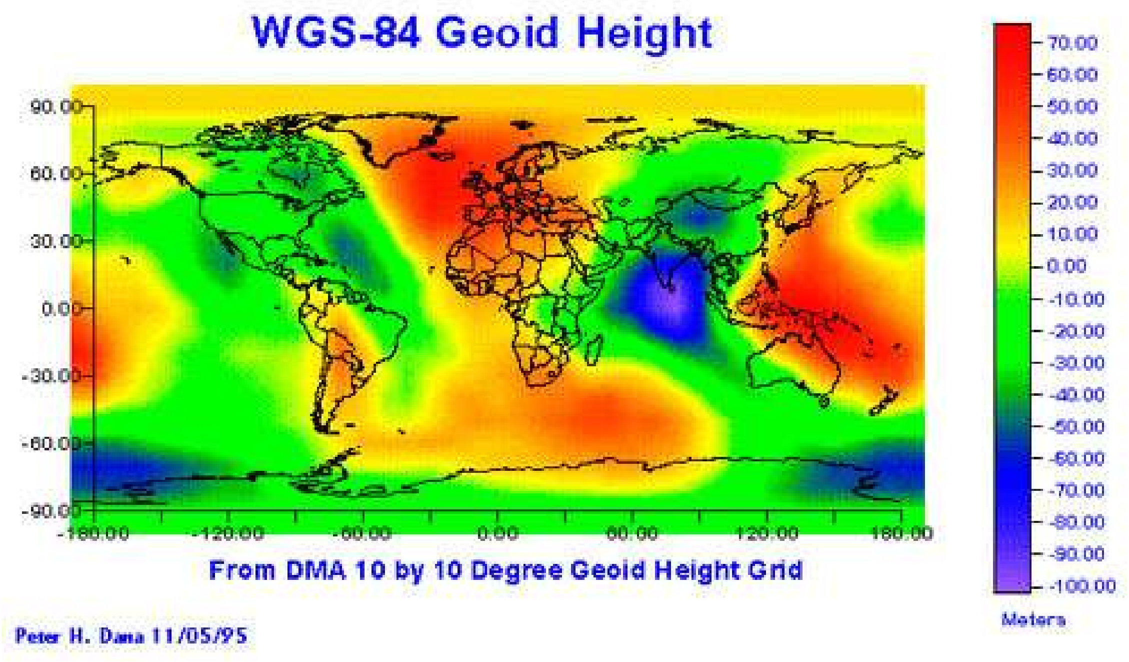


Figure 4.5.: Map showing the WGS84 geoid hight [2]

Position	Average phone Altitude	Standard deviation phone Altitude
Rooftop	54,510m	0,097m
Second floor	51,550m	0,110m
Ground	40,670m	0,067m
Water	38,340m	0,136m

(4.6)

The experiment for ship georeferencing shown in Chapter 5 are conducted by placing the system on the 2nd floor at the window facing the ship on the water in the port basin in front of the institute. We can observe that the water level, expected at 0m, presents an average altitude of 38.340m. When the camera is placed at the 2nd floor, the height difference between the 2nd floor and the water shows the altitude between the ship to be georeferenced and the camera. In light of these findings, the altitude utilized for georeferencing within the Python script is determined to be the average value from the 2nd floor subtracted by the average of the water level, with the result for the altitude of 13,210m, and this is the value used in the code of the script shown in A.4. For georeferencing and that will be explained in Chapter 5.

5. Experimental Set Up And Results

This chapter presents the conclusive experiment of this thesis: the offset values calculated in the previous Chapter 4 are used here in the Python script for georeferencing of ships on images captured with a high-resolution camera. Initially, the integration of the image acquisition process will be explained. Subsequently, the collection of sensor data from the Android smartphone will be addressed. Then, YOLOv8 is used for ship detection and raycasting for ship georeferencing. The goal of the experiment is to show quantitatively how the ship georeferencing works when using the integrated system.

5.1. Image Acquisition

This section focuses on the practical collection of images using the high-resolution camera and a connected lens. Initially, the system was installed at 13,210 meters altitude (2$^{\text{nd}}$ floor), as demonstrated in Chapter 4, facing the harbor basin, to capture the images. After the calibration offset calculated in the previous chapter, the system setup is illustrated in Figure 5.1 below.

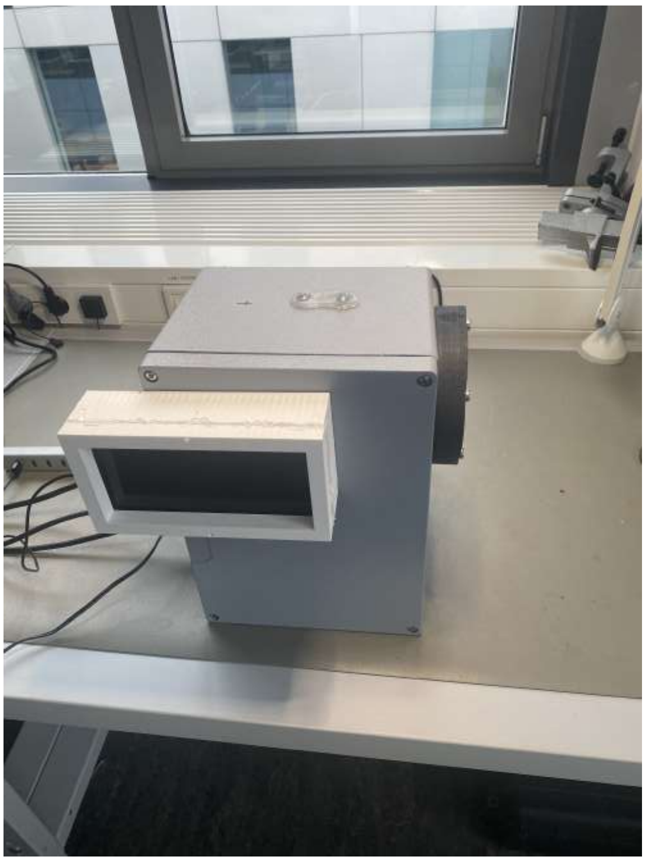


Figure 5.1.: Image of the completed system

To facilitate this process, the integration of two Python scripts is required. The first script adjusts the camera settings, including the exposure time, lens aperture, frames per second (FPS), and duration of the acquisition in minutes. These parameters are used to select the quality of the captured images which are 50 Megapixel large, and the number of images in the sets. The snippet of the configuration file 5.1 shows how these parameters can be configured in a JSON file. During the pipeline test, these parameters were adjusted to capture ten images per image set. Three sets of ten images were used for the experimental evaluation of ship georeferencing, allowing for the determination of average values and deviations in latitude and longitude coordinates for each ship marker. This approach aimed to assess the system's accuracy and precision in detecting and georeferencing a ship. Using the high resolution camera images, its intrinsic parameters, and the extrinsic parameters acquired with the Android phone.

```
Automatic gain and exposure time (True/False):   "False"
Exposure time (usec, not valid when automatic is True):     100000
Lens aperture f/ [from 1.8 to 22]:  17.5
Image data format:  "XI_RGB24"
Trigger (Internal/External):    "Internal"
Timeout (msec):     10000
Framerate (FPS, not valid when external trigger is True): 1
Duration (min, not valid when external trigger is True):  0.167
Automatic white balance (True/False):    "True"
```

Listing 5.1: An example of the acquisition parameter for the camera

Once the camera settings are configured, the next step is to capture the image series. This is achieved by executing a Python script on the embedded device. The complete code for this script can be found in the code snippet labeled A.1. After modifying the script to capture an image from the designated image set, it is restarted to capture subsequent images if any adjustments to the acquisition parameters are necessary. Figure 5.2 shows an example of a 50 Megapixel image of the ship which was taken with the help of the integrated system from the second floor of the institute, this is the acquisition part of the Diagram 3.1. In addition to the script for creating the image series, another script runs simultaneously, as described in the next section. In compliance with data protection guidelines, areas of the image not belonging to the water were anonymized automatically (blurred) during the acquisition process.

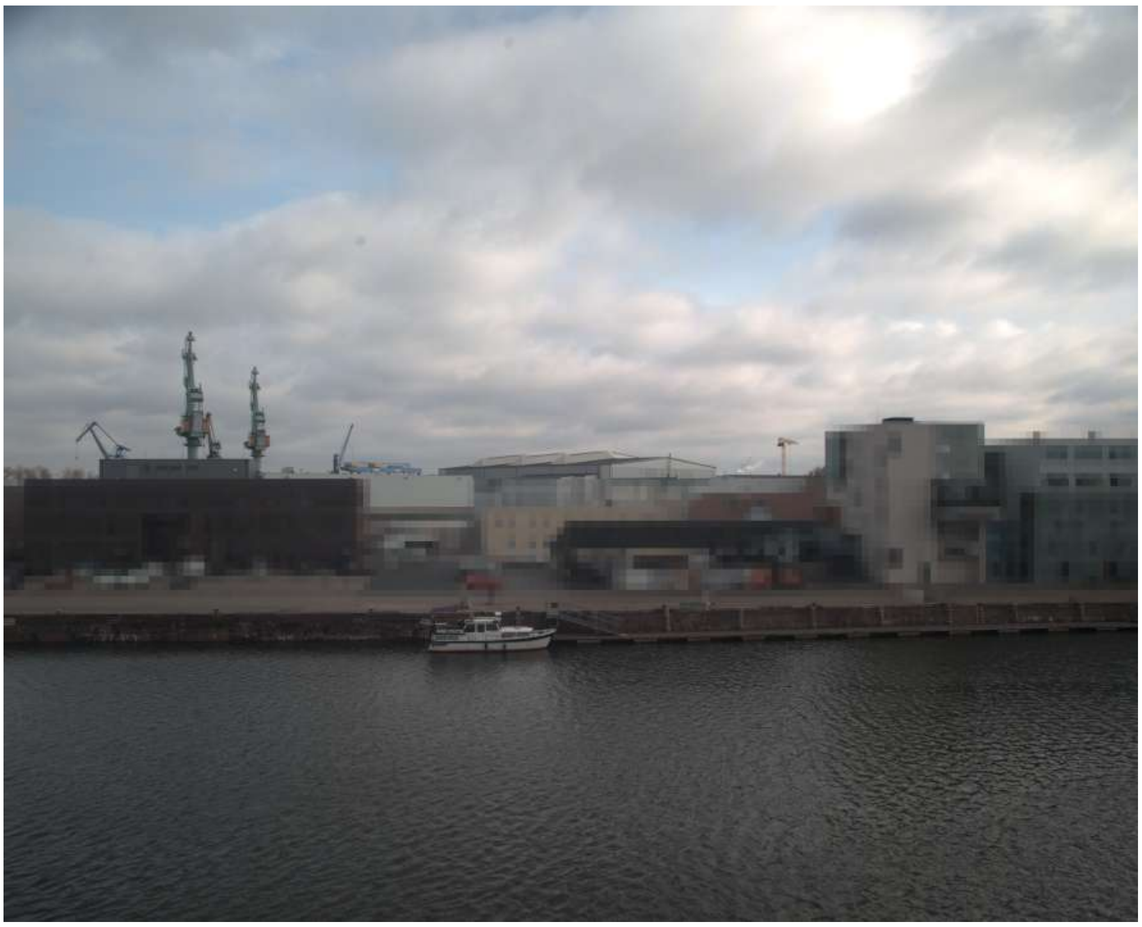


Figure 5.2.: 50 Megapixel image of a ship, taken with the integrated system from the second floor of the DLR.

5.2. GPS, Altitude and Orientation Data Acquisition

As shown in the diagram of the pipeline 3.1, a concurrent script is executed on the embedded device. This device is addressed and controlled via the terminal of the Jetson,which is connected to the phone via USB, to generate image sets while collecting extrinsic parameters from the smartphone. To achieve this, the phone must be accessed by the command `adb shell dumpsys`. The `adb shell dumpsys` command, when executed on a Linux system connected to an Android device via USB, generates a detailed report on the device's system services. This command is a diagnostic tool used to assess and debug the Android operating system and its services. It provides information on battery status, app activities, memory usage, and internal sensor data, by listing statuses and metrics from various system components.

The commands for retrieving sensor data and the sensors themselves are explained as follows. The code is referenced in the appendix see A.2. The initial stage of this script entails implementing date and time functions, crucial for synchronizing the creation of the photo series with the corresponding sensor data.

Subsequently, the script extracts the GPS data information from the android phone. In this context, the script selects the keyword *"fused"* as depicted in code snippet 5.2. The choice of the keyword "fused" is based on the observation that multiple packages related to GPS services are present when querying the smartphone's sensor service files. However, only the package labeled "fused" is relevant as it provides the most accurate GPS data. The command *"adb shell dumpsys location | grep last"* is employed to extract the smartphone's last-known GPS position in the Android phone, ensuring that the current position is reported each time the command is executed. Alongside the latitude and longitude of the system, the script receives additional information which are essential for the ship georeferencing. Such as the Horizontal Accuracy (hAcc), which denotes the radius of deviation of the system. Lastly, the script also retrieves the altitude data of the phone, as determined in the Section 4.2.2. The offset of 38.340m must be taken into account; it is essential to note that this offset can vary depending on the system's location.

```
current_datetime = datetime.now()
target_word = "fused"
pattern_1 = re.compile(rf'' + re.escape(target_word) + ' ([^\s]+)
   hAcc=([^\s]+) et=[^\s]+ alt=([^\s]+)')
command = "adb shell dumpsys location | grep last"
```

Listing 5.2: Excerpt from the GPS data recording

The second part of the script involves gathering values for the orientation (ϕ, θ and ψ) of the smartphone. This data extraction is illustrated the Appendix A.2. Similar to the GPS measurement, the command *"adb shell dumpsys"* is utilized to access the smartphone. The term *"sensorservice | grep -A 10"* specifies the orientation sensor for the three axes. The segment *"ORIENTATION: last ten events"* retrieves the last ten measurements received from the sensor, with each measurement comprising three values representing each rotation angle. This process is iterated ten more times to accumulate 100 measurements, each with three angles. Subsequently, the average of all these determined values is computed, and the output is formatted within the complete script.

Additionally, a provision is added to ensure the creation of a text document to store the script's collected results. Every time a new measurement is taken, it generates a new measurement for each run. This is crucial for subsequent steps, as the camera's extrinsic parameters in space for each photo set are required for reference.

5.3. Using Raycasting and YOLOv8

This section delves into the practical implementation of YOLOv8 for ship detection and raycasting for georeferencing, which are shown in diagram 3.1. Both processes run simultaneously in the embedded device. Here, ships can be directly detected and classified in images by YOLOv8 and then georeferenced by raycasting, as depicted in Appendix A.4. It is crucial to note that neither the object recognition software nor the code for raycasting itself was developed as part of this thesis. This work consists in the integration of these tools and proves that, together with the extrinsic parameters, they can be used to georeference ships on high-resolution systems using an embedded device. The code A.4, which encompasses both ship recognition and raycasting, is preceded by a separate script for executing raycasting, which is also available in Appendix A.3.

All previously collected extrinsic and intrinsic information is compiled and executed in this script. As seen in code snippet 5.3, the first parameters to be entered are the height and the width of the image to be referenced in pixels; these intrinsic parameters are the same for all images of the camera in the current setting. The FoV is also a fixed intrinsic parameter of the camera in combination with the used lens. This was calculated with formula 5.1. $FoV_{\text{horizontal}}$ represents the horizontal field of view, Sensor width denotes the camera sensor's width, and Focal length the lens's focal length. The values of those variables can be extracted from the respective camera datasheets A.2 and A.1.

$$FoV_{\text{horizontal}} = 2 \cdot \arctan\left(\frac{\text{Sensor width}}{2 \cdot \text{Focal length}}\right) \tag{5.1}$$

Following the intrinsic parameters, we collect the extrinsic parameters using our mobile georeferencing system.

When specifying the values for the rotation angles (ϕ, ψ, θ), it is important to deduct the average values of each angle calculated in Chapter 4 from the phone's values see 5.3. The variable `tz` is the altitude between the 2nd floor of the institute and the water level of the port basin area as calculated in section 4.2.2. The last extrinsic parameter collected is the current GPS position of the system, which is also extracted with the phone, at the time the images were taken, as these are not permanent due to the mobile setup of the system. These steps were repeated for each image set with their corresponding parameters.

```python
def main():
    # Define camera setup parameters for georeferencing
    w, h = 7920, 6004  # Image dimensions (width, height)
    fov = np.radians(74.35)  # Field of view in radians

    theta = np.radians(40.10-1.10)  # Camera pitch angle in radians
    phi = np.radians(1.93-0.57)  # Camera yaw angle in radians
    psi = np.radians(88.74-88.69)  # Camera roll angle in radians
    tz = 13.21  # Camera height above the water in meters

    # Convert the camera's geographic location to UTM coordinates
        latitude, longitude = 53.522028, 8.583522
```

Listing 5.3: Insert extrinsic and intrinsic parameters into the georeferencing code

Finally, the path of the image to be referenced must be inserted into the script. The selected image is then run by YOLOv8 to detect the ship automatically. In this case, a bounding box is created around the object to be recognized. In the example of this thesis, the x and y pixel coordinates of the bottom center of the box, which are relevant for the raycasting, are determined as shown in the image, see Figure 5.3.

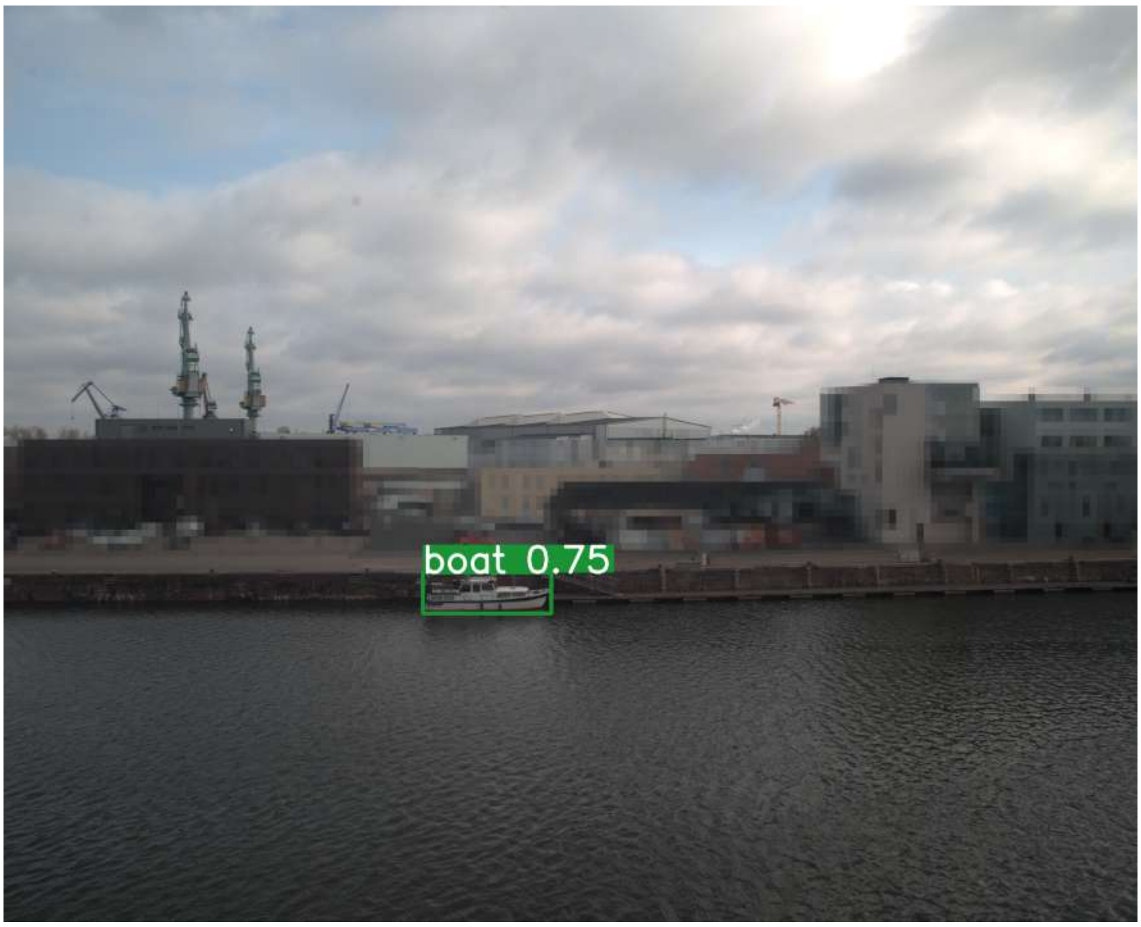


Figure 5.3.: Image of the georeferenced ship before

The image and ship detections are saved in a folder created for this purpose and then processed directly in the raycasting script to convert the latitude and longitude from the pixel coordinates of the bottom bounding box center. The determined coordinates are then returned to the system and displayed in the console. Additionally, it was implemented in the script that after the raycasting is completed, an online map opens using Leaflet map [1] via Folium [14], where the position of the determined GPS coordinates is displayed. This allows to quickly assess the result and its accuracy.

5.4. Final Results and Calculation

As mentioned at the beginning of the chapter, three sets of each ten images were taken with the camera and georeferenced by the script. This section calculates the validity of these measurements and presents the final results of this thesis by assessing the georeferencing accuracy of the integrated system.

5.4.1. Calculation of the Results

In order to assess both the accuracy and precision of the system, the average value and the standard deviation were determined from the results, using formulas previously described in Chapter 4. For comparison with the actual GPS coordinates of the ship, the actual latitude and longitude of the ship were obtained on-site with the assistance of the GPS handheld device. Ten measurements were again carried out, yielding average values for the ship's latitude of 53.522665° and longitude of 8.583962°. These values are considered the real coordinates of the ship for further calculations, in this thesis and are utilized as a reference for comparison. The display of the GPS coordinates in the decimal degree format by the system should be noted, which is why the ship coordinates were also provided in this format. Here, the sixth decimal place indicates an accuracy in the cm range.

In order to be able to make precise statements about the distance between the individual GPS coordinates, these were calculated using the haversine formula 5.2. The haversine formula takes into account the curvature of the earth and, therefore, provides more accurate distances between points on the earth's surface compared to a Euclidean calculation suitable for flat surfaces.

$$d = 2r \arcsin\left(\sqrt{\frac{1-\cos(\varphi_2-\varphi_1)+\cos(\varphi_1)\cdot\cos(\varphi_2)\cdot(1-\cos(\lambda_2-\lambda_1))}{2}}\right), \quad (5.2)$$

where:

- d is the distance between the two points,

- r is the radius of the sphere (for example, Earth's radius),
- φ_1, φ_2 are the latitudes of the two points,
- λ_1, λ_2 are the longitudes of the two points.

The subsequent table presents the average, standard deviation and the distance of the 30 coordinates obtained after raycasting. During the image collection process, the images were divided into three sets of 10 images each. After every set of 10 images, the camera's position was altered to simulate the system's mobility. Consequently, the average value, standard deviation and the distance for each of the three sets were initially calculated, as illustrated in the table. The final georeferenced values were then determined in the following Table 5.3.

Image set	Average ship Latitude	Average ship Longitude	Standard deviation Latitude	Standard deviation Longitude	Average Distance	Standard deviation Distance
Image 1-10	53.522718°	8.583745°	4.64×10^{-5} °	5.35×10^{-5} °	15,6m	6,55m
Image 11-20	53.522715°	8.583741°	4.57×10^{-5} °	5.31×10^{-5} °	15,7m	6,64m
Image 21-30	53.522712°	8.583749°	4.51×10^{-5} °	5.33×10^{-5} °	15,1m	6,32m
Final value	53.522715°	8.583745°	4.57×10^{-5} °	5.33×10^{-5} °	15,4m	6,50m

(5.3)

To evaluate the accuracy of the measurements, the final average values of latitude and longitude were compared with the values collected from the referenced ship.

In Figure 5.4, we compare the final determined GPS coordinates obtained through raycasting with the real ship coordinates. Upon conducting this test series, the error between the georeferencing coordinates and the real ship coordinates was 16 meters. For the uncertainty,the maximum value was taken, which is 6.64m, and approximate it to 7m as shown in the Table 5.3.

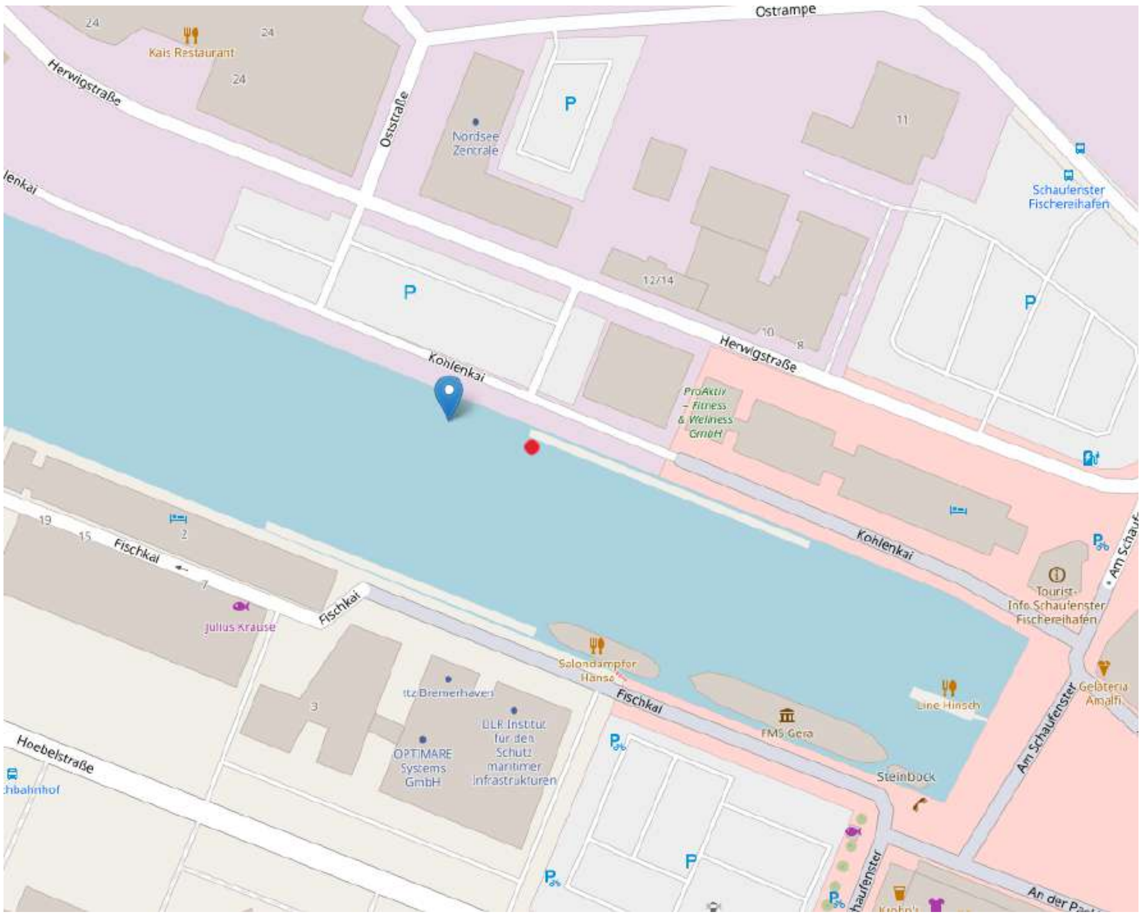


Figure 5.4.: Showing the offset between the georeferenced average GPS coordinates (blue) and the coordinates of the ship (red) [1]

At the end, the total error for the georeferencing system is determined to be: 16 ± 7 meters. Comparison with the values of the offset from previous work, such as [6], reveals that a value for the offset of 18 ± 13 was collected. However, as opposed to previous work, this result was achived by using the sensors of a android smartphone and a embedded high-resolution camera system. These results prove that the hardware and software integration of the pipeline 3.1 have proven to work together in this specific setup.

6. Conclusions

In this chapter, the integration for the high-resolution embedded camera for ship detection and georeferencing. The completion of the goals are discussed, and ideas for possible improvements and further development are offered.

6.1. Contributions

This thesis, in Chapter 3, presents a pipeline for integrating ship detection and georeferencing using an industrial camera, embedded device, and smartphone for data collection. The embedded device processes images and data collected by the smartphone to determine the precise position of a ship using the robust raycasting principle after detecting it using YOLOv8. This approach offers numerous benefits regarding of cost-effectiveness, reliability, and accuracy, providing an effective and efficient solution for ship detection and georeferencing.

Chapter 4 discusses the placement of an Android smartphone on the housing to obtain accurate extrinsic parameters for ship georeferencing using high-resolution images. After conducting experiments, it was determined that the optimal position for the phone was on the right side of the housing, which resulted in the lowest standard deviation for both latitude and longitude values. To secure the smartphone in this position, a bracket was designed and fabricated using a 3D printer. It was also crucial to calibrate the orientation and altitude of the system to ensure precise georeferencing. Calibration of the phone's angles was performed to align with the earth's coordinates, and the offset angles between the phone's position and the camera were projected to obtain reliable measurements of the phone angles with respect to the camera.

The mobile prototype developed during this project can recognize and georeference ships in images, as demonstrated in Chapter 5. This achievement was made possible by integrating the high-resolution camera with the Android smartphone and other system components. A system calibration was performed to align the extrinsic parameters of the smartphone sensors with those of the camera, enabling the camera's movement to be accurately represented by the smartphone. The experiment resulted in a georeferencing error of 16 ± 7 meters, which compares to the result of stationary systems, however now offering a mobile approach.

6.2. Future Work

While the system has demonstrated its functionality, there are exciting opportunities for further improvement in its current state. Future projects could explore the application of the pipeline in a real-time use case, measuring the timings and performing optimizations were needed. This enhancement would allow the camera to capture images and georefernece ships in real-world scenarios, and to send the ship coordinates to a situational awareness system. Another promising avenue is finding a solution for adjusting the zoom level, which could be achieved by controlling a servo motor. Once the system's performance and functionality are further validated through testing, more extensive field missions can be planned to demonstrate how mobile object recognition systems equipped with high-resolution cameras can enhance maritime monitoring.

List of Figures

Listings

.

Bibliography

[1] Volodymyr Agafonkin. an open-source javascript library for mobile-friendly interactive maps. https://leafletjs.com/.

[2] Mitchell Baldwi. Overview ellipsoid spheroid geoid datum projection coordinate system. https://slideplayer.com/slide/13618396/.

[3] A. Bartoli and P. Sturm. The 3d line motion matrix and alignment of line reconstructions. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[4] Bajramovic Brückner. Intrinsic and extrinsic active self-calibration of multi-camera systems. *Machine Vision and Applications*, 25:389–403, 2014.

[5] Borja Carrillo-Perez, Sarah Barnes, and Maurice Stephan. Ship segmentation and georeferencing from static oblique view images. *Sensors*, 22(7), 2022.

[6] Borja Carrillo-Perez, Angel Bueno Rodriguez, Sarah Barnes, and Maurice Stephan. Improving yolov8 with scattering transform and attention for maritime awareness. In *2023 International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 1–6, 2023.

[7] M.F. Cohen and J.R. Wallace. *Radiosity and Realistic Image Synthesis.* Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Elsevier Science, 1993.

[8] NVIDIA Corporation. Jetson agx xavier. Datasheet available online: https://ausdroid.co/wp-content/uploads/2020/08/Jetson-AGX-Xavier-Series-Datasheet.pdf.

[9] Gonçalo Cruz and Alexandre Bernardino. Aerial detection in maritime scenar-

ios using convolutional neural networks. In *Advanced Concepts for Intelligent Vision Systems*. Springer International Publishing, 2016.

[10] B LOUIS Decker. World geodetic system 1984. *Defense Mapping Agency Aerospace Center St Louis Afs Mo*, 1986.

[11] M.P. Deisenroth, A.A. Faisal, and C.S. Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[12] Game Development. Understanding the rendering of the raycasting on flat screen. https://i.stack.imgur.com/5q6Ex.png.

[13] Shenzhen Gotron Electronic. Ulefone power armor 13. Product page available online: https://www.ulefone.com/power-armor-13.html?PageName=specs.

[14] Folium. Python data, leaflet.js maps. https://python-visualization.github.io/folium/latest/getting_started.html.

[15] Institute for the Protection of Maritime Infrastructure. Methods and processing group. Available online: https://www.dlr.de/mi/en/desktopdefault.aspx/tabid-13115/22887_read-53251/.

[16] Garmin. etrex 32x. Product page available online: https://www.garmin.com/de-DE/p/669215.

[17] Amanda Geniviva, Jason Faulring, and Carl Salvaggio. Automatic georeferencing of imagery from high-resolution, low-altitude, low-cost aerial platforms. In Donnie Self, Matthew F. Pellechia, Kannappan Palaniappan, Shiloh L. Dockstader, Paul B. Deignan, and Peter J. Doucette, editors, *Geospatial InfoFusion and Video Analytics IV; and Motion Imagery for ISR and Situational Awareness II*, volume 9089, page 90890D. International Society for Optics and Photonics, SPIE, 2014.

[18] gfz potsdam. Visualization of gravity field models and their differences. https://icgem.gfz-potsdam.de/vis3d/longtime.

[19] Github. Pytorch yolov5 but with different results.

[20] Computer Hope. Ray casting. https://www.computerhope.com/jargon/r/ray-casting-diagram.png.

[21] Xiong Li and Hans-Jürgen Götze. Ellipsoid, geoid, gravity, geodesy, and geophysics. *Geophysics*, 66(6):1660–1668, 2001.

[22] Rina Mardiati, Edi Mulyana, Iyon Maryono, Koredianto Usman, and Tedi Priatna. The derivation of matrix transformation from pixel coordinates to real-world coordinates for vehicle trajectory tracking. In *2019 IEEE 5th International Conference on Wireless and Telematics (ICWT)*, 2019.

[23] Vincent Marié, Ikhlef Béchar, and Frédéri Bouchara. Towards maritime video-surveillance using 4k videos. In *Smart Multimedia*. Springer International Publishing, 2018.

[24] Vincent Marié, Ikhlef Béchar, and Frédéric Bouchara. Real-time maritime situation awareness based on deep learning with dynamic anchors. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018.

[25] M. McNicholas. *Maritime Security: An Introduction*. Elsevier Science, 2016.

[26] B. Micusik and T. Pajdla. Structure from motion with wide circular field of view cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1135–1149, 2006.

[27] Ahmed S. Mohamed, Mohamed I. Doma, and Mostafa M. Rabah. Study the effect of surrounding surface material types on the multipath of gps signal and its impact on the accuracy of positioning determination. *American Journal of Geographic Information System*, 8(5):199–205, 2019.

[28] Cornelia Nita and Marijke Vandewal. Cnn-based object detection and segmentation for maritime domain awareness. In *Artificial Intelligence and Machine Learning in Defense Applications II*, volume 11543, pages 13–21. SPIE, 2020.

[29] Goran Paulin, Sasa Sambolek, and Marina Ivasic-Kos. Application of raycast method for person geolocalization and distance determination using uav images in real-world land search and rescue scenarios. *Expert Systems with Applications*, 237:121495, 2024.

[30] Dilip K. Prasad, Deepu Rajan, Lily Rachmawati, Eshan Rajabally, and Chai 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, 2017.

[31] RITTAL. Cast aluminium enclosures ga. Available online: https://www.rittal.com/de-de/products/PG0002SCHRANK1/PG0003SCHRANK1/PG0011SCHRANK1/PRO0009?variantId=91192

[32] Elias T. Silva, Fausto Sampaio, Lucas C. da Silva, David S. Medeiros, and Gustavo P. Correia. A method for embedding a computer vision application into a wearable device. *Microprocessors and Microsystems*, 76:103086, 2020.

[33] Jia Song, Shaohua Gao, Yunqiang Zhu, and Chenyan Ma. A survey of remote sensing image classification based on cnns. *Big Earth Data*, 3(3):232–254, 2019.

[34] Ioannis Stellios, Panayiotis Kotzanikolaou, Mihalis Psarakis, Cristina Alcaraz, and Javier Lopez. A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services. *IEEE Communications Surveys & Tutorials*, 20(4):3453–3495, 2018.

[35] M.A. Sutton, J.J. Orteu, and H. Schreier. *Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts,Theory and Applications*. Springer US, 2009.

[36] Ultralytics. Yolov8. Blog post available online: https://www.ultralytics.com/de/blog/ultralytics-yolov8-turns-one-a-year-of-breakthroughs-and-innovations.

[37] Jian Wu, Liwei Ma, and Xiaolin Hu. Predicting world coordinates of pixels in rgb images using convolutional neural network for camera relocalization. In *2016 Seventh International Conference on Intelligent Control and Information Processing (ICICIP)*, 2016.

[38] XIMEA. Cb500cg-cm. Technical manual available online: https://www.ximea.com/downloads/cb/manuals/xib_xib64_technical_manual.pdf.

[39] Zhengyou Zhang. *Camera Parameters (Intrinsic, Extrinsic)*, pages 135–140. Springer International Publishing, Cham, 2021.

[40] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and*

Learning Systems, 30(11):3212–3232, 2019.

[41] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[42] Andrea Zingoni, Marco Diani, and Giovanni Corsini. Tutorial: Dealing with rotation matrices and translation vectors in image-based applications: A tutorial. *IEEE Aerospace and Electronic Systems Magazine*, 34(2):38–53, 2019.

A. Appendix

```
#Imports
#3rd party
import threading
import os
import sys

#local import
sys.path.insert(0, os.path.abspath("../")) #include path to mi50
    folde
sys.path.insert(0, os.path.abspath("../mi50")) #include path to
    ximea folder
from mi50 import acquisition

if __name__ == '__main__':

  #init variables (exposure time, lens aperture and frame rate)
  enable_aeag, exposure_val, lens_aperture_val, frame_rate, trigger
    , tgr_timeout, imgdataformat, auto_wb, n_total, flag =
    acquisition.set_parameters();

  #creation of new sequence of images
  acquisition.new_sequence(enable_aeag, exposure_val,
    lens_aperture_val, frame_rate, trigger, imgdataformat, auto_wb,
    n_total);

  #start acquisition
  for data,data_dict,n_im in acquisition.start(enable_aeag,
    exposure_val, lens_aperture_val, frame_rate, trigger,
    tgr_timeout, imgdataformat, auto_wb, n_total, flag):

```

```
    threading.Thread(target=acquisition.save_tiff,args=(data, data_dict, n_im)).start() #new thread created
```

Listing A.1: Image acquisition

```
import subprocess
import re
import numpy as np
import time
import os
from datetime import datetime

current_datetime = datetime.now()
target_word = "fused"
pattern_1 = re.compile(rf'' + re.escape(target_word) + ' ([^\s]+) hAcc=([^\s]+) et=[^\s]+ alt=([^\s]+)')
command = "adb shell dumpsys location | grep last"
output = os.popen(command).read()
match_1 = pattern_1.search(output)
extracted_data = match_1.group(1).split(",")


def get_orientation_data():
    adb_command = "adb shell dumpsys sensorservice | grep -A 10 \"ORIENTATION: last 10 events\""
    result = np.zeros((100,3))
    count = 0
    for i in range(10):
        output = subprocess.check_output(adb_command, shell=True, text=True)
        for line in output.split("\n"):
            pattern = re.compile(rf'([0-9]+) \(ts=[0-9.]+, wall=[0-9:.]+\) ([0-9\-.,\s]+),')
            match = pattern.search(line)
            if match:
                extracted_data = np.array(match.group(2).split(', '))
                y = extracted_data.astype(np.float)
                result[count] = y
                count += 1
        time.sleep(0.2)
    means = np.mean(result, axis = 0)
    print(f"{current_datetime}, {means}")
get_orientation_data()
print(f"{current_datetime}, Lat: {extracted_data[0]}")
print(f"{current_datetime}, Long: {extracted_data[1]}")
print(f"{current_datetime}, hAcc: {match_1.group(2)}")
print(f"{current_datetime}, Alt: {float(match_1.group(3))-h_1} ")
```

Listing A.2: Sensor reading

```
import numpy as np
import utm

def normalize(v):
    """
    Normalizes a vector to have a magnitude of 1.

    Parameters:
    - v: The vector to be normalized.

    Returns:
    - A normalized vector with the same direction but a magnitude
   of 1.
    """
    l = 1.0 / np.sqrt(np.dot(v, v))
    return np.array(v * l)

def calcRotationMatrix(theta, phi, psi):
    """
    Calculates a 3D rotation matrix from Euler angles.

    Parameters:
    - theta: Rotation around the X-axis (pitch) in radians.
    - phi: Rotation around the Y-axis (yaw) in radians.
    - psi: Rotation around the Z-axis (roll) in radians.

    Returns:
    - A 3x3 rotation matrix.
    """
    Rx = np.array([[1, 0, 0], [0, np.cos(theta), -np.sin(theta)],
   [0, np.sin(theta), np.cos(theta)]])
```

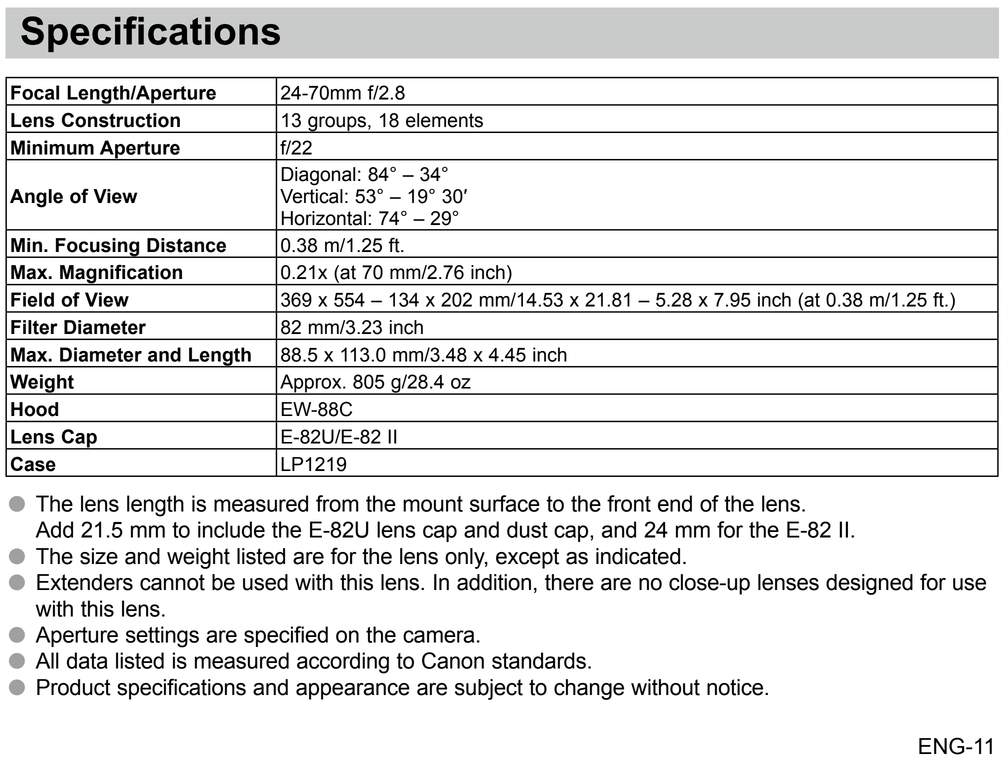
```
    Ry = np.array([[np.cos(phi), 0, np.sin(phi)], [0, 1, 0], [-np.sin(phi), 0, np.cos(phi)]])
    Rz = np.array([[np.cos(psi), -np.sin(psi), 0], [np.sin(psi), np.cos(psi), 0], [0, 0, 1]])
    return np.matmul(Rz, np.matmul(Ry, Rx))  # Order: ZYX (roll, pitch, yaw)

def calcCameraRay(u, v, fov, w, h, Rt):
    """
    Generates a ray from the camera to a point in the world.

    Parameters:
    - u, v: Pixel coordinates in the image.
    - fov: Field of view of the camera in radians.
    - w, h: Width and height of the image in pixels.
    - Rt: Camera's rotation and translation matrix.

    Returns:
    - A tuple containing the normalized ray direction and the ray origin in world coordinates.
    """
    aspect_ratio = w / h
    t_fov = np.tan(fov * 0.5)
    Px = (2 * ((u + 0.5) / w) - 1) * t_fov * aspect_ratio
    Py = (1 - 2 * ((v + 0.5) / h)) * t_fov

    rO = np.matmul(Rt, np.array([0, 0, 0, 1]))  # Ray origin: camera's position in world coordinates
    rP = np.matmul(Rt, np.array([Px, Py, -1.0, 1.0]))  # Ray through pixel in world coordinates

    return normalize(rP[:3] - rO[:3]), rO[:3]

def intersectPlane(n, p0, r0, rD):
    """
    Finds the intersection of a ray and a plane, if it exists.

    Parameters:
    - n: The normal vector of the plane.
    - p0: A point on the plane (used to define the plane).
    - r0: The origin of the ray.
    - rD: The direction of the ray, normalized.

    Returns:
    - A tuple of a boolean indicating if there is an intersection,
   and the distance along the ray to the intersection.
    """
    denom = np.dot(n, rD)
    if np.abs(denom) > 1e-6:  # Avoid division by zero
        delta_p = p0 - r0
        d = np.dot(delta_p, n) / denom
        return (d >= 0), d  # Intersection exists if d is positive
    return False, 0
```

Listing A.3: Existing script for raycastin g from another DLR project

```
import numpy as np
from raycasting_utils import calcCameraRay, intersectPlane,
   calcRotationMatrix
import utm
from ultralytics import YOLO  # Make sure to install the
   Ultralytics YOLO package
import folium

def main():
    # Define camera setup parameters for georeferencing
    w, h = 7920, 6004  # Image dimensions (width, height)
    fov = np.radians(74.35)  # Field of view in radians

    theta = np.radians(39.0080)  # Camera pitch angle in radians
    phi = np.radians(1.3640)  # Camera yaw angle in radians
    psi = np.radians(0.0549)  # Camera roll angle in radians

    tz = 13.21  # Camera height above the water in meters

    # Convert the camera's geographic location to UTM coordinates
    latitude, longitude = 53.522028, 8.583522

    utm_coords = utm.from_latlon(latitude, longitude)
    print(utm_coords)
    tx, ty = utm_coords[0], utm_coords[1]  # UTM coordinates of the
```

```
 camera

    # Calculate the camera's rotation and translation matrix
    R = calcRotationMatrix(theta, phi, psi)
    Rt = np.vstack([np.c_[R, [tx, ty, tz]], np.array([0, 0, 0, 1])])

    # Load the YOLO model and perform object detection
    model = YOLO('yolov8x.pt')  # Make sure the YOLOv8x model weights are accessible
    result = model('/media/xavier/hd/Bild 1.tif', classes=[8], save=True)  # Analyze an image, focusing on class 8 (boats)

    boat_coordinates = []  # Store georeferenced locations of detected boats

    # Process detected boats
    for box in result[0].boxes.xywh:
        cx, cy, bw, bh = box.cpu().numpy()  # Center, width, and height of the bounding box
        px = int(cx)  # Bottom center of the bounding box (x coordinate)
        py = int(cy + bh / 2)  # Bottom center of the bounding box (y coordinate)

        # Calculate the geographic coordinates of the detected boat
        rayD, rayO = calcCameraRay(px, py, fov, w, h, Rt)
        retval, d = intersectPlane(np.array([0, 0, 1]), np.array([1, 1, 0]), rayO, rayD)
        if retval:
            Pworld = rayO + rayD * d
            latlon = utm.to_latlon(Pworld[0], Pworld[1], utm_coords[2], utm_coords[3])
            boat_coordinates.append(latlon)
            boat_map = folium.Map(location=[boat_coordinates[0][0], boat_coordinates[0][1]], zoom_start=5)

    # Print the latitude and longitude of each detected boat
    for coord in boat_coordinates:
        print(f"Latitude: {coord[0]}, Longitude: {coord[1]}")
        folium.Marker([coord[0], coord[1]]).add_to(boat_map)
    boat_map.save("boat_map.html")

import webbrowser
webbrowser.open("boat_map.html")

if __name__ == "__main__":
    main()
```

Listing A.4: An example of the georeferencing of a image



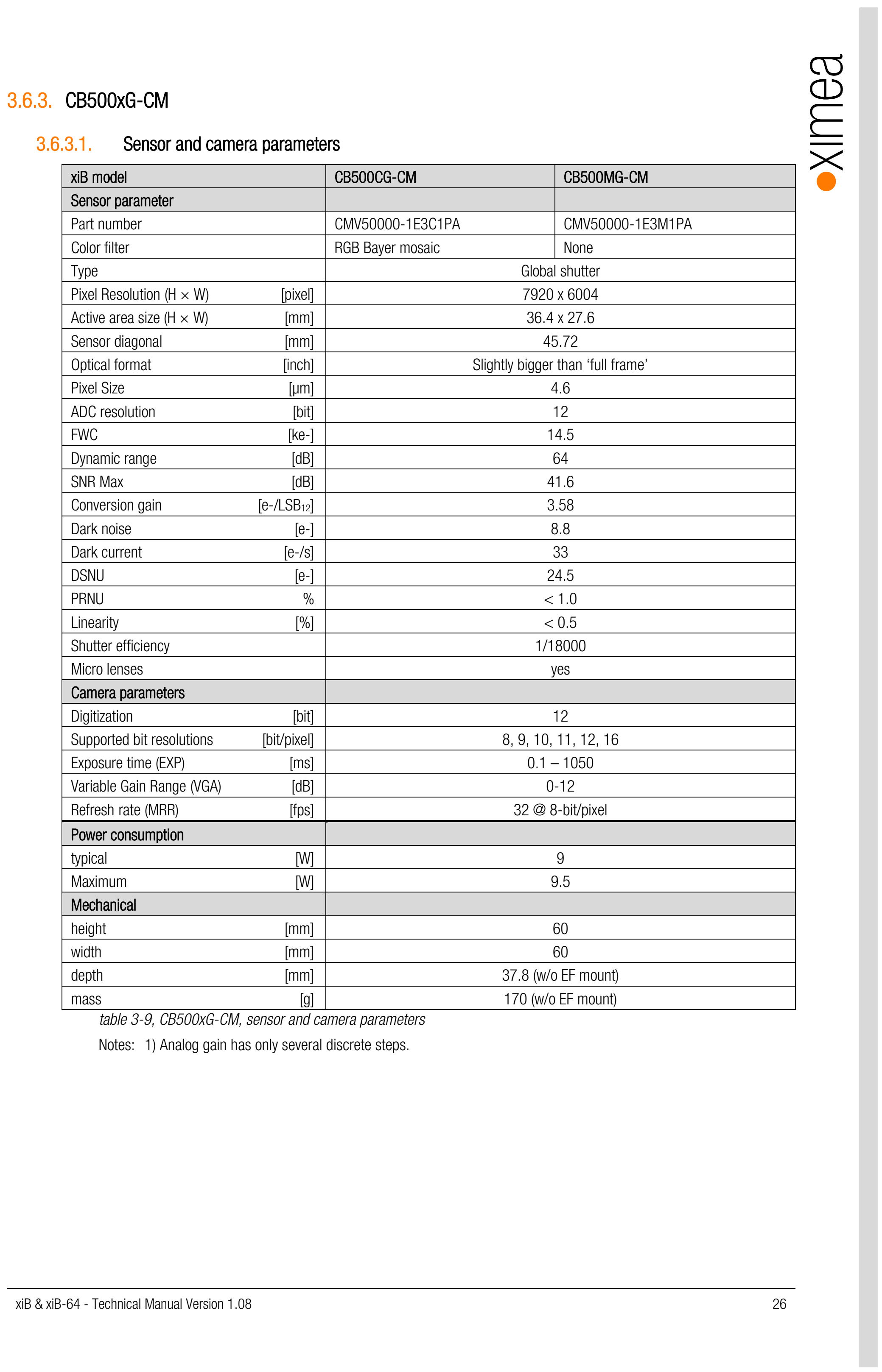
Specifications

Focal Length/Aperture	24-70mm f/2.8
Lens Construction	13 groups, 18 elements
Minimum Aperture	f/22
Angle of View	Diagonal: 84° – 34° Vertical: 53° – 19° 30′ Horizontal: 74° – 29°
Min. Focusing Distance	0.38 m/1.25 ft.
Max. Magnification	0.21x (at 70 mm/2.76 inch)
Field of View	369 x 554 – 134 x 202 mm/14.53 x 21.81 – 5.28 x 7.95 inch (at 0.38 m/1.25 ft.)
Filter Diameter	82 mm/3.23 inch
Max. Diameter and Length	88.5 x 113.0 mm/3.48 x 4.45 inch
Weight	Approx. 805 g/28.4 oz
Hood	EW-88C
Lens Cap	E-82U/E-82 II
Case	LP1219

- The lens length is measured from the mount surface to the front end of the lens. Add 21.5 mm to include the E-82U lens cap and dust cap, and 24 mm for the E-82 II.
- The size and weight listed are for the lens only, except as indicated.
- Extenders cannot be used with this lens. In addition, there are no close-up lenses designed for use with this lens.
- Aperture settings are specified on the camera.
- All data listed is measured according to Canon standards.
- Product specifications and appearance are subject to change without notice.

ENG-11

Figure A.1.: Data sheet for the lens



3.6.3. CB500xG-CM

3.6.3.1. Sensor and camera parameters

xiB model		CB500CG-CM	CB500MG-CM
Sensor parameter			
Part number		CMV50000-1E3C1PA	CMV50000-1E3M1PA
Color filter		RGB Bayer mosaic	None
Type		Global shutter	
Pixel Resolution (H × W)	[pixel]	7920 x 6004	
Active area size (H × W)	[mm]	36.4 x 27.6	
Sensor diagonal	[mm]	45.72	
Optical format	[inch]	Slightly bigger than ‘full frame’	
Pixel Size	[µm]	4.6	
ADC resolution	[bit]	12	
FWC	[ke-]	14.5	
Dynamic range	[dB]	64	
SNR Max	[dB]	41.6	
Conversion gain	[e-/LSB$_{12}$]	3.58	
Dark noise	[e-]	8.8	
Dark current	[e-/s]	33	
DSNU	[e-]	24.5	
PRNU	%	< 1.0	
Linearity	[%]	< 0.5	
Shutter efficiency		1/18000	
Micro lenses		yes	
Camera parameters			
Digitization	[bit]	12	
Supported bit resolutions	[bit/pixel]	8, 9, 10, 11, 12, 16	
Exposure time (EXP)	[ms]	0.1 – 1050	
Variable Gain Range (VGA)	[dB]	0-12	
Refresh rate (MRR)	[fps]	32 @ 8-bit/pixel	
Power consumption			
typical	[W]	9	
Maximum	[W]	9.5	
Mechanical			
height	[mm]	60	
width	[mm]	60	
depth	[mm]	37.8 (w/o EF mount)	
mass	[g]	170 (w/o EF mount)	

table 3-9, CB500xG-CM, sensor and camera parameters

Notes: 1) Analog gain has only several discrete steps.

xiB & xiB-64 - Technical Manual Version 1.08 26

Binning/skipping	Output resolution	Bit/px	fps	Readout time [ms]
1x1/1x1	7920 × 6004	8	30.9	32.36
1x1/1x1	7920 × 6004	10	28.8	34.70
1x1/1x1	7920 × 6004	12	24.1	41.62
1x1/1x2	7920 × 3002	8	61.4	16.29
1x1/1x2	7920 × 3002	10	57.1	17.48
1x1/1x2	7920 × 3002	12	47.9	20.94
1x1/2x2	3960 × 3000	8	61.4	16.29
1x1/2x2	3960 × 3000	10	61.4	16.29
1x1/2x2	3960 × 3000	12	61.4	16.29
2x2/1x1	3960 × 3000	8	30.8	32.43
2x2/1x1	3960 × 3000	10	30.8	32.43
2x2/1x1	3960 × 3000	12	30.8	32.43

table 3-10, CB500xG-CM, standard readout modes

3.6.3.2. Quantum efficiency curves [%]

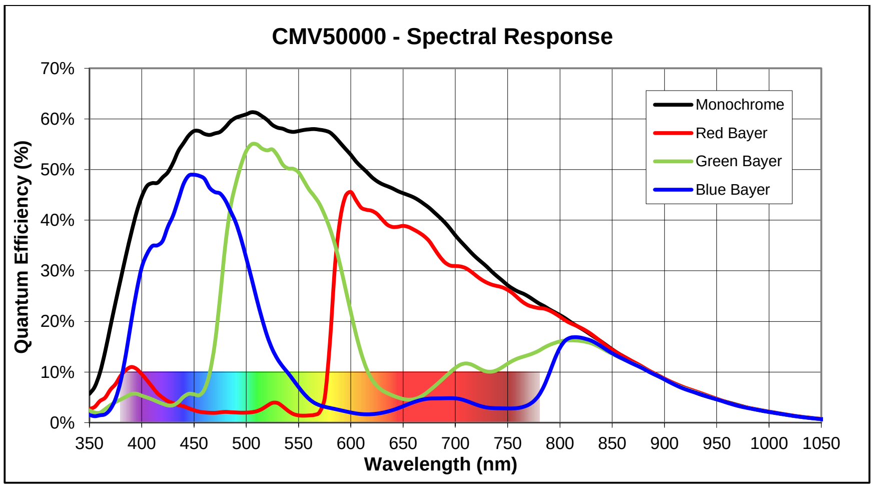


figure 3-10 CMV50000 Quantum Efficiency ©CMOSIS

3.6.3.3. Dimensional drawings CB500xG-CM (with and without EF mount)

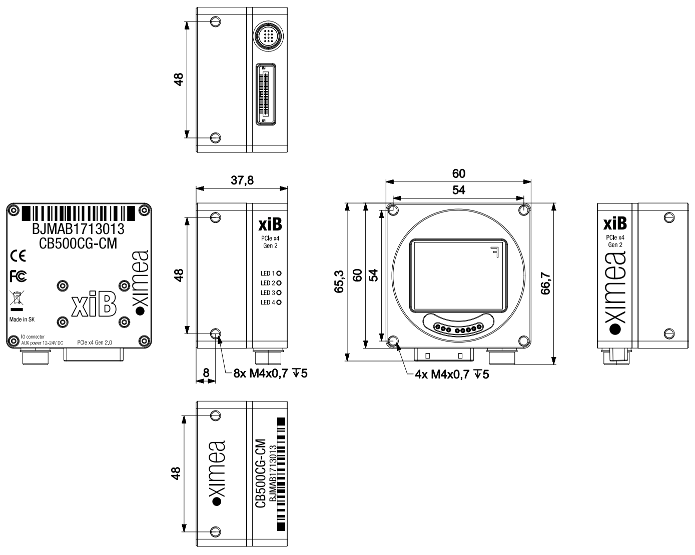


figure 3-11, dimensional drawing CB500xG-CM w/o EF-mount adapter

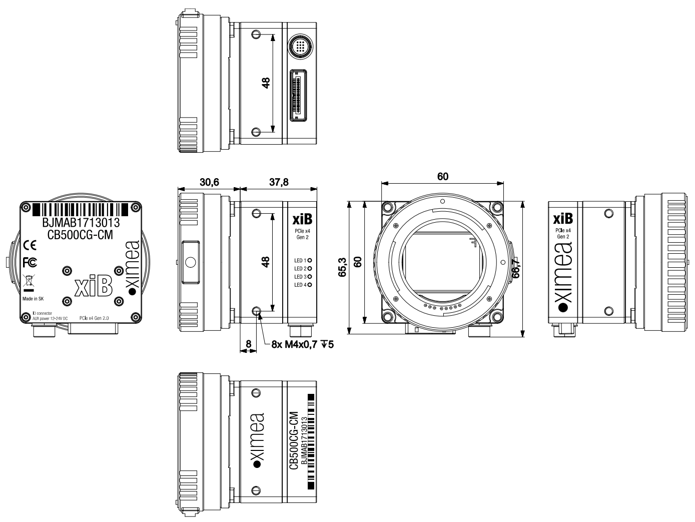


figure 3-12, dimensional drawing CB500xG-CM with EF-mount adapter

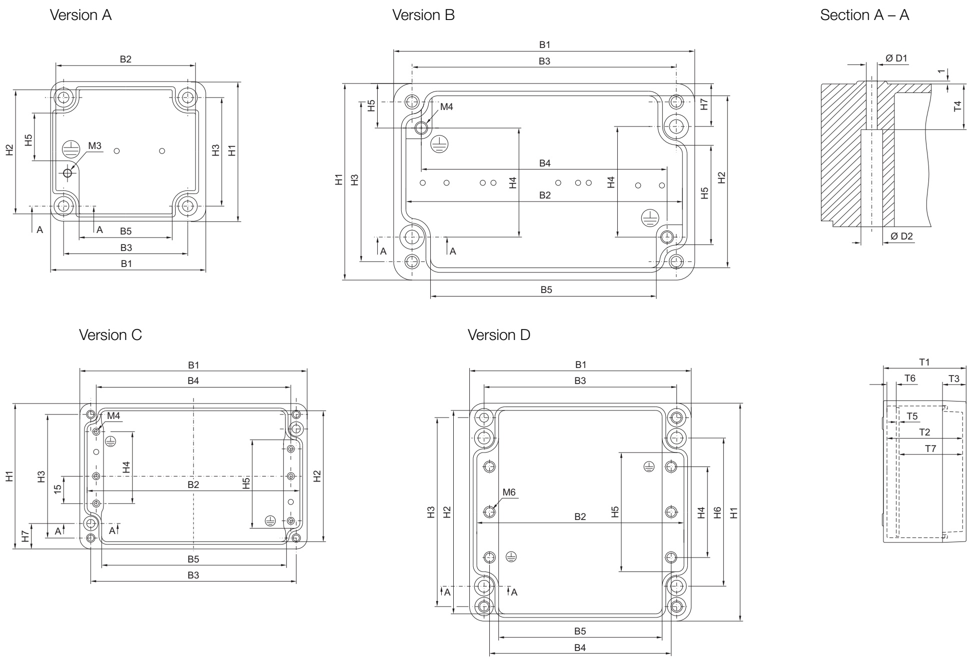
xiB & xiB-64 - Technical Manual Version 1.08 28

Figure A.2.: Data sheet for the camera

Enclosures

Small enclosures

Cast aluminium enclosures GA



Note:
- For installations manufactured by the customer, the width and height dimensions of the mounting plate must not be exceeded.
- For enclosures where no mounting plate is available, the following dimensions shall apply analogously:

Model No. GA	Width mm	Height mm
9101.210	48	54
9102.210	88	54
9104.210	64	69
9106.210	164	69
9107.210	239	69
9111.210	347	107

Model No. GA	Version	Width dimensions mm					Height dimensions mm							Depth dimensions mm							Diameter mm	
		B1	B2	B3	B4	B5	H1	H2	H3	H4	H5	H6	H7	T1	T2	T3	T4	T5	T6	T7	D1	D2
9101.210	B	58	50	46	40	34	64	56	52	33	32	–	14	34	29	9	8	–	–	–	4.5	8
9102.210	B	98	90	86	81	74	64	57	52	33	32	–	14	35	29	10	8	–	–	–	4.5	8
9104.210	C	75	66	63	56	52	80	71	68	39	48	–	14	57	50	15	9.5	–	–	–	4.5	8
9105.210	C	125	116	113	106	99	80	71	68	39	48	–	14	57	50	15	10	1.5	6	42.5	4.5	8
9106.210	C	175	166	163	156	152	80	71	68	39	48	–	14	57	50	15	8	1.5	6	42.5	4.5	7
9107.210	C	250	241	238	231	226	80	71	68	39	48	–	14	57	50	15	9.5	1.5	6	42.5	4.5	7.5
9108.210	D	122	112	106	95	90	120	111	104	52	64	82	–	80	72	20	15.5	1.5	8	62.5	6.5	10.5
9110.210	D	220	211	204	195	183	120	111	104	50	64	82	–	91	82	30	15	1.5	9	71.5	6.7	11
9111.210	D	360	349	344	333	322	120	111	104	48	62	82	–	82	72	20	9	2	8.5	61.5	6.5	10.8
9112.210	D	160	151	140	132	120	160	151	140	76	89	110	–	91	82	20	20	2	8.5	71.5	7	12
9113.210	D	260	251	240	230	220	160	151	140	76	90	110	–	91	82	20	19	1.5	8.5	72	7	13
9114.210	D	360	350	340	330	316	160	151	140	76	89	110	–	91	82	20	19	2	9	71	7	13.5
9116.210	D	202	190	180	170	159	232	221	210	144	159	180	–	111	102	20	21	2	9	91	6	13
9117.210	D	280	271	260	250	239	232	221	210	144	159	180	–	111	102	20	21	2	9	91	6	13
9118.210	D	334	321	310	300	289	233	223	210	144	160	180	–	111	102	20	25	2	9	91	6.4	13.5
9119.210	D	330	321	310	300	290	230	221	210	144	160	180	–	181	170	20	9	2	9	159	7.5	11

Technical details/Enclosures/02.2014

dri1308002en.fm – 1-102 – 1 of 1

Figure A.3.: Data sheet for the housing