Maritime monitoring system against illegal fishing

By Onne Iping (s3041050)
Creative Technology
Supervisor: dr. Andreas Kamilaris
Critical observer: dr.ir. Alex Chiumento

4th of July, 2025

UNIVERSITY OF TWENTE.

Abstract

Illegal fishing is a critical global issue, threatening marine biodiversity and disrupting ecosystems. Additionally, it creates unfair competition for legal fishing companies. In order to effectively combat this, continuous improvements in maritime monitoring are essential. This project presents a concept for a system that combines satellite imagery and Automatic Identification System (AIS) data, which uses deep learning to detect and evaluate potentially unauthorized vessels. The system integrates a YOLOv11-Oriented Bounding Box (OBB) model for ship detection on both optical and SAR satellite imagery. Detected ships are geolocated using metadata from GeoTIFF images and then matched with AIS broadcasting vessels based on positional and time data. The plausibility of each match is assessed using average speed thresholds, and the length and classification of each detected ship are estimated. An interactive visualization interface enables users to explore detections and their AIS matches through an online web tool. Deep learning models were trained for both optical and SAR imagery, achieving high performance with F1-scores of 94.8% for optical and 99.1% for SAR. However, the integrated system incorporating detection, AIS matching, and visualization was only implemented and tested on optical imagery. Evaluation of the integrated system demonstrated effective matching of detected vessels with AIS, although some errors occurred due to reliance on a limited amount of parameters, like position and average speed. A user study of the visualization confirmed its clarity, intuitiveness, and practical relevance, although no domain experts participated in the study. While further improvements are needed for real-world application, the system shows a strong foundation for enhancing maritime surveillance.

Table of contents

1. Introduction

Illegal fishing drives global overfishing and harms marine environments. It causes disruptions in food chains, which imbalances marine ecosystems, and leads to a decrease in (endangered) fish populations. Additionally, it can damage vulnerable habitats like coral reefs [16]. However, it does not only hurt the natural environment. Illegal fishing also hurts legal fishing companies. By overfishing and depleting fish stocks, illegal operators create unfair competition, driving down prices and making it harder for legitimate companies to survive. In some cases, this pressure forces legal fishers to lower their prices or even engage in illegal fishing themselves to remain financially viable [16]. Next to that, illegal fishing also threatens food security and is often linked to organized crime and human rights abuses [15]. However, the improvement of maritime monitoring is not only important in order to combat illegal fishing. Unauthorized maritime migration also presents a significant challenge. For example, at the start of 2024, Cyprus saw an increase in unauthorized migration over sea, mostly from Lebanon, with many individuals fleeing conflict and poor economic conditions [38]. These dangerous journeys place migrants at great risk, highlighting the urgent need for improved maritime monitoring. Enhanced surveillance can help protect human lives by enabling faster detection and response, thereby improving rescue operations and overall safety at sea.

All of these facts highlight the importance of improving maritime monitoring and developing new methods to achieve this. With modern technology, there are more opportunities than ever to address this problem. Systems like the Automatic Identification System (AIS) allow for global vessel tracking, while satellite imagery provides visibility even in the most remote areas. As of today, many remote sensing technologies are available for satellite imagery generation, notably Synthetic Aperture Radar (SAR) and optical systems. SAR offers deployability during both day and night and in all weather conditions, while optical imagery provides very high spatial resolution [21]. Additionally, the field of artificial intelligence presents a significant opportunity to interpret this data, which can help improve maritime monitoring as well. By applying artificial intelligence on satellite imagery, ships can be detected in specific regions. This can be combined with AIS data, a communication system where ships share their position among other information. Vessels are required to keep AIS active, and disabling it could indicate involvement in illegal activities [6]. All of these technologies together make it possible to identify illegal fishing activities or other unauthorized vessels in real-time.

Combining these facts into a problem statement gives the main research question for this project, which is as follows.

- *How can an effective maritime monitoring system be developed using earth observation methods, Automatic Identification System data and deep learning technologies?*

In order to make the research more clear, three sub research questions have been formulated to support the main question.

- *What remote sensing technology is most optimal for ship detection methods?*
- *Which deep learning model is best for ship detection on satellite imagery?*
- *How can Automatic Identification System data be used in order to identify unauthorized vessels?*

The goal of this project is thus to develop a system which uses earth observation techniques, AIS data and deep learning technologies, in order to detect unauthorized activity at sea. In order to maximize the possible impact, the system must be cost-effective and

require minimal computational resources, making it accessible even to countries with limited technological infrastructure and financial resources. By ensuring affordability and ease of use, this solution can help strengthen global efforts to combat illegal fishing, particularly in regions where enforcement capabilities are currently lacking.

2. Background information

In this chapter, the state-of-the-art of different aspects related to the project will be discussed. These are the Automatic Identification System, remote sensing technologies, and deep learning models. These components will also be discussed in relation to ship detection. The chapter concludes with a summary and discussion of the findings.

2.1 Automatic Identification System

The Automatic Identification System (AIS) was originally created to improve safety for commercial vessels. It allows them to be aware of nearby ships' positions, reducing the risk of collisions, especially in poor weather conditions [6]. This is because AIS continuously transmits data like speed, course and current position to all nearby vessels equipped with the system. AIS transmits this data using VHF radio. If a ship is relatively close to the shore (not more than around 40 nautical miles), this data is gathered and processed by coastal receivers. If a ship is far out on the ocean, the communication goes via satellites [6].
Not all data is sent continuously; a distinction is made between static and dynamic information [6, 7]. Static information is transmitted every 6 minutes or when requested. It includes details such as the vessel's length, type, and identification numbers [6]. Dynamic information is sent more frequently, depending on the ship's speed and course [6]. This includes the vessel's estimated position, the time of recording that position, and its course [6]. For vessels with Class B AIS, this data is transmitted every 30 to 180 seconds, while for Class A AIS this is every three minutes while at anchor and every two to 10 seconds while in transit [7]. Whether a ship needs Class A or B, depends on if it's a SOLAS vessel [7]. A SOLAS (Safety Of Life At Sea) vessel refers to a ship that is subject to the SOLAS regulation, which states that all passenger vessels (at any size) as well as all vessels of 300 GT (gross tonnage) or larger need to have Class A AIS [6]. Other ships, for example pleasure crafts only need to have Class B AIS onboard. IMO (International Maritime Organization) guidelines state that AIS should always be active on vessels, both when at anchor and when in movement [6]. There are only a limited number of circumstances in which it is legal to deactivate AIS. Therefore, turning it off could indicate potential involvement in illegal activities [6].

2.1.1 Datasets

There are several ways to access AIS data for this study. One option is to use large companies, such as Spire [8], which offer high-quality AIS data on request, both live and historical. While this would likely be the most efficient method for obtaining data, it is beyond the scope of this project and not realistic due to financial constraints.

Another option is to use public datasets. One example is the National Oceanic and Atmospheric Administration (NOAA), who make historic AIS data available for certain time periods for waters around the USA [9]. This includes positions of vessels (latitude and longitude) and the time of recording, speed, type of AIS (A or B), identification number, and more. Additionally, a dataset on AIS data around the Piraeus port in Greece was found [10]. This dataset contains more than 244 million records of AIS, covering the activity over a time span of 2.5 years, from May 2017 to December 2019. The data consists of positional data, and information like vessel identification and speed.

2.2 Remote sensing technologies

There are three main remote sensing technologies available, which are optical imaging, SAR (Synthetic Aperture Radar) and LiDAR (Light Detection And Ranging). Optical has a large coverage of the earth, and has been around for a long time which means it has been well optimized and tested [21]. Additionally, it often has a relatively high spatial resolution. However, it can only be used during the day when the weather is clear, since it can't penetrate through clouds [22].

SAR on the other hand works in all weather conditions and is not dependent on sunlight. It also has a very large coverage [21]. Additionally, it is very effective in identifying changes in materials. One disadvantage is the fact that the imagery often has a lot of noise. Additionally, the spatial resolution is often lower than optical imagery [22].

Finally, LiDAR (Light Detection And Ranging) is another technology named in the literature [21]. It is mostly used to provide 3D elevation data, and has a very high spatial resolution, which can come down to a centimeter. It can also be used during both day and night. However, the coverage is low, it needs a lot of data storage and the costs are very high [22].

Concluding, LiDAR does not seem like a viable option for ship detection due to its limited coverage, which is a major drawback for maritime monitoring, where large-area surveillance is essential. Additionally, the high cost further reduces its practicality. SAR on the other hand seems like a better option, since it's effective for detecting changes in structure/textures. For example, a ship on an open sea would be very easy to detect. Being able to operate day and night in all weather conditions is also a major advantage for maritime monitoring, as continuous surveillance increases the likelihood of detecting illegal fishing and other unauthorized activities. However, the noise in SAR imagery may require extensive preprocessing to ensure its suitability for deep learning models. Optical imaging avoids this issue, with noise-free images often available with high spatial resolution. However, its effectiveness is limited to daytime and clear weather conditions, making it less reliable for continuous maritime monitoring.

To determine which technique is most optimal thus depends on what is seen as most important for the project. If all-weather, round-the-clock monitoring is essential, SAR is the better choice. However, if occasional monitoring with minimal preprocessing is preferred, optical imagery is more suitable.

2.2.1 Combining SAR and optical imagery

Occasionally, SAR and optical imagery are both used together for ship detection. Not a lot of research has been done about these so-called USD (Universal Ship Detection) models, mostly because of the complexity involved with this [11]. Most ship detection models are optimized for a specific domain (SAR, optical). When using it on a different domain, the performance might decrease significantly. So, it is important to work with domain-invariant object features in order to reduce the effects of domain shift. However, distinguishing between domain-specific and domain-invariant features is complex [11].
Zhang et al [11] present a solution to these obstacles by implementing an MDCM (Multilevel Domain Classification Network) together with a DCM (Domain-Centric Cut-Paste Module). The DCM takes the ships from images from one domain, and puts them on a background from the other domain [11]. This way, the model is forced to generalize better. Additionally, the MDCM ensures that the model does not get optimized specifically for one domain.

The performance metrics show this approach is effective, with an average precision of 93.3% across 6 different datasets containing imagery of different domains. This is the highest score compared to other models like RetinaNet and Yolov3 on the same datasets [11]. In conclusion, while combining SAR and optical imagery for ship detection is possible, it requires careful handling of domain shifts and the use of techniques that ensure the model generalizes well across different domains.

2.2.2 Datasets

In order to train and test the deep learning model, it is necessary to either create or acquire a dataset for this purpose. For this dataset, it is important that it contains images from satellites of marine environments, and with a substantial amount showing ships as well. In order to save time, the priority is to find an existing dataset from previous research.

One example that was found was the MASATI dataset [19]. This dataset contains optical satellite imagery from coastal areas as well as open sea areas. It also consists of images showing either one or multiple ships, or none at all. Some examples can be seen in the figure below.

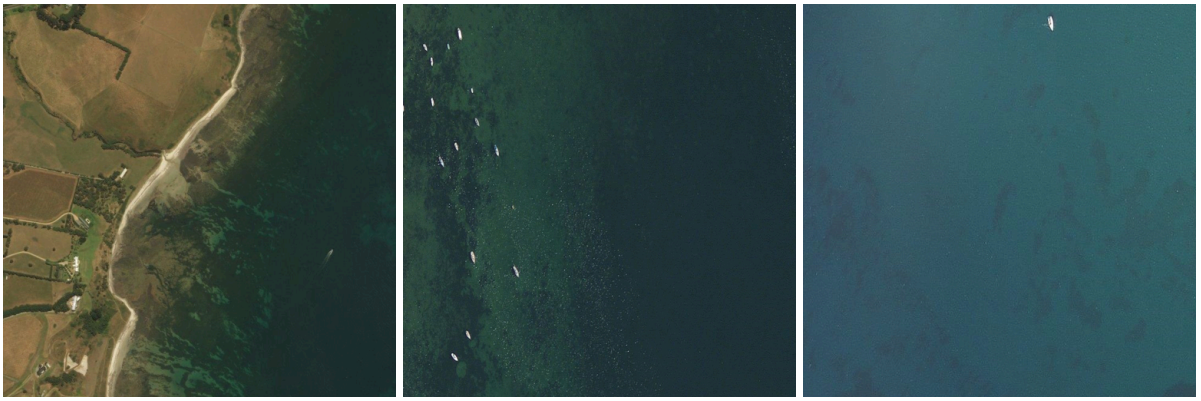


Figure 1: Example pictures from MASATI dataset [19]

In total, there are around 7000 images in the dataset, made of all kinds of geographical areas, including the Mediterranean Sea and the Pacific and Atlantic oceans, also including several different weather conditions. It also includes labeling with a box in order to determine the location of a ship.

Additionally, SAR datasets were also found for this research. One example is the HRSID dataset [20]. It contains around 5000 images, both with or without one or more ships. All the images have a spatial resolution of either 0.5, 1 or 3 meters. The images contain visuals of coastal areas, harbors and open sea regions. Additionally, data on box labels is also available for every image in this dataset. Some examples are shown below in figure 2.

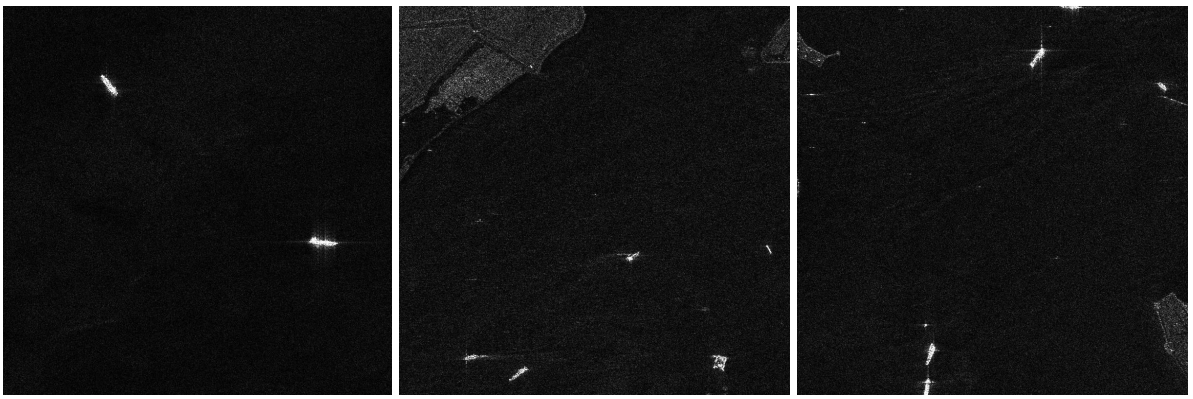


Figure 2: Example pictures from HRSID dataset [20]

2.3 Deep learning models

Ship detection using artificial intelligence has been done often in previous research. However, many different approaches have been used for this. In order to get a good overview of the previous research, a literature review has been done on this.

Before discussing specific models, it is important to make a distinction between different detector models. Firstly there are two stage detectors. As indicated by the name, these detectors operate in two phases. In the first stage regions within the image are identified that possibly contain the object that needs to be detected [25]. These regions are called region proposals. In the next step, features from these regions are extracted and are analyzed by a classification network, which then determines the category the object should be classified to, as well as correcting the position of the region [25].

Secondly, there are single stage detectors. Instead of having multiple steps, these detectors process images in one go. This is done by dividing images into a grid, with each cell in the grid detecting objects at its center [25]. Each grid then places some bounding boxes of different sizes and assigns a score to each of these boxes, indicating the probability of a certain object being present within the box. Finally, overlapping detections are removed and only the boxes with the highest scores remain [25].

In order to research what specific deep learning models are most commonly used for ship detection, a literature review has been done. A total of 12 previous researches have been analysed with the use of a literature matrix, which has been added as Appendix A at the end of this report. From this research, it was concluded that YOLO models are one of the most commonly used single stage detectors for ship detection. For example, Bakirci [1] discusses a method using the YOLOv9 model. Here, YOLO models are described as faster and more accurate than two-stage detectors. Bakirci also states that YOLOv9 is very suitable for ship detection, for different reasons [1]. Firstly, the structure of the model allows for detecting small and complex features, which makes detecting ships easier. For example, the GELAN (The Generalized Efficient Layer Aggregation Network) module allows for more efficient extraction of features, which improves detection of small ships, ships on complex backgrounds, or many ships close together [1]. Additionally, the PGI (Programmable Gradient Information) module improves the learning process of the model, which leads to high accuracy even with limitations on computational power. All of this together makes YOLOv9, or any newer version of YOLO, a good fit for ship detection [1].

An example of a two stage detector which was often named in the researched papers is a CNN (Convolutional Neural Network). For example, Yang et al [3] present a method

using a region based CNN (R-CNN). One of the reasons given why this method works well for ship detection, is the fact that a Balanced Feature Pyramid (BFP) is present. This module improves ship detection of different sizes by balancing details in the image. Some parts of the image have clear details but little meaning, while others have useful information but are blurry. The BFP fixes this, making it easier to detect both big and small ships accurately [3].

2.4 Conclusion

In this background research, the goal was to gain insights in the different technologies like AIS, remote sensing and deep learning and their relation to ship detection. Firstly, it was found that AIS is a tracking system used to share data like location and speed in order to enhance marine safety. Turning it off is illegal and often indicates participating in illegal activities. Additionally, it was concluded that there are three remote sensing technologies most often used, which are optical sensing, SAR and LiDAR. For ship detection, SAR and optical sensing are mostly used. Optical sensing has the advantage of having a relatively high spatial resolution, but is dependent on sunlight and clear weather. In contrast, SAR can be used in all weather and lighting conditions, but often has noisy imagery. Since they both have their advantages and disadvantages, it depends on the preferences which technology is the best. Finally, it was found that there are two types of detector models used for object detectors, which are two-stage and one-stage detectors. A CNN is an example of a two stage detector, which was named often in the researched papers. A one-stage detector that was named very often is YOLO, from which many different versions were discussed in the reviewed literature.

This information is relevant for the continuation of this research, since it provides valuable information that can be considered when making important design choices on the system. For example, the available remote sensing technologies to choose from has been brought back to two. Additionally, finding out the most common models used for ship detection also helps for the decision for a model.

Several of the reviewed studies have developed ship detection models using similar combinations of remote sensing data and deep learning techniques. For example, Bakirci [1] achieved an F1-score exceeding 90% on optical imagery, while Zhao et al. [18] reported comparable results using SAR data. These studies demonstrate the feasibility of achieving high detection performance with both types of satellite imagery. Their results therefore serve as practical baselines for this research, helping to establish realistic goals and provide points of comparison for the system’s performance.

3. Methodology

In this chapter, the different methods used in order to complete this project will be discussed. Firstly, the approach of making design decisions will be highlighted. Afterwards, the method of background research will be explained. Finally, data collection and evaluation will also be discussed.

3.1 MoSCoW method

Since most of the core concepts had already been formed before the project started, no formal brainstorming method was used during the start of the project. However, it was still essential to define clear requirements for the design of the system, in order to make the process of developing more clear. For this purpose, the MoSCoW method has been applied. This method is used to sort requirements into 4 different groups, which are ‘Must have’, ‘Should have’, ‘Could have’ and ‘Won’t have’. This approach ensures that the most critical features and functionalities are identified, while also making clear which potential additions could be made later on in the process.

3.2 Background research

For this project, the state-of-the-art of certain technologies had to be reviewed in order to gather the necessary information for making informed design decisions. The chosen method for this was literature research. The reason for this was because this method would provide the widest range of examples and views on the topic. For example, doing this via interviews would be more time-consuming, while the number of different perspectives gained with this method would be more limited. For this research however, it was especially important to collect as many examples as possible in order to investigate which approaches are used most often.

For this literature research, the goal was to find around 12 to 14 papers focused on approaches of deep learning based ship detection systems. Since the topic is fairly specific and the available options for remote sensing technologies and deep learning models used are limited, this number was considered to be an acceptable number in order to get a clear overview of the variety of methods most commonly used.

In order to analyse the papers found for the literature research, a literature matrix was used. The completed matrix can be found in Appendix A. This method helped to create a clear overview of the differences and similarities between the various studies, making it easier to draw conclusions from them.

3.3 Data collection

In order to be able to get the system working, it is important to collect data that can be used to test and train the system. There are a couple of sources used to collect the main data necessary for this project. Firstly, the Planet Labs [26] software is used to acquire optical satellite imagery from the preferred regions. Most (optical) imagery from here has a spatial resolution from 3 meters per pixel, which is relatively high. Additionally, a lot of historical data is available on this database, which allows for many available images for a possible dataset. Next to this, all SAR imagery used in this project was obtained through the Copernicus Browser [36], a free platform that provides access to Sentinel-1 data. It offers a large amount of data, including extensive historical coverage and varying spatial resolutions.

In order to create datasets with this, labeling had to be done manually. In order to do this, Label Studio [29] was used, an open-source software tool that enables efficient and intuitive data annotation. These datasets are necessary for training the deep learning models, as well as for evaluating the model's performance. Additionally, in order to test the system as a whole, AIS data from the same regions was necessary. For the project, it is preferable to be able to acquire this data based on personalized requests. This can be done via the databases of Marine Traffic [27], and this will thus be the main source for AIS data for this project. Basic subscriptions allow users to download AIS data from selected regions, but access to historical data is limited to individual vessels. If broader or more specific datasets are required, these can be obtained through custom data requests, which are available for an additional fee.

Finally, all implementation and experimentation will be carried out using Python, with PyCharm as the primary development environment.

3.4 Evaluation

Finally, the methods used to evaluate the performance of the system should be discussed. The deep learning model is the most significant part of the process and will be tested using several performance metrics. The metrics that will be used are precision, recall, F1-score and mean average precision (mAP). Precision represents the percentage of the model's positive predictions that are correct, while recall indicates the percentage of actual instances that the model successfully detects. The F1-score combines these two metrics into a single metric by calculating their harmonic mean, providing a balanced measure of the model's accuracy. Finally, the mAP summarizes the precision-recall curve across different confidence thresholds, giving an overall measure of detection performance. Additionally, next evaluating the deep learning model(s), the system as a whole will be evaluated when it is finished based on the requirements set up beforehand. The entire evaluation will be discussed in detail later on in Chapter 7 'Evaluation'.

4. Ideation

In this chapter, the requirements for the system will be set up. Additionally, a first concept of the system will be discussed and described, with argumentation for design choices.

4.1 Problem analysis

Illegal fishing poses a big threat to marine ecosystems, since it drives overfishing and often damages vulnerable environments [16]. It also endangers food security and there are often links to human rights abuses and organized crime [15]. Next to this, groups participating in illegal fishing are often an unfair competitor to legal fishing companies [16]. Improved maritime monitoring could help combat this issue effectively.

In addition to combating illegal fishing, improved maritime monitoring could also help track unauthorized maritime migration, which is a significant challenge for Cyprus. In early 2024, Cyprus saw a sharp increase in irregular sea arrivals, primarily from Lebanon, with many refugees risking dangerous journeys to reach the island [38]. These migrants often face life-threatening conditions at sea and limited access to asylum procedures once they arrive. Enhancing maritime surveillance could improve the safety of these vulnerable groups by enabling quicker rescues and better management of migration flows.

This is why the client for this project, the Pervasive Systems Group [28] based at the University of Twente, gave the assignment of developing a ship detection method using both satellite imagery and AIS data as a foundation. Additionally, it was also specified that this method should be able to operate using data from the geographical area of Cyprus.

Since this proposal was already specific enough, no further brainstorming was necessary during the ideation phase. This is why the decision was made to jump straight to formulating requirements.

4.2 Requirements

Looking at the requirements, some are more important than others. This is why the MoSCoW method was used, in order to get a clear overview of which requirements are prioritized and which are less important. Starting with the ‘Must have’ category, a key requirement is that the system should correctly identify at least 85% of the ships present in satellite imagery. Additionally, it must be able to work on optical imagery. This is because during the background research, this was found out to be the easiest technology to use for ship detection, since it does not need a lot of preprocessing. Additionally, it is also important that the system is able to determine the approximate location of the ship detected in an image. This is essential, since no action can be taken by authorities if no location is known at all. Finally, the system must be able to determine if a ship is unauthorized using AIS data. This is of significance, since without this no vessel would be flagged as unauthorized, and the system would be of no use.

After discussing the ‘Must have’ requirements, there is the ‘Should have’ category. These are still important aspects, but are not of such importance as the ‘Must have’ category. Firstly, the system should be effective on satellite imagery from the geographical area of Cyprus. It is important that the system is optimized for this region in the first place, since this is one of the requirements set by the client. In addition, the system should achieve an F1-score of at least 80%. While the ‘Must have’ requirement stated that the system should detect more than 85% of the cases, this alone does not guarantee that the model is

performing effectively. A high detection rate could still come with a large number of false positives. Because of this, an F1-score of 80% is proposed as a more balanced and meaningful performance target, as it considers both precision and recall. Next to this, the system should not be expensive to deploy. This is important, since a high cost could limit its accessibility, especially for smaller organizations or governments with limited budgets. When keeping the costs low, the solution can also be implemented in regions where illegal fishing is a problem, but resources are limited. Finally, the system should be made to be applicable with both SAR and optical imagery. This would be a big upgrade for the system, since SAR allows for monitoring both at night and in bad weather conditions [21]. This would mean the system would be deployable at any time.

Moving on to the ‘Could have’ requirements, which are aspects that are not essential for the initial version of the system, but could be added in case time allows this at the end of the process. The first of these is the fact that the geographical scope of the project could be broadened to different regions with other climates. If the system could be changed in a way that it works in a lot of different regions, this would increase the flexibility and real-world applicability of the system. This would make the system more valuable to a wider group of stakeholders. Another potential addition is the estimation of ship classification and speed, which would provide more detailed information about detected vessels. Another thing that could be added to the system is a clear online user interface. This would improve the usability of the system, and make it more accessible for authorities to use. Lastly, determining ship trajectories over time would add valuable information vessel movements, This could help identify movement patterns and detect unusual behavior.

Finally, there is also the ‘Won’t’ category, which includes features that have been excluded from the scope of this project. The only aspect sorted into this category, is the requirement that the system should be able to work real-time without manually uploading data. Real-time satellite imagery is not something that can be acquired with the limitations of this project, and this aspect has thus been excluded for now. However, it would be valuable in order to act fast against illegal fishing, and is thus something that could be studied in the future.

To give a clear overview of all the requirements and their priority, a diagram has been displayed in Figure 3.

Must	Should	Could	Won't
Be able to detect at least 85% of the ships in the provided satellite imagery.	Be able to operate with imagery from Cyprus and surroundings	Be made to work in different regions (with different climates)	Be able to work real-time (without uploading data manually)
Work on optical imagery.	Have an F1-score of 80% or higher.	Add estimation of classification and speed of ships	-
Be able to determine approximate location of detected ship	Not have a high operating cost.	Include an online user interface	-
Be able to determine whether ship is (potentially) illegal or not using approximated location and AIS data	Be able to work with both SAR and optical imagery	Determine ship trajectories	-

Figure 3: Table showing requirements sorted using the MoSCoW method.

4.3 Final concept

Concluding, a first concept of the eventual model can be formulated based on the problem analysis and the requirements set. The proposed system will use a combination of satellite imagery and AIS data. A deep learning model will be trained to detect ships on optical satellite imagery, as this imagery is most accessible for the initial version of the system. The system will also need to determine the approximate coordinates of the detected ships, using location data of the satellite imagery. Finally, the model will cross-check this location with the location transmitted by AIS signals from ships in the region. If no matching AIS signal is found for a ship at the same location and time as the satellite image recording, the vessel should be flagged as potentially unauthorized. This brings us to the next chapter, where this concept will be specified in more detail.

5. Specification

In this chapter, the concept proposed in Chapter 4 will be further explained. First, the functioning and goals of the deep learning model will be discussed. Afterwards, the role and requirements of the satellite imagery will be addressed. Following this, the integration of these components with AIS data will be highlighted. Finally, the chapter discusses the way of visualizing the findings of the monitoring system.

5.1 Deep learning model

The main purpose of the deep learning model within the system is to detect as many ships as possible on the provided satellite imagery. The first priority is to make it effective on optical imagery only, but ideally the system should also work on SAR imagery, given the complementary strengths of both technologies, as discussed in Chapter 2. However, as mentioned in this chapter as well, creating a model suitable for both of these technologies is challenging and time-consuming, which is why a separate model will be created for both of these technologies, given the time constraints of this project.

Additionally, the deep learning model(s) should be evaluated and optimized accordingly in order to mitigate the risk of not detecting a ship. Getting the precision and recall as high as possible is of importance. However, achieving a high recall is more critical than maximizing precision, as it is preferable to encounter some false positives rather than overlook potentially unauthorized vessels. Failing to detect a ship could mean missing illegal activity, while a false positive can be filtered out in later analysis.

Next to that, it is also important to gain information on the abilities and limitations of the model. This includes identifying scenarios in which the system may perform less accurately, as well as establishing basic thresholds, like the minimum ship size that can be detected. Information like this is crucial to better understand the operational boundaries of the monitoring system, and helps to set realistic expectations when applying the system in real-world scenarios.

5.2 Satellite Imagery

In order to properly train and test the deep learning model, both independently and as a part of the entire system, it is also important that satellite imagery is acquired. Since the scope of this project is on the maritime regions surrounding Cyprus, the first priority is to gain imagery from this area. However, in order to simplify the project, only imagery of open-sea imagery will be used. Images containing coasts or ports will not be used to validate the system with, since the complexity of terrain could complicate the training of the model. Excluding these regions only leaves a narrow coastal strip unmonitored, which is considered acceptable for this project. Additionally, including ports would likely lead to unreliable results, since the high density of docked vessels in a small area increases the risk of incorrect detections and inaccurate AIS matching.

Another important requirement is that the satellite imagery used for training the model should have the same spatial resolution as the imagery that will eventually be used in practice. A higher spatial resolution can also be used, as the data can always be downscaled afterwards. Using data of the same spatial resolution is important, as training the model on very high resolution data could decrease its performance on lower-resolution imagery, and vice versa.

The final requirement for the satellite imagery is the fact that it should be acquired from a timestamp for which AIS data is also available. This is a significant point, as the entire system is meant to determine possible unauthorized vessels based on the differences in the findings on satellite imagery and AIS data. This brings us to the second important part of the system.

5.3 Combination with AIS data

Before linking the satellite imagery and AIS data, it is important to first discuss the requirements for the AIS data. As also mentioned previously, it is important that the AIS data is acquired of a similar timestamp as the satellite imagery. If the two are from two different timestamps far apart, it is difficult to compare the two and gain conclusions from it. Additionally, it is important for the AIS data to be of the same region as the satellite imagery was acquired from. For this, a certain area can be set in order to have a testing area.

The main challenge of this monitoring system lies in determining whether a certain ship is unauthorized or not. However, this can be achieved by using a combination of satellite imagery and AIS data. If a vessel is detected in the satellite imagery but no corresponding AIS data is available for that location, it could indicate that the vessel is not broadcasting AIS and could be attempting to hide its position. This behavior may suggest involvement in illegal activities [6].

This comparison between the satellite imagery and the AIS data could be done by simply taking the vessel coordinates determined by the deep learning model, and then trying to find an exact match within the AIS data. However, this is not a realistic method. Ships are always moving, even when at anchor because of the wind and currents, which could already lead to a change in coordinates within a second. Since the recordings of the AIS data and satellite imagery are unlikely to be of the exact same timestamp, this is not a feasible method. It is thus better to determine the distance between the detections on the satellite imagery and the data points given in the AIS data. From this, the closest match can be found. Then, by calculating the average speed from the distance and time difference between the satellite detection and AIS position, it can be assessed whether the match is realistic or not. Deciding whether it is realistic or not can be done by setting a speed threshold, for example assuming that speeds above 100 km/h are unrealistic, and could indicate that the match is not likely. This could mean the ship is not broadcasting an AIS signal, and is possibly unauthorized. To give a more clear overview of the entire process, a flowchart of the proposed detection system is shown in Figure 4, summarizing the sub-processes as discussed in this chapter so far.

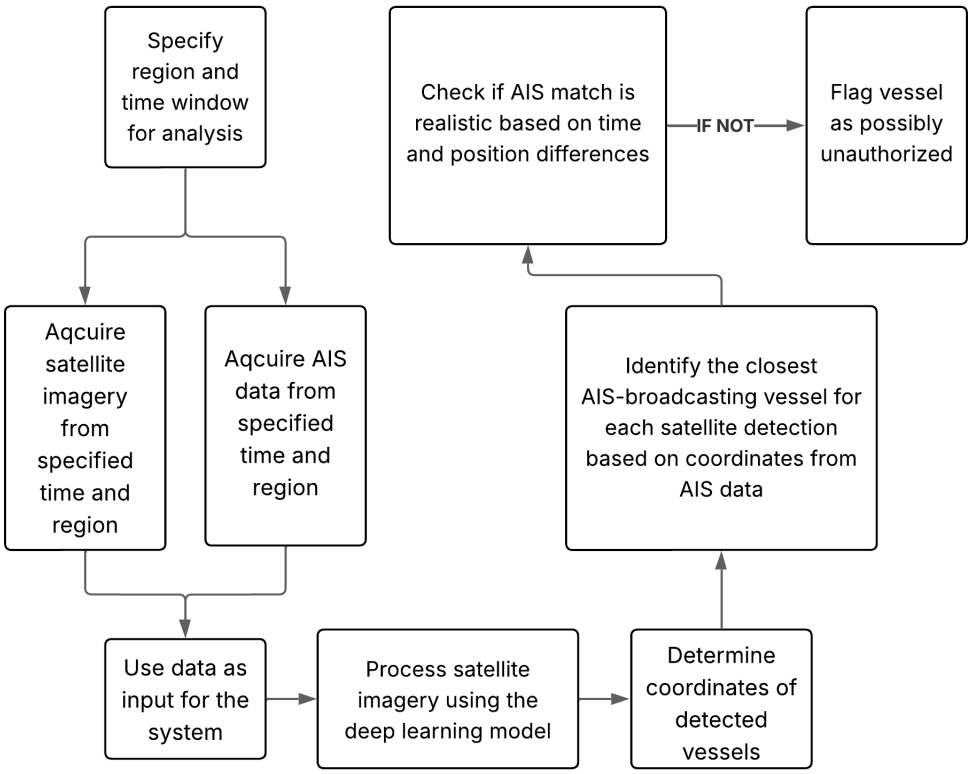


Figure 4: Flowchart of the proposed detection system

5.4 Visualisation

Next to making sure the system functions well, it is also important to consider how the detection results should be visualized. First of all, it should be possible to look back at the detections made by the deep learning model. This is important, as it allows for manual review by an authorized operator, in case of a possible unauthorized vessel. Next to this, it is also of essence to give a good overview of all the different detections and their closest matches according to AIS data. The most basic way to do this would be by outputting a file containing the satellite images including the detections (e.g. with bounding boxes), as well as a text file containing the most relevant information of every ship detected. This information consists of the coordinates of both the ship detected on the satellite imagery, and those of the closest AIS match. Additionally, the distance between these two points should be given, as well as the time difference between the data recording timestamps. Finally, some indication should also be given on the probability of this match being true, based on the difference in time and distance. This could simply be done by assigning a label ‘Likely true’ or ‘Unlikely’ to each match.

Since having a single file as output is simplistic and might not be very intuitive, something more visual would be better, if possible. An option for this would be to display the original satellite image, and display the detections of the deep learning model on this as well as the closest AIS matches, for example with a dot. These points could then be connected with a line, showing the distance between them. This would provide a clearer overview of the detections, and whether the AIS matches are realistic or not.

An even more effective solution however, would be an interactive visualization. The principle would be the same as mentioned in the previous option, but would then be made interactive. This interactive visualisation could include a date picker to select a specific day, which then displays a map showing all detections in the chosen region along with their closest AIS matches on the chosen day. Hovering over these points could reveal detailed information about the vessels, such as length, name, coordinates and distance to the closest AIS match. This interactive approach offers a lot of advantages. It allows operators to quickly filter detections by time, which for example makes it easier to focus on specific periods of interest without having to go through large amounts of data. Additionally, by showing everything on a map and sorted by date, certain patterns could be discovered that would otherwise go unnoticed.

6. Realization

Within the previous chapter, the final concept was explained in detail. This chapter will discuss the realization of the system, including all steps made in order to come to the final product.

6.1 Deep learning model

The first step of the realization process was to set up and train a deep learning model. As described before, it was decided to make use of one of the YOLO models available. For this research, the Ultralytics implementation of YOLOv11 was used [30]. This choice was made because this framework offers full integration with Python and provides a wide range of built-in functions that simplify the entire workflow, especially for training, testing and validating deep learning models.

To train the YOLOv11 model, the initial approach was to use a pre-existing dataset. If this dataset would lead to sufficient results, it would save a significant amount of time by avoiding manually having to create a custom dataset. The previously discussed MASATI dataset [19] was used for this purpose. The dataset was divided using an 80/20 split, where 80% of the images were used for training and the remaining 20% for testing. The training process was conducted over 100 epochs. The code used for training the model can be found in Appendix B.

The results from this first training showed promising performance, with the model achieving a precision of over 90% and a recall of around 85% based on its performance on the testing set. However, the real question was how it would perform on imagery collected from PlanetLabs [26], the software used for this project. As seen in the figure below, the model also seemed effective for detecting certain types of vessels on PlanetLabs [26] imagery. However, the model began to struggle when detecting more complex vessels, such as those with brighter colors or significantly larger sizes. This could suggest that the model is overfitting to the MASATI dataset [19], or it could indicate significant differences between the imagery from PlanetLabs [26] and the dataset used for training.

Figure 5: Two examples of the performance of the first version of the deep learning model on PlanetLabs [26] imagery.

Something that can be seen as well is that the bounding boxes are only rotated horizontally, while vessels are rotated in all directions. To address this, a version of YOLOv11 that supports Oriented Bounding Boxes (OBB) was used instead [30], which was also included in the Ultralytics framework [30]. By doing so, the model is able to predict not just the position and size of each object, but also its rotation angle.

The next step was to create a dataset containing imagery from PlanetLabs [26] that could be used to further train the model. To achieve this, imagery from the target region, Cyprus and its surrounding waters, was collected. Most of the imagery was sourced from areas near harbors, as these locations frequently contained ships, which was crucial for the model's training. However, no images including coasts or harbor infrastructure were used, as the complex terrain in these areas complicates the training process. Only purely marine environments with ships were included.

PlanetLabs [26] provides imagery in GeoTIFF format, which is not directly suitable for training the model and requires preprocessing, also since the downloaded images were very large in size, making them impractical to use as a whole. Therefore, the processing stage consisted of splitting each image into non-overlapping 512×512 pixel tiles, in the form of .png files. However, no other preprocessing steps were done. These images were then labelled manually using Label Studio [29]. Some examples of these labelled images can be seen in the figure below. After labeling all images, the dataset was split into a training and testing set using an 80/20 ratio.

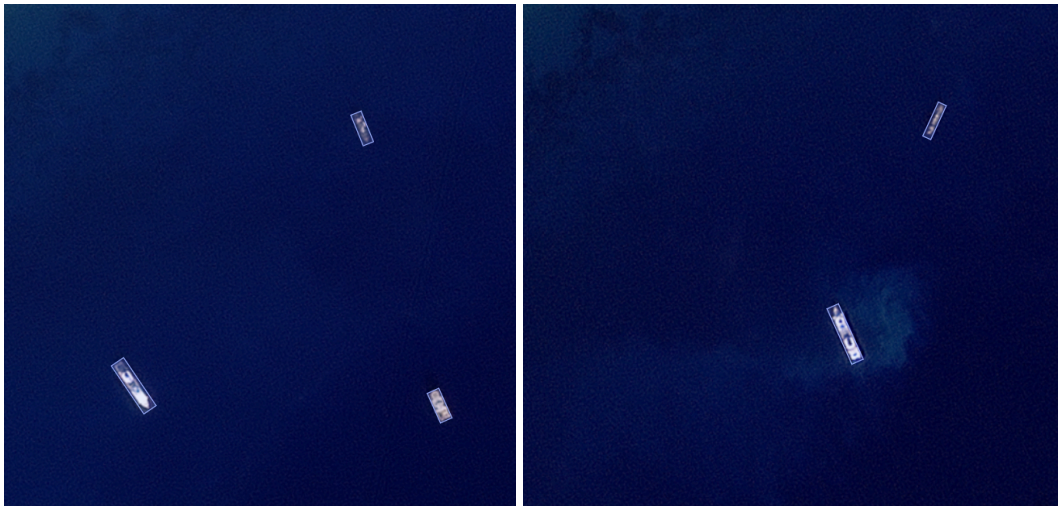


Figure 6: Examples of PlanetLabs [26] imagery labelled using Label Studio [29].

However, one important limitation of this dataset is that the labels are purely based on human inspection of the satellite images, without external verification to confirm whether each object was indeed a ship. This could have led to some labels being incorrect, especially for small objects. This limitation should thus be kept in mind when considering the reliability of the model.

The training set was then used to train the YOLOv11 model from scratch again, using the same training method as before. After evaluating the model using the testing set, the results were very promising with a precision of 100%, a recall of 98.2% and an F1-score of 99.1%. It shows that all detections made by the model were indeed ships, and only 1.8% of all the ships were left undetected. Some examples of the detections of the model are displayed in Figure 7.

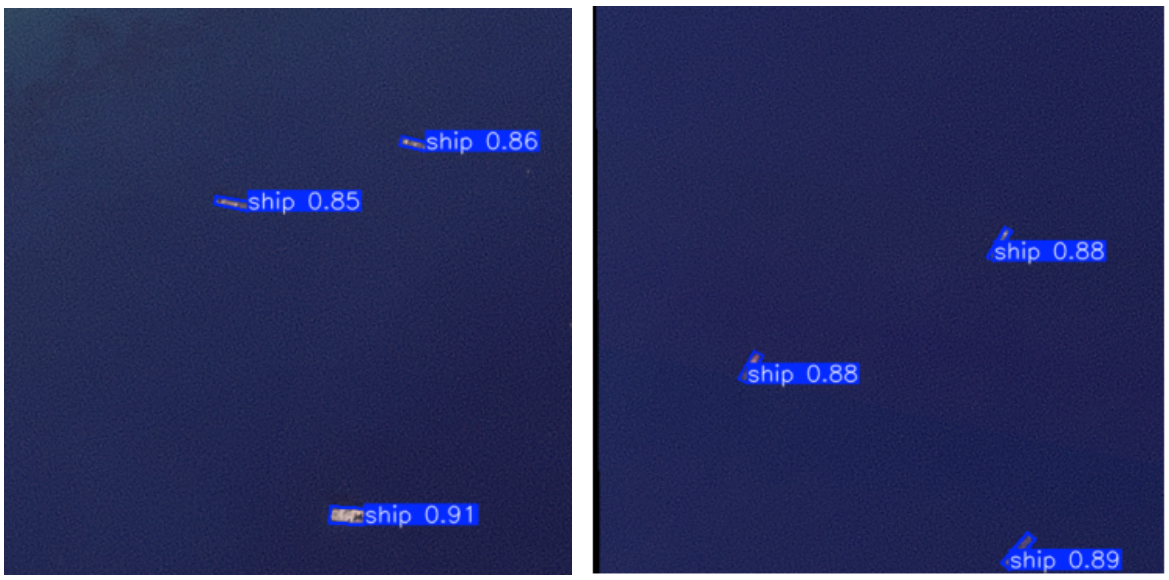


Figure 7: Examples of performance of the YOLOv11 model trained on PlanetLabs [26] imagery.

However, the 1.8% of the ships that were missed also had to be analyzed in order to get a better view of the weaknesses of the model. From this, it appeared that most of these errors occurred with smaller vessels, particularly those leaving a visible wave trail behind. These types of ships are often not detected at all. Some examples of ships that remained undetected are displayed below in Figure 8.

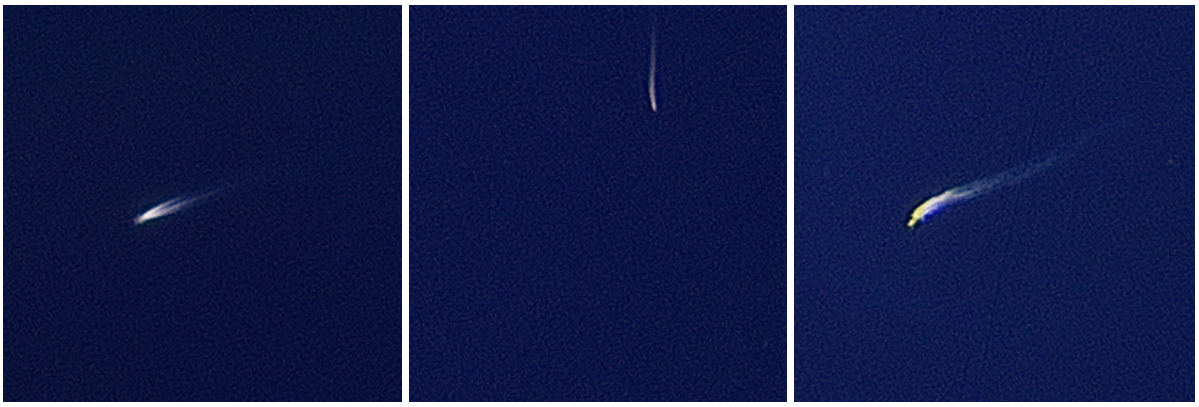


Figure 8: Examples of vessels often missed by the deep learning model.

The following step was thus to also train the model more on these cases, in order to remove this vulnerability of the model. The first approach for this was to create a new classification specifically for these kinds of ships. However, this significantly lowered the performance of the model down to scores of below 50% for precision and recall, which is why this was decided not to be effective. Alternatively, the dataset used before was expanded with more images, specifically containing more cases as displayed in Figure 8. These images were again acquired using the PlanetLabs software [26], and labeled using Label Studio [29]. The expanded dataset was then split into a training and testing set, again using an 80/20 ratio. Afterwards, a new model was trained from scratch, using the training set and the YOLOv11 architecture provided by Ultralytics [30] as a foundation. This approach proved to be more effective, as shown by the evaluation on the testing set; A score of 100% was reached for

recall, as well as a 90.1% score for precision. Figure 9 shows two examples of the model's performance after this training.

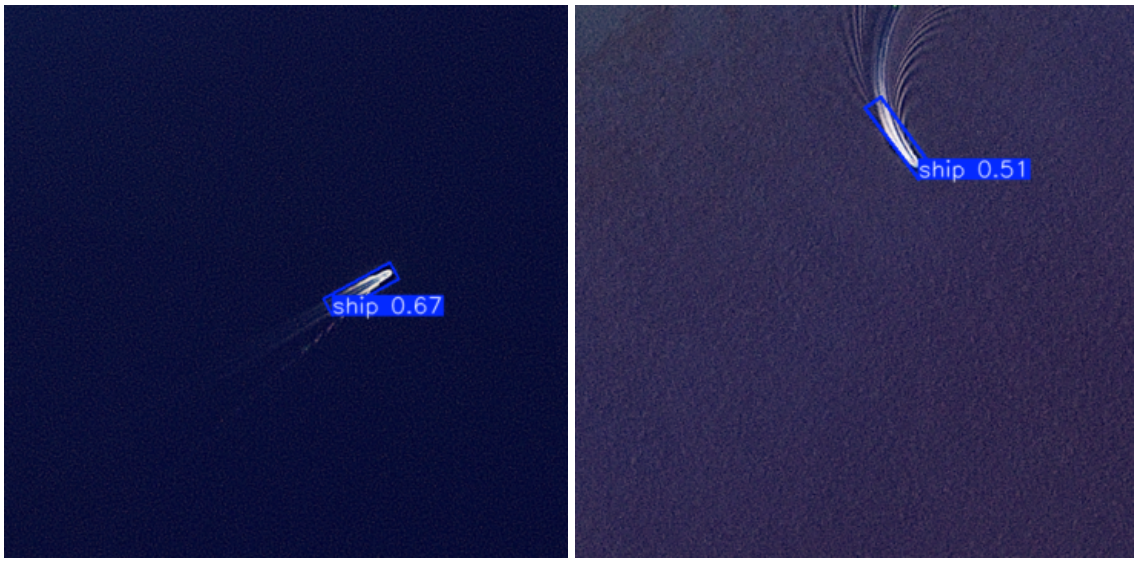


Figure 9: Examples of the YOLOv11 model's performance after optimizing it on small ships with a wave trail.

However, there is also a limitation within this model. As the ships with wave trails are often only a few pixels in size, the trail is the most visually prominent feature. This is why it was deliberately chosen to include the wave trail in the annotation when manually drawing the bounding boxes. However, the trail sometimes fades gradually over a long distance, making it difficult to determine where exactly the bounding box should end. As a result, the extent to which the trail was included could have varied between annotations. This issue will be further analyzed alongside the performance metrics in Chapter 7, after discussing the realization of the remaining elements within the system.

6.2 Geolocation of detected ships

Now that the YOLOv11 model was able to detect ships on our satellite imagery, the next step was to be able to determine the exact geographic coordinates of these ships. In order to do this, the GeoTIFF files, a specific type of Tagged Image File (TIF) acquired from PlanetLabs [26] when downloading satellite imagery, play a crucial role. While GeoTIFFs are standard image files, they also include embedded spatial metadata stored in so-called TIF tags. These tags contain information such as the coordinate reference system, resolution, and spatial extent, enabling us to calculate the precise geographic coordinates of each pixel in the image [31]. A tool that simplifies this task is the open-source Python library Rasterio [32]. This add-on provides easy access to geospatial raster data and offers a wide range of functions for reading metadata, transforming coordinates, and working with GeoTIFF files efficiently. The YOLOv11 model first determines the pixel coordinates of the bounding boxes. These pixel coordinates are then sent through to the function which calculates the geocoordinates, using Rasterio [32]. The code used for this can be found in Appendix C, within the function '*geo_location*'.

Before the bounding box pixel coordinates are passed to the coordinate transformation function, one important aspect must be addressed. The original TIF image has to be split into smaller .png tiles of 512x512 pixels, before running the YOLOv11 model for ship detection. These tiles may slightly overlap to ensure complete coverage of the entire satellite image, including border areas, so that no ships are missed.

Because these detections are made on the individual tiles, the pixel coordinates of the bounding boxes are local to each tile, which means they do not directly correspond to the pixel coordinates of the original TIF image. To map the detected bounding boxes back to the original image, an offset based on the tile's position is added to each detection's coordinates. Additionally, to avoid duplicate detections caused by overlapping tiles, a post processing step is applied to remove redundant bounding boxes. This is done by calculating the distance between the centers of detected ships, again using functions provided by Rasterio [32], and removing any duplicates that are located within 30 meters of one another. The code responsible for tiling the image and handling these coordinate transformations is provided in Appendix C, specifically in the '*main*' function.

6.3 AIS data integration and matching

After the geographical location of the detected ships has been determined, the next step is to compare this information with AIS data from the same region and time period. To create the AIS datasets for this, CSV files were downloaded from MarineTraffic [27], containing all vessel records within the relevant geographic region and around the timestamp of the satellite image. In addition, separate AIS datasets were collected for several timestamps following the satellite image timestamp, in order to gain insight into the trajectories of ships over time. The raw AIS data includes information such as geographic coordinates, vessel name, classification, and timestamp of the broadcast. Before using the AIS data, preprocessing was required. This consisted of filtering out all AIS entries recorded more than 30 minutes before or after the satellite image timestamp. This 30-minute time threshold helps avoid outdated or unrelated records from being matched. In order to achieve this filtering, the AIS data was handled using Pandas, a data manipulation library in Python [33].

The filtered AIS data is then used in a function called ais_check, which also receives a list containing all the geographic coordinates of the detected ships, in the format 'latitude, longitude'. The coordinate system called WGS84 is used for this, which is the standard geographic coordinate system used at sea [34]. This is also the format in which the coordinates are listed within the AIS data. Each detected ship is then matched to the closest vessel within the AIS records based on location, while ensuring that no AIS entry is matched to more than one detection. Distance calculations were again handled using functions provided by Rasterio [32].

For each matched pair, a dictionary is created that stores relevant information about the potential match between a detected ship and an AIS-recording vessel. This dictionary includes the coordinates of both the satellite detection and the AIS position. It also stores the distance between these two positions, the name of the vessel from the AIS data, the timestamps of both the satellite image acquisition and the AIS record, as well as the time difference between them. In addition, it includes the estimated speed the vessel would need to travel to cover the distance in the given time. Next to that, the estimated length of the detected ship is added, based on the largest side of its bounding box. In addition the directory to the satellite image is also added. Furthermore, AIS positions recorded later on the same day are also collected and added when available, enabling the visualization of

vessel trajectories over time. Finally, a label is added based on the probability that a match is true. This is based on the average speed. If a vessel has an average speed of higher than 45 km/h, it will be labelled as unrealistic. This threshold was chosen based on research by Reaper [37], which found that most ships do not exceed speeds of 20 knots (approximately 38 km/h). Ships that do surpass this speed are typically container ships or vehicle carriers. Therefore, a threshold of 45 km/h was set to account for these faster vessels while still identifying unrealistic matches, helping to reduce false positives in the system.
An example of a dictionary as saved for every AIS match is displayed in Figure 10. All Python code used to achieve the actions described are included in Appendix C, specifically within the function *'ais_check'.*

```
ais_match = {
      'sat': (34.697795615052456, 33.339429362159166),
      'ais': (34.698254, 33.339062),
      'name': 'PRATINCOLE PACIFIC',
      'length': 195.1,      # In meters
      'distance': 61.0,     # In meters
      'delta_time': 3.7,    # In seconds
      'speed': 5.0,         # In km/h
      'status': 'Realistic',
      'time_satellite': '2025-05-18 08:50:40',
      'time_ais': '2025-05-18 08:47:00',
      'tif_path': 'Visual_clip.tif',
      'later_ais_positions': [
        {'lat': 34.698246, 'lon': 33.339115, 'time': '2025-05-18 08:53:00'},
        {'lat': 34.698193, 'lon': 33.33942, 'time': '2025-05-18 08:59:00'},
        {'lat': 34.698513, 'lon': 33.340118, 'time': '2025-05-18 10:01:00'}
      ]
    }
```

Figure 10: Example of one of the saved dictionaries containing information on a match with AIS data.

6.4 Visualization

Now that all information that is needed is determined and calculated by the system, the findings should be visualized. First, all match dictionaries are grouped and sorted by date, and stored in a larger dictionary. This data is then passed to the '*add_cropped_images*' function. In this function, each detection is processed by cropping the corresponding satellite image around the detected ship location. A zoomed-in image is saved, and the file path to this image is added to the corresponding match dictionary. After going over all detections, the entire dictionary, consisting of all matches sorted by date, is returned. The code of this function can be seen in Appendix C. Then, this dictionary is sent to the main visualization function, *'generate_ship_maps'*. This function generates an interactive map for each day in the data, using the Folium library for Python [35]. It also generates a starting page, on which a date can be chosen which wants to be reviewed. Once a date is picked, the detections made by the YOLOv11 model and their AIS matches from this specific date are plotted on a map using markers, and connected with a line between them. Additionally, the trajectory of the AIS broadcasting ship is visualized with dark blue dots.

When clicking on the marker of a detected ship, the image created in the '*add_cropped_images'* function will pop up, as well as information on this ship. This information consists of the coordinates, estimated length of the detected vessel and the timestamp of acquisition of the satellite image. Next to this, an estimated classification is shown; If a ship is larger than 60 meters, it is labelled as 'Tanker/Cargo/Passenger/Other'. If the vessel is estimated to be smaller, it will get the label "Fishing/Pleasure Craft". This classification threshold was set up based on observations of vessel size distributions within the information displayed on MarineTraffic [27]. However, no other studies were used to define the 60-meter threshold. The estimated length ranges are presented in the table below. Example images have also been included. However, since it was not possible to verify the exact classification of these ships, only those with the highest certainty were selected. This is also the reason some categories were left open. Therefore, it cannot be guaranteed that the depicted ships correspond precisely to the stated classifications.

Classification	**Length range**	**Example image**
Tanker	100 - 300 meters	
Cargo	100 - 300 meters	
Passenger	60 - 300 meters	-
Fishing	10 - 60 meters	-
Pleasure craft	5 - 40 meters	-

Figure 11: Table showing classifications and their estimated length ranges based on data found on MarineTraffic [27]. Additionally, example images are added for two of the classes, acquired using PlanetLabs software [26].

Additionally, when hovering over the marker of the AIS match, the coordinates appear, as well as the name and classification of the vessel, and the timestamp of the recording. When hovering over the connecting line, more information is displayed, specifically distance, time difference (between AIS recording and satellite image acquisition), average speed (in both km/h and knots), and whether the match is realistic or unrealistic. In the figure below, two examples of this visualization have been displayed.

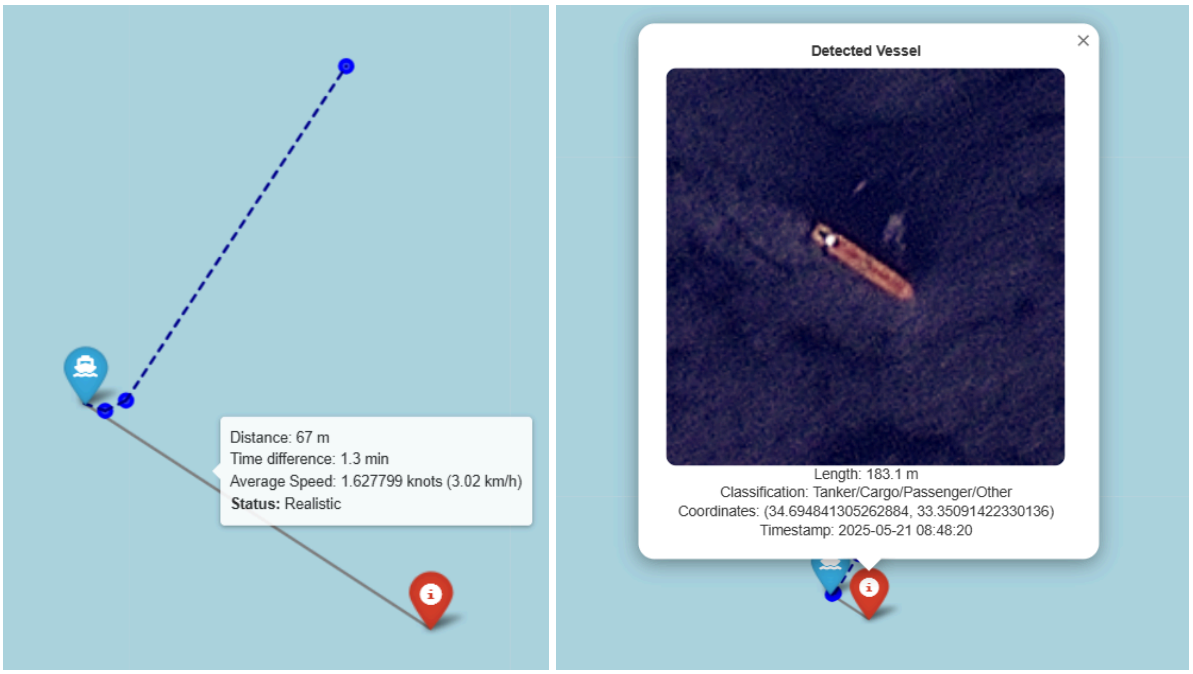


Figure 12: Example of the visualizations shown on the interactive webpage

To create the web pages, HTML code was written. This code, together with the rest of the code used for the visualization, can be found in Appendix C, specifically in the function *'generate_ship_maps'.* For the sake of simplicity, the output was not hosted as a live website, but rather consists of locally saved static HTML files.

A simplified overview of the code structure is presented in Figure 13, illustrating the roles and functionalities of the main functions. The complete Python script is included in Appendix C.

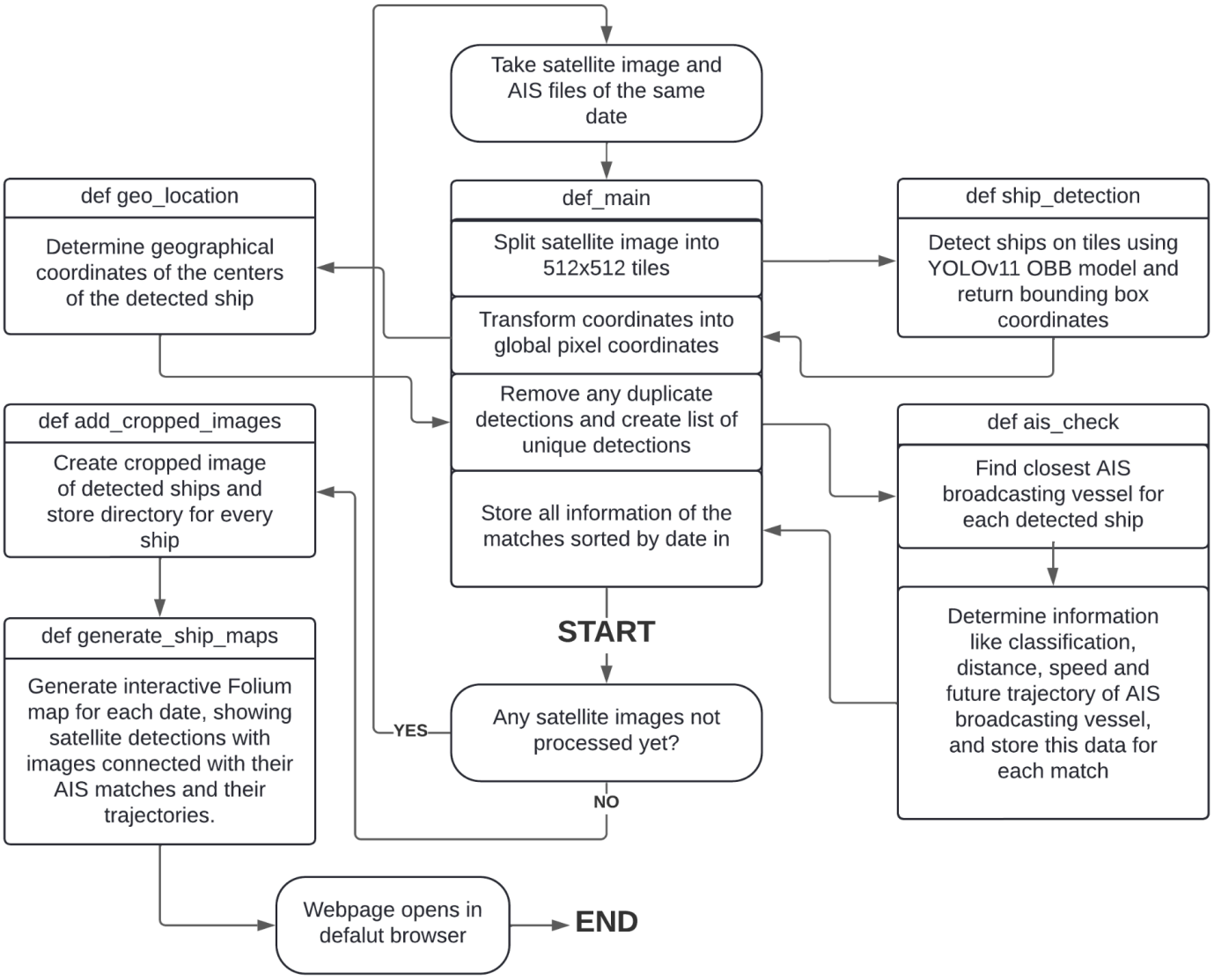


Figure 13: Overview of the Python script and its main functions.

6.5 SAR model

After finalizing the system for optical imagery, a second deep learning model was developed, this time optimized for SAR imagery. As with the previous model, the dataset was manually created using Label Studio [29]. The SAR imagery was obtained from the Copernicus Browser, specifically from Sentinel-1 data covering the southern maritime region of Cyprus [36]. The preprocessing of this SAR data began by splitting the large satellite images into non-overlapping 1024×1024 pixel tiles. Due to the noise in the SAR data, which could negatively affect model performance, the tiles were converted to grayscale. Once this preprocessing step was complete, all image tiles were manually labeled. The dataset was then split into training and testing sets using an 80/20 ratio.

However, there is a limitation to this dataset. Unlike optical imagery, SAR imagery is more challenging to interpret due to its abstract representation of surfaces and objects. Since there was no way to verify whether a particular shape in the image was actually a ship, labeling certain objects as ships was based solely on human judgment. This introduces

a degree of subjectivity to the dataset which could affect performance, and should thus be kept in mind.

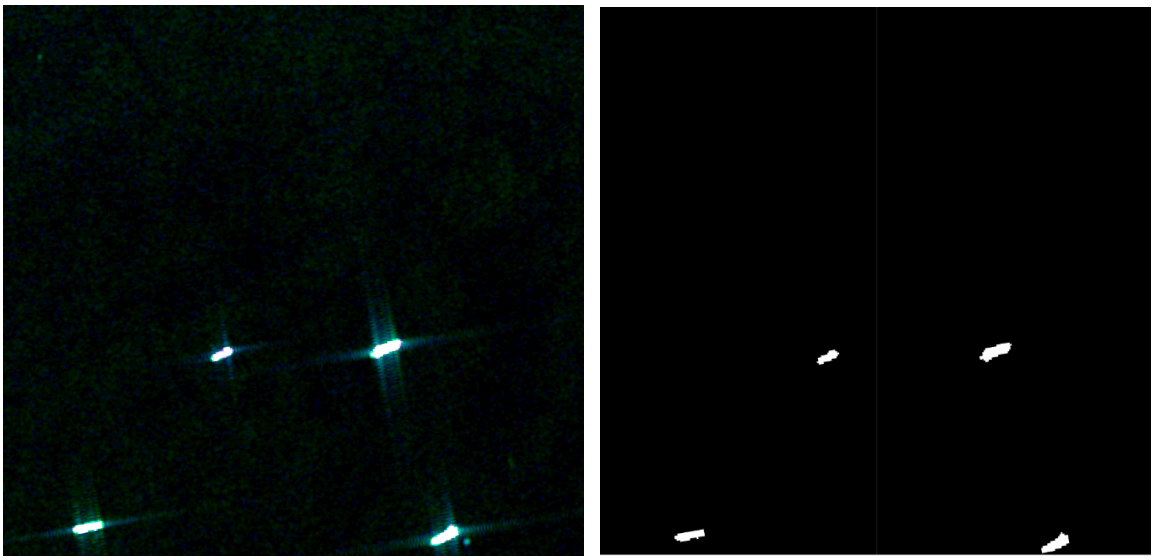


Figure 14: Image before (left) and after processing (right).

Same as previously, a similar code structure as shown in Appendix B was used to train Ultralytics' YOLOv11 model [30]. After training the model on the training set, the model was evaluated on the testing set, which showed high performance. The model reached a score of 98.1% for precision, as well as 99.9% for recall. This was more than sufficient for the project, and thus no further training sessions were executed.

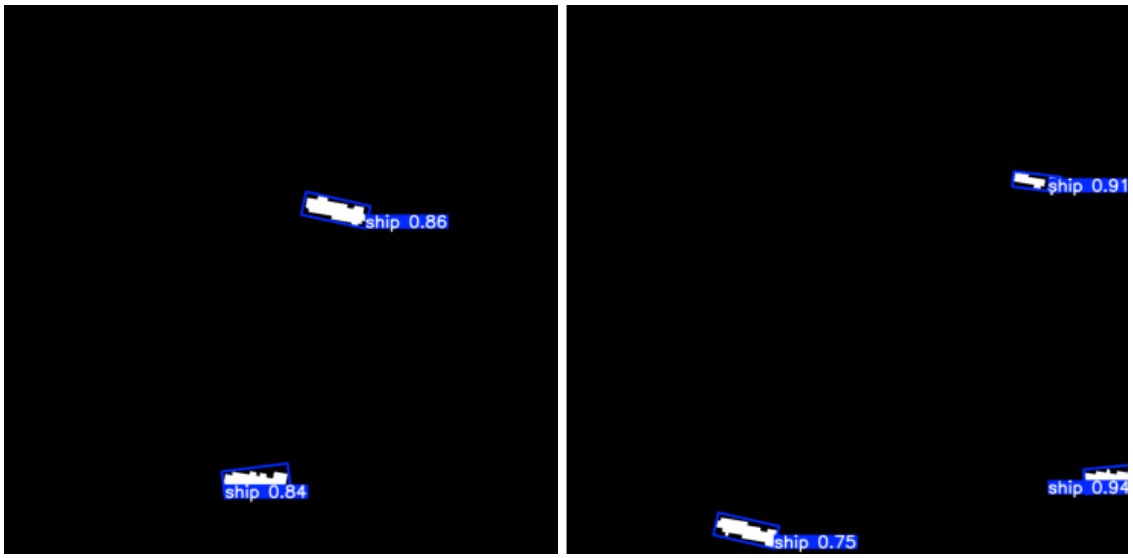


Figure 15: Examples of performance of the YOLOv11 [30] model trained on SAR imagery.

7. Evaluation

In this chapter, all components of the system will be evaluated based on their performance. Their capabilities and limitations will be discussed in detail. First, the performance of the deep learning models will be evaluated, followed by an analysis of the AIS matching process. Finally, the potential and usability of the visualization will be assessed.

7.1 Deep learning models

To evaluate the performance of the deep learning models, a set of standard metrics will be used: precision, recall, F1-score, and mean Average Precision (mAP). Both models were trained on a manually created dataset, as described in Chapter 6, using an 80/20 train/test split. The evaluation will thus be performed on the images within the testing part, containing images the model(s) have not seen during training. This process is carried out using the built-in evaluation functions provided by the Ultralytics framework [30]. The full code used for evaluating the models can be found in Appendix D.

7.1.1 PlanetLabs model

The performance metrics for the model trained on optical imagery from PlanetLabs [26] can be seen in the figure below. These were determined based on the performance on the testing set, consisting of images the model had not seen during training.

Precision	**Recall**	**F1-score**	**mAP 50**	**mAP 0.5-95**
0.9011	1.000	0.9480	0.9801	0.6440

Figure 16: Performance metrics of the YOLOv11 model trained on optical imagery.

The first notable result is the recall score of 100%, indicating that all ships in the images were detected by the model. This is a promising outcome, as it suggests that no ships were missed. The precision score of 90% shows that the majority of detections were correct, although around 10% of the detections were false positives. Most of these errors involved detecting the same ship twice, often occurring with vessels that had a visible wave trail, as can be seen in the figure below. For this project however, a balance like this, high recall with slightly lower precision, was preferred. Prioritizing a high recall ensures that all ships, including potentially unauthorized ones, are detected. While this approach may result in some false alarms, it minimizes the risk of leaving vessels undetected. In contrast, optimizing for very high precision would likely reduce recall, increasing the number of false negatives. This could result in ships going undetected, which is more problematic in the context of monitoring unauthorized maritime activity. Therefore, the current trade-off is considered acceptable and well-suited to the goals of this project. Next to this, a precision of 90.1% is still high, which is also indicated by the high F1-score of 94.8%.

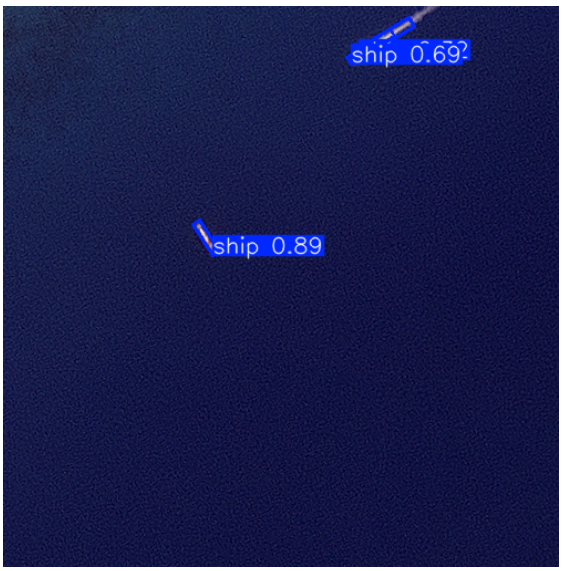


Figure 17: Example of an incorrect detection resulting in a duplicate bounding box for the same ship.

In addition to precision and recall, the mAP score provides valuable insight into the model's performance, as it also accounts for Intersection over Union (IoU), a measure of how much the predicted bounding box overlaps with the ground truth box. For instance, the score of 98% for mAP 50 indicates that 98% of the correct detections had an IoU of at least 50%, meaning the predicted boxes overlapped reasonably well with the actual ship locations. However, there is also the mAP 0.5-0.95 score, which is calculated as the mean of the Average Precision scores at ten different IoU thresholds, ranging from 0.5 to 0.95 in increments of 0.05. In contrast to mAP 50, this score is a lot lower for the model, with 64.4%. This lower score suggests that while the model is highly effective at detecting ships, the bounding boxes are not consistently placed with high precision. In other words, although the ships are found, the predicted boxes are often not tightly aligned with the ground truth. This however can be explained. As also discussed in Chapter 6, small ships with wave 'tails' were a challenge for the model. To address this, these wave trails were included in the bounding boxes during annotation. However, because these trails gradually faded out rather than ending abruptly, it was often difficult to determine where the bounding box should end. This likely led to variations between the predicted bounding boxes and the manually annotated ones. It was thus not necessarily due to poor model performance, but because there was no clear boundary and therefore no definitive "correct" or "incorrect" bounding box.

It is also important to understand the limitations of the model in terms of the minimum ship size it can detect. Based on all detections from the testing set, the smallest length of a correctly predicted bounding box was 10.73 pixels. Given the spatial resolution of PlanetLabs imagery is 3 meters per pixel [26], this corresponds to a ship length of approximately 32.19 meters. The largest detected ship had a bounding box length of 93.11 pixels, which translates to roughly 279.33 meters. Although these examples do not define the absolute detection boundaries of the model, they do reflect the size range of ships that were most frequently detected during testing. Smaller vessels may still be within the model's detection capabilities, but they were likely underrepresented in the testing set, possibly due to chance, or human bias during the manual labeling process. For instance, smaller ships may have been more difficult to identify and were unintentionally excluded when creating the dataset. This could have led to a model that is more optimized for detecting larger vessels.

7.1.2 SAR model

The performance metrics for the model trained on SAR imagery from the Copernicus browser [36] can be seen in the figure below. These were determined based on the performance on the testing set, including images the model had not seen during training.

Precision	Recall	F1-score	mAP 50	mAP 0.5-95
0.984	0.999	0.9913	0.995	0.628

Figure 18: Performance metrics of the YOLOv11 model trained on SAR imagery.

The scores show exceptionally high scores for precision and recall, with 98.4% and 99.9% respectively. This means only 1.6% of the detections were false positives, and only 0.1% of the ships within the testing set was left undetected. Of the few errors that still occur, most involve the same ship being detected multiple times. An example of this is shown in Figure 19. The high recall and precision is also portrayed by the high F1- score of 99.13%, indicating the strong performance of the model overall. However, just as the model trained for optical imagery, a big drop can be seen from mAP 50 to mAP 0.5-95, which does indicate that the bounding boxes are not placed with a very high precision. This could be the result of small errors made during the manual bounding box placement, leading to a mismatch between predicted and true boxes. Even a small deviation could significantly affect the mAP score for higher IoU thresholds, IoU ≥ 0.9.

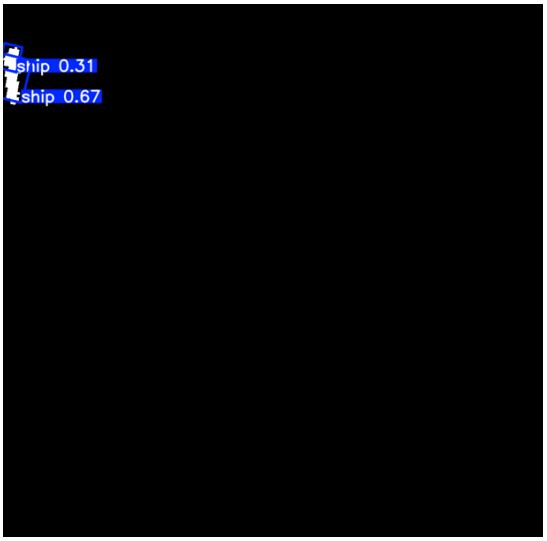


Figure 19: Example of an incorrect detection resulting in a duplicate bounding box for the same ship.

Finally, regarding the lengths of the ships detected, the smallest ship detected within the testing set was equal to 22 pixels, and the largest around 141 pixels. However, the spatial resolution of the imagery from the Copernicus browser [36] differs widely depending on the region. This can go from as small as 1 meter per pixel to 5 meters per pixel. This means that a ship measuring 22 pixels in length could correspond to anywhere between 22 and 110 meters in reality, depending on the spatial resolution of the imagery. Similarly, a ship of 141 pixels could range from 141 meters up to approximately 700 meters. However, there is therefore no exact answer to the question about the maximum or minimum ship size that can

be detected. It is also worth noting that all labeling was done manually, and interpreting SAR imagery is generally a challenging task. As a result, the dataset may have unintentionally become biased toward larger vessels, which are more visually prominent and easier to identify with confidence. Smaller ships may have been overlooked or excluded during annotation due to uncertainty, potentially leading the model to be less optimized for detecting those smaller targets.

7.2 AIS matching

In order to test the final system, imagery and AIS data from around 3-4 days was gathered, from the same region and time. However, as the amount of AIS data that could be acquired was limited, the system was tested with just the deep learning model trained on PlanetLabs [26] imagery, as this was the initial focus of this research.

In order to test the performance of the system, the visualization tool was used in combination with the imagery including the detections in the form of bounding boxes. However, as there was no exact way to verify whether a certain match was correct, these matches had to be evaluated using just manual human judgment.

Across all dates used to evaluate the system, it can be observed that most detected ships were correctly matched to AIS-broadcasting vessels. As shown in the figure below, each detected ship was matched to an AIS signal located very close by, within a maximum distance of 80 meters. This distance is reasonable, especially considering that the ships in question are estimated to be over 100 meters in length. This estimated length provides an additional way to validate whether a match is likely to be correct.

While vessel dimensions were not available within the AIS data used for this project, this information can be verified manually. For instance, the ship located farthest to the right on the map was estimated by the system to be 183.1 meters long. This detection was matched with the vessel ‘Histria Prince’, which MarineTraffic [27] lists as having a length of 179.91 meters. With a difference of just over 3 meters, and given the very close proximity on the map, this appears to be a very realistic match.

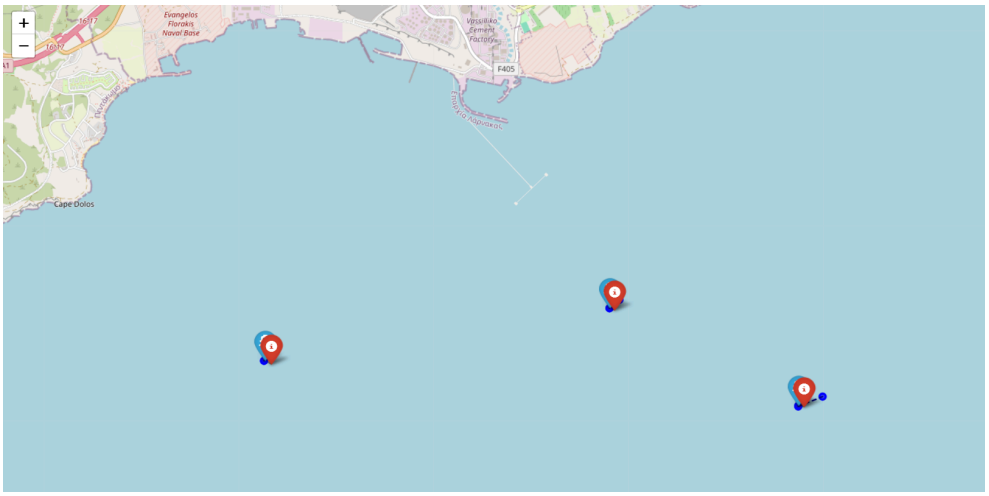


Figure 20: Map showing detections from satellite imagery (red markers) and their AIS matches (blue markers) south of the Vassiliko port.

The other two ships show slightly larger differences between estimated and actual lengths. The system estimated their lengths at 198 and 168 meters, while MarineTraffic lists both vessels as being 183 meters long. Despite these larger differences, these matches remain reasonable, especially since the precision of the bounding boxes (with which the length is calculated) was found to be lacking sometimes, as discussed in the previous section of this chapter. However, there is one specific situation where the estimated length can be significantly inaccurate. This is the case for ships with a long wave trail, a frequently discussed issue in this research. Since the bounding boxes almost always include this trail, the resulting length is much greater than the ship's actual size.

Another segment of the system is the prediction of the classification. However, since this classification is based only on the estimated vessel length and limited to just two possible categories, its precision is low. For instance, a very large ship could be anything from a passenger vessel to a tanker. While the classification often appears correct in cases where the match is realistic, this accuracy cannot be considered a meaningful achievement, as it results more from broad assumptions than from detailed analysis.

Moving on, one of the most important components of the system is the decision to determine whether a detected ship is possibly unauthorized. In other words, whether it is broadcasting AIS signals or not based on the plausibility of a match with AIS data. Naturally, it is not possible to verify with certainty whether a detected vessel is unauthorized. Therefore, evaluating how realistic or unrealistic the AIS match appears has only been based on manual human judgement.

For many of the vessels labeled as unrealistic, the system seems to make correct judgments. For example, one detected ship was matched to an AIS-broadcasting vessel located closest to it, but still nearly ten kilometers away, despite only a 20-second time gap between the detections. This was labelled as an unrealistic match by the system, since the average speed would have been around 2000 km/h for this to be true. Most of these mismatches involve small vessels and are often linked to stationary ships docked in harbors, which clearly indicates an unrealistic match.

However, there are also cases where the estimated speed slightly exceeds the threshold used to classify a match as unrealistic. An example is shown in the figure below. The red marker indicates the detected vessel, while the blue marker shows the matched AIS location. The two points are separated by nearly 3.5 kilometers, and the time difference is approximately four minutes. Based on this, the system calculates a speed of 48 km/h and labels the match as unrealistic. Yet, when looking at the next AIS point in the ship’s trajectory, which is also about four minutes later and continuing in the exact same direction, it strongly suggests that the match was in fact correct. However, we cannot be sure, as this is only based on manual human judgement, and no other verification tool was used for this. However, this example still highlights a key limitation of the current justification system, which is the fact that the system is relying purely on average speed to assess match validity can lead to false negatives. A more robust approach should also consider additional context, such as trajectory consistency and vessel behavior over time, to improve reliability.

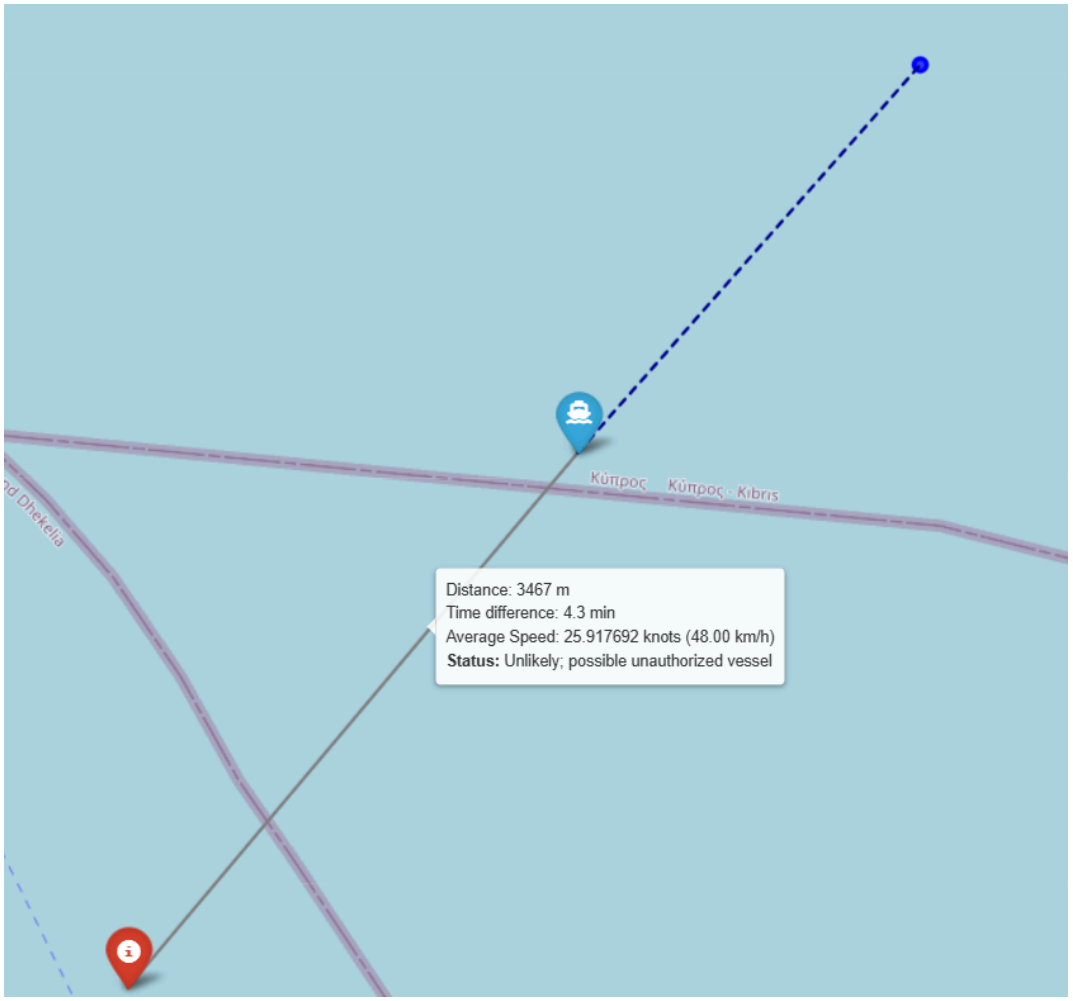


Figure 21: Example of a vessel possibly mislabeled as unauthorized.

Additionally, a specific scenario in which the system was observed to perform unreliably is when multiple ships are located very close to each other in the imagery. During testing, it was noted that in such situations, the detection model occasionally produced incorrect bounding boxes, mistakenly identifying a single vessel as two separate ships or the other way around. This not only affected the accuracy of the detections, but also introduced complications in the AIS matching process. When multiple ships are close to each other, their AIS-broadcasted positions (if they are broadcasting) are also close, making it difficult to confidently associate the correct AIS entry with the corresponding detection. As a result, mismatches can occur. These issues reduce the reliability of the system in crowded maritime areas, and thus shows the need for improved methods, such as incorporating heading or trajectory information into the matching algorithm.

Another thing worth noting is the surprisingly low accuracy of the deep learning model for this evaluation experiment. While the model performed nearly flawlessly on the testing set, a considerable number of vessels were missed in this real-world test. Examples of vessels that were not detected can be seen in the figure below.

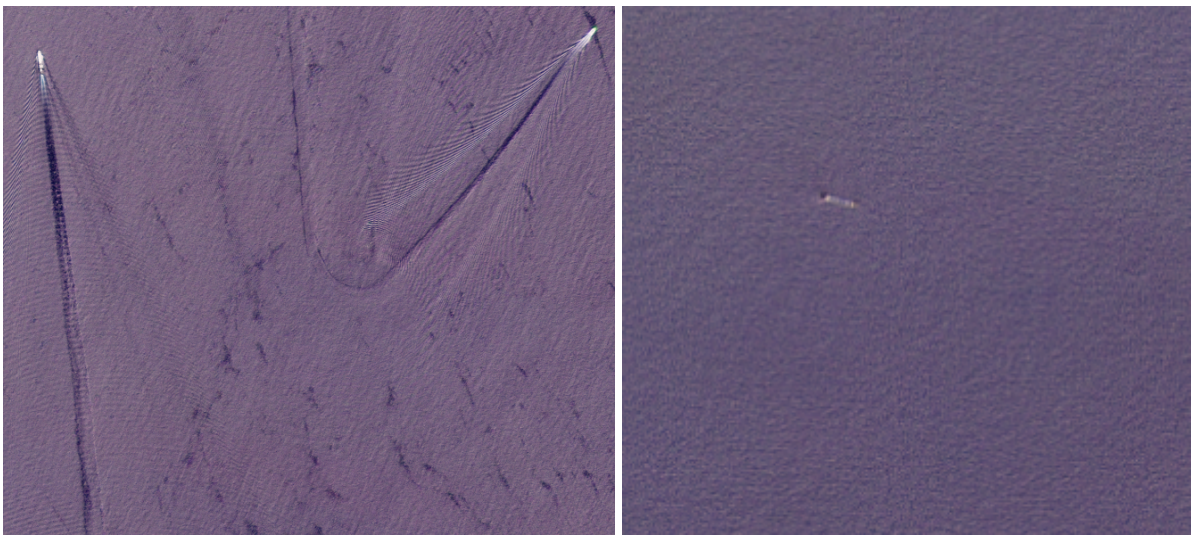


Figure 22: Examples of vessels left undetected during evaluation of the system as a whole

Several factors could have caused this drop in performance. One potential reason is a domain shift caused by differences in imagery. The evaluation images may have had varying sea backgrounds, lighting conditions, or weather patterns, especially considering that most of the training data was collected during a different season. These changes could have significantly affected model generalization. Another explanation could be overfitting. Although the model achieved high accuracy during evaluation on the testing set, it's important to note that both the training and testing sets originated from the same source and were created using similar acquisition and annotation methods. While the specific images were different, they were sampled from a similar distribution. In other words, the model was likely exposed to similar lighting conditions, sea backgrounds, and vessel types in both sets. As a result, the high performance might not fully reflect the model's ability to generalize to more diverse or unseen scenarios. For instance, if the dataset contained a large number of ships with similar colors, sizes, or orientations, or if certain vessel types were more consistently annotated, the model may have involuntarily been optimized for these conditions.

7.3 Visualization

Another important aspect to evaluate is the user-friendliness of the website. This was assessed through a small user study, where participants were asked to explore the visualization and then complete a short questionnaire. All questions and responses are listed in Appendix E.

To gain insight into the background of the participants, the first question of the questionnaire asked about their experience in the field. Out of the seven participants, only two indicated that they had some familiarity with the subject. No domain experts were thus involved in the evaluation of the visualization. Based on the users' responses, both the intuitiveness of controlling the visualization and the clarity of all elements on the map were rated highly, with average scores of 4.3 and 4.7 out of 5, respectively. In addition, all participants agreed that the visualization could be useful for real-world monitoring applications. However, one user expressed concerns about existing competitors in the market. This participant thus suggested that focusing on something specific, such as the

detection of smaller ships, could help differentiate the system. This was also highlighted by another user, who claimed smaller ships are often the ones involved in illegal activities.

Another option that could be added according to the participants, included the ability to select specific areas on the map and download data for those regions. Additionally, a recommendation was to implement alert systems, for instance to notify users when a vessel enters a protected area. Next to that, many participants emphasized the need for more detailed data in general. Suggestions included adding images of AIS-identified vessels for comparison with detections, incorporating historical trajectory data, and providing more detailed vessel information. Finally, one user recommended incorporating SAR suitability, which aligns with the desired system features identified during the initial concept development.

7.4 Assessment of set requirements

In Chapter 4, a set of requirements was formulated, following the MoSCoW method. For clarity, the table with requirements has been displayed below. Within the ‘Must’ category, there were a total of four requirements.

Must	Should	Could	Won’t
Be able to detect at least 85% of the ships in the provided satellite imagery.	Be able to operate with imagery from Cyprus and surroundings	Be made to work in different regions (with different climates)	Be able to work real-time (without uploading data manually)
Work on optical imagery.	Have an F1-score of 80% or higher.	Add estimation of classification and speed of ships	-
Be able to determine approximate location of detected ship	Not have a high operating cost.	Include an online user interface	-
Be able to determine whether ship is (potentially) illegal or not using approximated location and AIS data	Be able to work with both SAR and optical imagery	Determine ship trajectories	-

Figure 23: Table showing requirements sorted using the MoSCoW method.

The first core requirement, ensuring that 85% of all the ships are detected, was more than sufficiently fulfilled, as both models achieved a recall score of approximately 100%. Additionally, the system was required to operate on optical imagery. This was successfully achieved, as the evaluation was conducted using optical satellite images, and the corresponding model demonstrated strong performance based on standard metrics. Another key requirement was the ability to determine the geographical location of each detected vessel. This was implemented successfully. The system was also able to evaluate whether a

detected ship could realistically be matched to an AIS broadcasting vessel, using average speed as a criterion. Regarding the next category, the first "Should" requirement stated that the system should be able to operate on data from Cyprus. This was achieved, as both the training and evaluation datasets were derived from the southern Cyprus maritime, and the system was thus optimized for this geographical region. The second requirement was that the system should reach an F1-score of at least 80%. This was more than met, with the optical model achieving an F1-score of 94.8% and the SAR model reaching 99.1%. The third requirement was that the system should not have high operating costs. During development and testing, all components were run at no cost using freely available tools and data. While incorporating real-time satellite and AIS data would introduce expenses, these would be the system's primary cost driver and remain relatively manageable. Finally, the system was also desired to have both optical and SAR imagery integration. However, even though a model was developed for SAR imagery, the system was only tested and optimized for optical imagery. This requirement was thus not achieved.

Of the optional "Could" requirements, three of the four requirements were implemented. Firstly, the system estimates the length of every ship, and uses that to also make a rough estimation of the classification. This requirement was thus achieved, even though the accuracy is lacking in some situations. Additionally, the system features an online user-friendly, interactive interface that visualizes all detections and their matched AIS vessels, including relevant data such as location, vessel length and distance. Next to that, also the 'future' trajectory of AIS broadcasting vessels is shown, which fulfills another "Could" requirement. However, the system was only tested in one specific region. As a result, the final "Could" requirement, application to different geographical regions, was not yet achieved.

7.5 Conclusion

Concluding, this evaluation shows that both deep learning models performed above expectations based on their performance metrics, with F1-scores exceeding 90%. This indicates strong overall detection capabilities. However, both models still show room for improvement in terms of bounding box precision, which affects other estimations such as vessel length. The system also effectively integrates these detections with AIS data, as it successfully links detected vessels to AIS-broadcasting ships in most cases. These matches are generally also judged correctly in terms of realisticness. Most vessels flagged as potentially unauthorized were found to be small and fast-moving vessels. However, in some cases, the system misclassified a realistic match as unrealistic, as it bases this decision only on average speed. Using additional parameters such as vessel heading and trajectory could improve these decisions. The current classification approach is also limited, as it relies only on estimated ship length, which itself can be inaccurate. For example, in several cases wave trails were included in the bounding box, leading to an overestimation of ship size which affected the reliability of classification results. Finally, the performance of the deep learning model trained on optical imagery was unexpectedly low during the real-world evaluation of the full system, despite achieving high scores in its individual evaluation. This drop in performance could be due to differences in lighting conditions or sea background in the evaluation imagery. However, it could also be a sign of overfitting. Additionally, the user study on the visualization of the findings provided valuable insights into the system's usability and potential. The visualization was positively received for its clarity, intuitiveness,

and practical relevance. However, the feedback also revealed several opportunities for improvement. Additionally, the participant group did not include experts in this research field.

In conclusion, while the system has proven its potential, the findings also highlight the importance of further refinement and testing before it can be deployed reliably in real-world maritime surveillance.

8. Discussion & Future work

This chapter discusses the results and limitations of the developed system and highlights areas that require further improvement. Additionally, it outlines potential directions for future research to further improve the system.

8.1 Discussion of results

The system was developed with the goal of detecting ships from satellite imagery and matching them with AIS data to assess vessel legitimacy. Based on the evaluation, these objectives were achieved for the most part, as most vessels were successfully detected, and AIS integration worked as intended in isolated or less crowded environments. Additionally, most matches were also accurately classified as authorized or unauthorized. Next to that, both deep learning models achieved high scores on standard metrics such as precision, recall, and F1-score. With F1-scores of 94.8% and 99.1% for optical and SAR respectively, the results are aligning well with the baseline performance in related literature, where values of 90% or higher are often achieved.

However, even though most of these initial aims have been achieved, several important limitations remain, as highlighted in the previous chapter during the evaluation. Firstly, while the deep learning models achieved very high scores on standard performance metrics, the model trained on optical imagery underperformed during the evaluation of the system as a whole, failing to detect vessels that were clearly visible. A likely explanation for this is the limited size and diversity of the training dataset. Since all annotations were created manually and the amount of imagery that could be acquired was constrained by the financial limitations of the project, the dataset may have included an overrepresentation of specific ship types or background conditions. This could have led to the model overfitting to certain visual features and performing poorly on images that differed even slightly from those in the training data. In addition, all bounding boxes were manually drawn, which may have introduced inaccuracies. One specific issue was the inclusion of wave trails behind ships in the annotated bounding boxes, which was especially a problem for smaller vessels. This led to overestimated vessel lengths, which in turn affected other components of the system. For example, these length estimates were also used as the only basis for predicting vessel classification, further reducing reliability. However, even when the length would have been estimated correctly all of the time, it still does not provide sufficient information to make a reliable prediction on the classification of a vessel. Parameters such as vessel speed, heading, and movement patterns could be used to improve the accuracy of classification in future versions.

Additionally, while two deep learning models were developed for this project, one trained on optical satellite imagery and another on SAR imagery, only the optical model was integrated into the full system. The SAR-based model was evaluated separately and not incorporated into the detection-matching pipeline due to time constraints, and was thus not tested in combination with AIS data or real-world scenarios.

Another significant limitation lies in how the system matches detected vessels to AIS-broadcasting ships. Currently, this process is based only on locational data, linking each detection to the nearest AIS broadcasting ship. While this method can work in isolated or sparsely populated maritime areas, it is not reliable in regions with many vessels close together. In such situations, the system could easily assign detections to the wrong AIS

track. A more advanced approach would involve using multiple parameters, such as vessel size, classification and trajectory to improve match accuracy in complex scenarios.

Furthermore, the current method for determining whether a match is realistic is simplistic, as it relies only on the calculated average speed between the detection and the AIS signal. This can result in false negatives, especially in cases of smaller faster ships. Incorporating additional factors into the decision-making process, like trajectory and vessel length could therefore significantly improve the accuracy of this process as well.

Lastly, another limitation of the system is that all data processing is performed manually and offline. No real-time or automated data retrieval is integrated into the system. While this was acceptable for the scope of this prototype, transitioning the system into a tool that can be used in real-world scenarios would require live access to AIS data and frequent satellite imagery updates. This was not feasible in the current project due to the high cost of such services, which was beyond the available budget.

8.2 Future work

Based on the limitations of the project, there are some interesting points for future research. For example, future work could focus on fully incorporating the SAR-based deep learning model into the system, or even creating a new deep learning model suitable for both optical and SAR imagery. Making the system suitable for SAR imagery could significantly improve performance, especially under poor lighting or weather conditions. This would also enable the system to operate at all times, rather than being limited to clear weather during daylight hours, which is essential for continuous maritime monitoring.

Additionally, future research should focus on creating larger and more diverse datasets that include a wide range of vessel types and varying environmental conditions. This would reduce the risk of model overfitting and improve detection accuracy. Furthermore, incorporating imagery from different geographical regions would enhance the model's applicability, which is essential for effective maritime monitoring on a global scale.

Next to that, future research should focus on incorporating more logic to the process of matching detected vessels with AIS-broadcasting ships. Instead of relying solely on proximity, the matching algorithm should take into account factors such as vessel heading, trajectory, and estimated vessel dimensions. Incorporating these parameters would help distinguish between closely situated ships and reduce the risk of incorrect matches, especially in busy maritime areas. Similarly, the criteria for labeling a match as realistic or unrealistic should be improved beyond simple average speed calculations. Evaluating the alignment of vessel trajectories over time, changes in speed, and other behavioral patterns could provide a more accurate assessment of match validity, which could reduce false positives and negatives.

Finally, enabling the system to operate in real time by integrating live satellite imagery and real-time AIS data streams would significantly enhance its suitability for real-world applications. At the same time, it is important that the system remains cost-effective, to ensure its accessibility for a wide range of organizations and countries, including those with limited financial resources. This would help to contribute to more effective and inclusive maritime monitoring worldwide.

9. Conclusion

At the start of this research, the main research question was formulated as: “How can an effective maritime monitoring system be developed using earth observation methods, Automatic Identification System data and deep learning technologies?” Throughout the study, this question was addressed by conducting a literature research and answering three supporting subquestions. The first subquestion was: “What remote sensing technology is most optimal for ship detection?” Based on the state-of-the-art review, both SAR and optical imaging proved to be suitable technologies for ship detection. Since each has its own advantages and disadvantages, they are considered more interchangeable than complementary. SAR has the advantage of being usable under all weather and lighting conditions, while optical imagery typically offers higher spatial resolution and requires less preprocessing.

The second sub question asked: “Which deep learning model is best suited for ship detection on satellite imagery?” After reviewing existing research, it became clear that YOLO deep learning models are the most commonly used and effective in this field.

Finally, the third sub question was: “How can AIS data be used to identify unauthorized vessels?” The literature showed that AIS data is broadcasted by ships to share their location and other information to promote marine safety. However, some ships turn off their AIS transponders, which is illegal and may indicate attempts to hide illicit activities. Therefore, if a ship is visible in satellite imagery but not present in AIS data, it may be considered as possibly unauthorized.

Based on this information, the concept for the system was developed. The core component was decided to be a YOLO deep learning model that processes satellite imagery to detect ships and determine their locations. These detections are then matched to the closest AIS broadcasting vessels from the same time and region. The plausibility of each match is assessed by calculating the average speed the vessel would need to travel for the match to be valid. Finally, the findings are displayed on a visualization.

To realize this system, datasets for both SAR and optical imagery were manually created, and two YOLOv11 models were trained accordingly. Both models demonstrated promising performance, achieving F1-scores of 99.1% for SAR and 94.8% for optical imagery. A Python script was developed to automate ship detection, geolocation, and AIS matching, including the evaluation of match realism based on vessel speed. Finally, the results are visualized in the form of an interactive webpage, displaying a map with detected ships and their AIS matches connected by markers, alongside additional details such as vessel length, coordinates, distances, trajectories, and images of the detected ships.

The system successfully matched most detected ships with AIS data, with realistic outcomes in the majority of cases. However, some errors seem to have occurred due to limitations in using only average speed and location as a decision factor. The classification method also showed weaknesses, as it relied only on estimated ship length, which was not always accurate itself. This highlights the need for more robust input parameters such as heading and trajectory for improved accuracy. Additionally, the user study evaluating the visualization component offered important perspectives on how the system is perceived in terms of usability. Participants generally responded positively, highlighting the interface as clear, user-friendly, and practically useful. At the same time, the feedback also identified a number of areas where the tool could be enhanced to further improve its functionality and user experience.

In summary, the system effectively answers the main research question by detecting ships on satellite imagery with a deep learning model, and comparing it with AIS data to identify possibly unauthorized vessels. While there are limitations and areas for improvement, the results show a promising foundation for creating an effective system using earth observation and deep learning technologies.

Reference list

[1] M. Bakirci, “Advanced Ship Detection and Ocean Monitoring with Satellite Imagery and Deep Learning for Marine Science Applications,” *Regional Studies in Marine Science*, p. 103975, Dec. 2024, doi: 10.1016/j.rsma.2024.103975.
[2] X. Yang, X. Zhang, N. Wang and X. Gao, "A Robust One-Stage Detector for Multiscale Ship Detection With Complex Background in Massive SAR Images," in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-12, 2022, Art no. 5217712, doi: 10.1109/TGRS.2021.3128060.
[3] H. Guo, X. Yang, N. Wang, B. Song and X. Gao, "A Rotational Libra R-CNN Method for Ship Detection," in IEEE Transactions on Geoscience and Remote Sensing, vol. 58, no. 8, pp. 5772-5781, Aug. 2020, doi: 10.1109/TGRS.2020.2969979.
[4] G.-S. Xia et al., "DOTa: A large-scale dataset for object detection in aerial images", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 3974-3983, Jun. 2018.
[5] B. Snapir, T. Waine, and L. Biermann, “Maritime Vessel Classification to Monitor Fisheries with SAR: Demonstration in the North Sea,” Remote Sensing, vol. 11, no. 3, p. 353, Feb. 2019, doi: 10.3390/rs11030353.
[6] “AIS (Automatic Identification System) overview” shipping.nato.int. https://shipping.nato.int/nsc/operations/news/2021/ais-automatic-identification-system-overview
[7] Nai, “How AIS works and what it does.” https://www.nautinst.org/resources-page/how-ais-works-and-what-it-does.html
[8] Spire, “Spire : Global Data and Analytics,” *Spire : Global Data and Analytics*, Feb. 07, 2025. https://spire.com/
[9] “AccessAIS,” *NOAA: Digital Coast*. https://coast.noaa.gov/digitalcoast/tools/ais.html
[10] A. Tritsarolis, Y. Kontoulis, and Y. Theodoridis, “The Piraeus AIS dataset for large-scale maritime data analytics,” *Data in Brief*, vol. 40, p. 107782, Jan. 2022, doi: 10.1016/j.dib.2021.107782.
[11] X. Zhang, X. Yang, D. Yang, F. Wang and X. Gao, "A Universal Ship Detection Method With Domain-Invariant Representations," in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-11, 2022, Art no. 5629311, doi: 10.1109/TGRS.2022.3200957
[12] H. Guo, X. Yang, N. Wang, and X. Gao, “A CenterNet++ model for ship detection in SAR images,” *Pattern Recognition*, vol. 112, p. 107787, Dec. 2020, doi: 10.1016/j.patcog.2020.107787.
[13] K. Patel, C. Bhatt, and P. L. Mazzeo, “Improved Ship Detection Algorithm from Satellite Images Using YOLOv7 and Graph Neural Network,” Algorithms, vol. 15, no. 12, p. 473, Dec. 2022, doi: 10.3390/a15120473.
[14] T. Zhang, X. Zhang, J. Shi, and S. Wei, “HyperLi-Net: A hyper-light deep learning network for high-accurate and high-speed ship detection from synthetic aperture radar imagery,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 123–153, Jul. 2020, doi: 10.1016/j.isprsjprs.2020.05.016.
[15] “Illegal, unreported, and unregulated fishing causes and effects” World Wildlife Fund. https://www.worldwildlife.org/threats/illegal-fishing
[16] Windward, “What is Illegal Fishing?,” Windward, Jun. 10, 2024. https://windward.ai/glossary/what-is-illegal-fishing/
[17] S. Zhang, R. Wu, K. Xu, J. Wang, and W. Sun, “R-CNN-Based Ship Detection from High Resolution Remote Sensing Imagery,” *Remote Sensing*, vol. 11, no. 6, p. 631, Mar. 2019, doi: 10.3390/rs11060631.
[18] C. Zhao, X. Fu, J. Dong, S. Cao and C. Zhang, "MLC-Net: A Robust SAR Ship Detector With Speckle Noise and Multiscale Targets," in IEEE Journal of Selected Topics in Applied

Earth Observations and Remote Sensing, vol. 17, pp. 19260-19273, 2024, doi: 10.1109/JSTARS.2024.3401723
[19] A.-J. Gallego, A. Pertusa, and P. Gil, "Automatic ship classification from optical aerial images with convolutional neural networks," *Remote Sens.*, vol. 10, no. 4, 2018, doi: 10.3390/rs10040511.
[20] Shunjun Wei ; Xiangfeng Zeng ; Qizhe Qu ; Mou Wang ; Hao Su ; Jun Shi. HRSID: A High-Resolution SAR Images Dataset for Ship Detection and Instance Segmentation . IEEE Access
[21] S. Agrawal and G. B. Khairnar, "A COMPARATIVE ASSESSMENT OF REMOTE SENSING IMAGING TECHNIQUES: OPTICAL, SAR AND LIDAR," The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences/International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLII-5/W3, pp. 1–6, Dec. 2019, doi: 10.5194/isprs-archives-xlii-5-w3-1-2019.
[22] S. Gui, S. Song, R. Qin, and Y. Tang, "Remote Sensing Object Detection in the Deep Learning Era—A Review," Remote Sensing, vol. 16, no. 2, p. 327, Jan. 2024, doi: 10.3390/rs16020327.
[23] T. Singha, T. Babu, R. R. Nair, and P. Duraisamy, "Ship detection in Synthetic Aperture Radar Imagery: An active contour model approach in Computer Vision Deep Learning," ELSEVIER B.V., journal-article, 2024. doi: 10.1016/j.procs.2024.04.170.
[24] A.-J. Gallego, A. Pertusa, and P. Gil, "Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks," *Remote Sensing*, vol. 10, no. 4, p. 511, Mar. 2018, doi: 10.3390/rs10040511.
[25] B. Karbouj, G. A. Topalian-Rivas, and J. Krüger, "Comparative performance evaluation of One-Stage and Two-Stage object detectors for screw head detection and classification in disassembly processes," *Procedia CIRP*, vol. 122, pp. 527–532, Jan. 2024, doi: 10.1016/j.procir.2024.01.077.
[26] Planet Labs PBC, "Planet imagery and data." [Online]. Available: https://www.planet.com/
[27] MarineTraffic, "Live vessel tracking and maritime data." [Online]. Available: https://www.marinetraffic.com/
[28] "PS | Pervasive Systems group | University of Twente," Universiteit Twente. https://www.utwente.nl/en/eemcs/ps/
[29] "Open Source data Labeling | Label Studio," Label Studio. https://labelstud.io/
[30] G. Jocher, J. Qiu, and A. Chaurasia, Ultralytics YOLO, Ultralytics, Jan. 10, 2023. [Software]. Available: https://github.com/ultralytics/ultralytics
[31] "Earth Data Analytics Online Certificate," Earth Data Science - Earth Lab, Feb. 05, 2018. https://www.earthdatascience.org/courses/use-data-open-source-python/intro-raster-data-python/fundamentals-raster-data/intro-to-the-geotiff-file-format/
[32] "Rasterio: access to geospatial raster data — rasterio 1.4.3 documentation." https://rasterio.readthedocs.io/en/stable/index.html
[33] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 51–56. [Online]. Available: https://doi.org/10.25080/Majora-92bf1922-00a
[34] Ministerie van Defensie, "Coordinate reference systems for positioning at sea," Hydrography | Defensie.nl, Oct. 16, 2018. https://english.defensie.nl/topics/hydrography/coordinate-systems-at-sea/coordinate-reference-systems-for-positioning-at-sea

[35] S. N. Hagberg, "Folium: Python Data. Leaflet.js Maps," GitHub repository, 2014. [Online]. Available: https://github.com/python-visualization/folium.
[36] European Space Agency (ESA), *Copernicus Open Access Hub*. [Online]. Available: https://scihub.copernicus.eu
[37] R. Leaper, “The role of slower vessel speeds in reducing greenhouse gas emissions, underwater noise and collision risk to whales,” Frontiers in Marine Science, vol. 6, Aug. 2019, doi: 10.3389/fmars.2019.00505.
[38] Mixed Migration Centre, “Cyprus migration dilemma: Hardline measures, regional conflict and rising pressures,” Sep. 2024. [Online]. Available: https://mixedmigration.org/cyprus-migration-dilemma-hardline-measures-regional-conflict-and-rising-pressures/.
[39] OpenAI, ChatGPT. [Online]. Available: https://chat.openai.com/. Accessed: Mar.–Jul. 2025. See Appendix F for additional context.

Appendix A - Literature Matrix

Paper title	Aims	Remote sensing technology	Spatial resolution	POI	Geographical area
"Advanced ship detection and ocean monitoring with satellite imagery and deep learning for marine science applications" [1]	Testing the YOLOv9 model for detecting ships in maritime environments, compared to other YOLO models.	Optical	From 15 meters per pixel in remote areas to 0.3 meters per pixel in more inhabited areas (coastal regions)	Ships in general (no specific type). Both at open sea and in harbours.	Coastal regions in Texas, China, Japan, Brazil, Northern Africa, Turkey, Greece, Indonesia and Malaysia, India and around the Persian gulf. Additionally, open sea coverage of the Mediterranean, the South China Sea and the Bay of Bengal.
"A Robust One-Stage Detector for Multiscale Ship Detection With Complex Background in Massive SAR Images" [2]	Develop a high-performance ship detection algorithm for SAR images which can handle complex backgrounds and small ships as well. Several different backbones are tested.	SAR	Not specifically specified.	Ships in general (no specific type). Both at open sea and in harbours.	Not specified.
"A Rotational Libra R-CNN Method for Ship Detection" [3]	The goal is to develop a rotational R-CNN model that enhances ship detection accuracy,	Optical	Not specified. However, since this study is from 2020, only DOTA	Ships at sea, in harbours, and on land (for example in storage on a certain	Not restricted to a single geographic area.

	addressing challenges like varying ship sizes and dense distributions. The model will be tested on the DOTA dataset [4] to evaluate its effectiveness.		versions of up to v1.5 could have been used. For this version, objects as small as 10 pixels are labelled. [4]	terrain).	
"Maritime Vessel Classification to Monitor Fisheries with SAR: Demonstration in the North Sea" [5]	To develop a RF (random forest) classifier to differentiate 'normal' ships and fishing ships using AIS data, and applying this to SAR imagery for monitoring fishing activity in the North Sea.	SAR	Imagery of S-1 has been used, which has a spatial resolution of 20 meters according to the paper.	Fishing ships (and distinguishing them from other ships)	The North Sea.
"A Universal Ship Detection Method With Domain-Invariant Representations" [11]	Creating a universal ship detector able to operate in multiple domains.	SAR and Optical	6 different datasets were used, from all kinds of different satellites, and no specific spatial resolution has been given.	Ships in general (both at sea and in harbors).	Not limited to a specific region.
"A CenterNet++ model for ship detection in SAR images" [12]	Improve the existing CenterNet model to create the new CenterNet++, optimized for ship detection in SAR imagery.	SAR	3 different datasets have been used from different satellites, and no specific spatial resolution was provided.	Ships in general (both at sea and in harbors).	Not limited to a specific region.

"Improved Ship Detection Algorithm from Satellite Images Using YOLOv7 and Graph Neural Network" [13]	Combining YOLOv7 and a Graph Neural Network (GNN) to create a ship detection algorithm.	SAR	0.5 - 3 meters (HRSID dataset was used)	Ships in general (both at sea and in harbors).	Not limited to a specific region.
"HyperLi-Net: A hyper-light deep learning network for high-accurate and high-speed ship detection from synthetic aperture radar imagery" [14]	Creating a new 'HyperLi-Net' model using 5 external modules, in order to get high-accurate and high-speed ship detection on SAR imagery	SAR	Three different datasets were used, all with a wide range of spatial resolutions from 1 to 25 meters.	Ships in general (both at sea and in harbors).	Waters around India, China, South Korea and Japan.
"R-CNN-Based Ship Detection from High Resolution Remote Sensing Imagery" [17]	Coming up with a ship detection method effective on small and gathering ships using high resolution satellite imagery.	Optical	1 meter	Ships in general, also on inland waters (for example rivers)	Yangtze river (China)
"MLC-Net: A Robust SAR Ship Detector With Speckle Noise and Multiscale Targets" [18]	Improving ship detection on SAR imagery with a new model, addressing challenges like speckle noise and multiscale targets	SAR	1 - 15 meters	Ships in general	Not limited to a specific region.
"Ship Detection in Synthetic Aperture Radar Imagery: An Active Contour Model Approach in Computer Vision Deep Learning" [23]	Improving ship detection on SAR imagery by proposing new method, using 6 different models (including YOLOv8)	SAR	1 - 15 meters	Ships in general	Not limited to a specific region.

"Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks" [24]	Goal is to present a ship detection method for optical imagery, using CNNs and kNNs	Optical	Not specified for all datasets. However, also tested with a dataset with spatial resolution 0.08 - 2 meters.	Ships in general	Not limited to a specific region.

Paper name	**Time /year**	**Detection/classification problem**	**ML technique**	**Performance metrics**	**AI model score**	**Lessons learned**	**General comments**
"Advanced ship detection and ocean monitoring with satellite imagery and deep learning for marine science applications" [1]	2024	For smaller ships, it was often seen as several objects instead of 1. This was solved using NMS (non-maximum suppression)	CNN (deep learning)	Precision, Recall, mAP (mean average precision), F1 score, inference time.	Precision: 0.946 Recall: 0.872 mAP: 0.959 F1 score: 0.908 Inference time: 6.28 Scores for the other models were also given, but only the scores of the best model (YOLOv9) are listed above.	Smaller ships still remained a challenge. All images were turned into 480 x 480 images, and while for ships between 250-500 pixels the standard deviation was 0.024, this was 0.13 for ships smaller than 250 pixels.	Paper focuses mostly on comparing the AI model with its predecessors, and for example does not compare results with AIS data.
"A Robust One-Stage Detector for Multiscale Ship Detection With	2021	If two ships are very close to each other with a complex background, it is	CNN (deep learning)	Precision, recall, F1 scores and average precision at	Precision: 0.927 Recall: 0.881 F1 score: 0.93 AP50: 0.927	Image super-resolution construction networks can increase resolution	Paper is about 1 model with different backbones, tested on several datasets, from

Complex Background in Massive SAR Images" [2]		often seen as one ship instead of two separate ships.		IoU 0.5.	These are the scores for the model using Darknet-53 as a backbone on the HRSID, the dataset with the most complex backgrounds/small ships.	in order to better identify two ships very close to each other.	which HRSID is said to be most complicated and thus shows effectiveness of the method the best.
"A Rotational Libra R-CNN Method for Ship Detection" [3]	2020	Most mistakes were made when trying to detect ships on land, since it would often confuse cars or small houses for boats. Additionally, uniquely shaped boats were not always detected since they are not present in the dataset a lot.	R-CNN (deep learning)	mAP	mAP: 75.62% This score was the highest of all models it was compared with, for example YOLOv2 and RetinaNet.	It might be important to check which ships are most used for (illegal) fishing in certain regions that the model should be used for, in order not to miss any ships that are common for that region, while it might not be common in the rest of the world. (e.g. outrigger boats)	Most errors were for ships on land, which is not relevant for us. However, the DOTA dataset could be a good one to use to train the model. [4]
"Maritime Vessel Classification to Monitor Fisheries with SAR: Demonstration in the North Sea" [5]	2019	Places with a lot of marine traffic (e.g Strait of Dover) made it more difficult to correctly identify fishing vessels. More	Random Forest	Overall classification accuracy and precision	Classification accuracy: 91% Precision: 58% The accuracy shows that 91% of all predictions (fishing	This research shows it would be possible to also check if a detected vessel is likely to be a fishing vessel or not, based on	SAR imagery is used in a different way than in our research here, since a SUMO detector is used to detect the vessels

		remote areas could potentially have better results.			vessel or no fishing vessel) were correct. However, the precision tells that just 58% of the vessels predicted to be fishing vessels were actually fishing vessels.	data like position, length and time. This could potentially be used to distinguish between illegal fishing and 'non-illegal' vessels, to decrease the amount of false alarms.	and provides data like length and position of the vessel. The study is thus more about using position and length (among more) to distinguish fishing boats from normal vessels.
"A Universal Ship Detection Method With Domain-Invariant Representations" [11]	2022	For SAR, the complex background and noise remain a challenge. For optical images, the varying size of ships is the biggest obstacle.	A multilevel domain classification network (MDCN), so deep learning.	AP50	Using all kinds of different datasets (SAR & optical) with different resolutions, the average AP50 of the model was 93.33%	A domain-centric cut-paste module (DCM) was added, which is a way of data augmentation, and increased the average precision by 0.7%.	This paper gives comparisons of the model performance for several datasets, which gives insights on the change in performance for different remote sensing technologies.
"A CenterNet++ model for ship detection in SAR images" [12]	2020	Ships close to each other are detected as one, and sometimes one ship gets detected twice.	Deep learning, specifically CenterNet(++), which is a type of CNN.	Precision, Recall, F1 and mAP.	Precision: 83.5% Recall: 97.6% F1: 90.0% mAP: 95.4% These are the scores for the SAR-ship dataset, which gave the best results.	Instance segmentation could resolve the issue of detecting multiple ships as one.	In this paper it is thoroughly explained how they specialized a model for SAR imagery, which could be useful when struggling with this.

"Improved Ship Detection Algorithm from Satellite Images Using YOLOv7 and Graph Neural Network" [13]	2022	There was no mention of any big obstacles encountered. However, if the dataset's amount of pictures would have been enhanced, it could have led to better performance according to the researchers.	Deep learning, specifically YOLOv7 combined with a GNN.	Accuracy	The highest achieved accuracy was 93.4%	The paper states that the devices used did not have outstanding specifications, which shows it should definitely be possible to get good results even with the technical limitations of the project.	Paper does not compare the model with other deep learning models, so this is a bit of a downside.
"HyperLi-Net: A hyper-light deep learning network for high-accurate and high-speed ship detection from synthetic aperture radar imagery" [14]	2020	Ships too close to each other, for example in harbors, would not get detected. Also, sometimes the background gets detected as a ship.	Newly developed deep learning model, called HyperLi-Net.	Detection probability, missed-detecti on probability, false-alarm probability, recall, precision, mAP.	Detection probability: 96.74% Missed-detection probability: 3.26% False-alarm probability: 9.64% Recall: 96.74% Precision: 90.36% mAP: 96.08% These were the results for the SSDD dataset, which gave the best results.	The model gave similar results for an entire new dataset on pictures from a different location, which might indicate location should not matter too much (depending on the model of course).	Process of creating the model including all the modules is extensively described in the paper.
"R-CNN-Based Ship Detection from High Resolution Remote Sensing	2019	Shadows were sometimes classified as a ship, but this was fixed by using NMS	Deep learning, R-CNN.	Stage recall, combine recall, precision.	Precision: 95.79 Stage recall: 96.46 Combine recall: 91.66	Changing the confidence threshold increased performance for	In their method they also remove the non-water parts of the pictures during

Imagery" [17]		(Non Maximum Suppression)				the more blurry images.	preprocessing, to only keep the water with the potential ships and avoid seeing things on land as ships.
"MLC-Net: A Robust SAR Ship Detector With Speckle Noise and Multiscale Targets" [18]	2024	Most models struggle with the varying size of ships, as well as noise in SAR imagery. This model is optimized for this.	Deep learning, specifically MLC-Net, a type of CNN.	AP, F1, Recall, Precision	AP: 99.16 F1: 97.00 Recall: 97.8 Precision: 96.22 These are the scores on the SSDD dataset.	Three architectural modules were used in order to optimize the model for noisy images and ships both large and small.	Several versions of Yolo (v7/v8) also scored very high.
"Ship Detection in Synthetic Aperture Radar Imagery: An Active Contour Model Approach in Computer Vision Deep Learning" [23]	2024	Noise in the images and unwanted reflections of coastlines are often a problem, and the model is optimized to not be affected by this.	Deep learning, YOLOv8 and compares it to other models.	Precision, recall and mAP score	Precision: 94.5 Recall: 94.1 mAP: 98.7		Compared with other versions of YOLO, Tensorflow2 and Detectron2 models, with YOLOv8 giving best results.
"Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks" [24]	2018	Ship detection in optical imagery is a challenge because of things like waves, rocks and other things also being present in the pictures.	Deep learning through a CNN, in combination with a KNN classifier.	F1 score	F1 score: 99.8	Makes use of the MASATI dataset, which could be useful for training models. [19]	The method proposed in the paper does not include location identification, just detecting if a ship is present or not.

Appendix B - Model training code

```
from ultralytics import YOLO

# Dataset directory contains the following structure:
# ├── images/
# │   ├── train/
# │   └── val/
# ├── labels/
# │   ├── train/
# │   └── val/
# └── dataset.yaml

# Path to dataset.yaml
yaml_path = "dataset.yaml"

# Load YOLOv11-OBB model
model = YOLO("yolo11n-obb.yaml")

# Train the model
model.train(
   data=yaml_path,
   epochs=100,
   imgsz=512,
   device="cpu",
)

# Validate the model
model.val()

# Save the trained model
model.save("Ship_detection_model.pt")
```

Appendix C: Main Python script

```
from PIL import Image
import pandas as pd
from ultralytics import YOLO
import rasterio
import pyproj
from datetime import datetime, timedelta
from geopy.distance import geodesic
from rasterio.warp import transform
import folium
import webbrowser
import os
import glob
import re
import uuid
import numpy as np


def main(tif_path, ais_file, meta_file, delta_time,
all_ais_files=None,
              tile_size=(512, 512), stride=(256, 256),
dedup_threshold_m=30):

   # Main function that coordinates ship detection and AIS
matching
   # by calling functions, and returns the combined results.

   # Loading full satellite image
   img = Image.open(tif_path)
   width, height = img.size

   # Read metadata to get timestamp of acquiring satellite image
   meta_df = pd.read_json(meta_file)
   acquired_iso = meta_df['properties']['acquired']
   acquired_dt = datetime.strptime(acquired_iso,
"%Y-%m-%dT%H:%M:%S.%fZ")
   formatted_date = acquired_dt.strftime("%Y-%m-%d")

   # Setting tile size and stride (overlap) for the cut-outs of
the satellite image
   tile_w, tile_h = tile_size
   stride_x, stride_y = stride

   all_global_boxes = []  # Store all bounding boxes in global
image coordinates

   # Iterate over image with overlapping tiles
```

```
    for i in range(0, width, stride_x):
        if i + tile_w > width:
            i = width - tile_w  # Adjust to stay within the image
bounds
        for j in range(0, height, stride_y):
            if j + tile_h > height:
                j = height - tile_h  # Adjust to stay within the
image bounds

            # Crop tile from full image
            tile = img.crop((i, j, i + tile_w, j + tile_h))
            ship_boxes = ship_detection(tile)  # Send cropped image
to ship_detection function and get box coordinates

            # Convert local box coordinates to global image
coordinates
            for box_coords in ship_boxes:
                global_box = [[x + i, y + j] for x, y in
box_coords]
                all_global_boxes.append(global_box)

            # Stop when bottom-right corner is reached
            if i + tile_w >= width and j + tile_h >= height:
                break

    # Send TIFF directory and all bounding boxes (in global pixel
coordinates) to geo_location function
    # to get geographical coordinates.
    all_geo = geo_location(tif_path, all_global_boxes)

    # Remove duplicate detections (same ship seen in multiple
tiles) and save info of unique detections
    unique_detections = []
    for idx, geo in enumerate(all_geo):
        center = geo['center']
        length = geo['length']
        # Only keep if no previous detection is within threshold
distance
        if not any(geodesic(center, det['center']).meters <
dedup_threshold_m for det in unique_detections):
            unique_detections.append({'center': center, 'length':
length, 'global_box': all_global_boxes[idx]})

    # Send unique detections to AIS matching function to associate
with broadcasting vessel positions
    if all_ais_files:
        matched_results = ais_check(tif_path, unique_detections,
ais_file, acquired_dt, delta_time, all_ais_files)
```

```
    else:
        matched_results = ais_check(tif_path, unique_detections,
ais_file, acquired_dt, delta_time)

    # Return results in dictionary by date
    return {formatted_date: matched_results}


def ship_detection(image):
    # Takes image and detects ships using a YOLOv11-OBB model.
Returns bounding box coordinates of detected ships.

    # Load model and get detection results
    model = YOLO("YOLO_model.pt")
    results = model(image, imgsz=512)

    obb = results[0].obb  # Get oriented bounding boxes
    ship_boxes = []

    # Convert bounding boxes to four-corner format
    for i in range(len(obb)):
        corners = obb.xyxy[i].tolist()
        x1, y1, x2, y2 = corners
        ship_boxes.append([[x1, y1], [x2, y1], [x2, y2], [x1, y2]])

    return ship_boxes


def geo_location(tif, ship_boxes):
    # Function takes TIF image and pixel coordinates of ships, and
turns these into
    # geographical coordinates, using Rasterio

    dataset = rasterio.open(tif)  # Open raster file
    transformer = pyproj.Transformer.from_crs(dataset.crs,
"EPSG:4326", always_xy=True)  # Reproject to WGS84

    geo_boxes = []
    for box in ship_boxes:
        geo_box = []
        for x, y in box:
            lon, lat = dataset.xy(int(y), int(x))  # Convert pixel
to geographical coordinates
            lon, lat = transformer.transform(lon, lat)  # Transform
to right format (same as used in AIS data)
            geo_box.append((lat, lon))

        # Estimate maximum side length of box
```

```
        d1 = geodesic(geo_box[0], geo_box[1]).meters
        d2 = geodesic(geo_box[1], geo_box[2]).meters
        max_d = max(d1, d2)

        # Calculate center of box
        center_lat = sum(pt[0] for pt in geo_box) / 4
        center_lon = sum(pt[1] for pt in geo_box) / 4
        geo_boxes.append({'center': (center_lat, center_lon),
'length': max_d})

    return geo_boxes


def extract_time_from_filename(filename):
    # Function that extract the timestamp of AIS data from the file
name
    # Example filename:
MarineTraffic_Vessels_Export_2025-05-16_8.48.csv
    match = re.search(r'_(\d+\.\d+)\.csv$', filename)
    if match:
        return float(match.group(1))
    else:
        return float('inf')  # If no time, put at end when sorting


def ais_check(tif_path, detections, ais_file, time_satellite,
delta_time_minutes, all_ais_files=None):

    # This function matches satellite-based ship detections to AIS
vessel broadcasts based on geographic location and
    # time. It returns a list of matched detections for a given
satellite image including information on the detected
    # vessel as well as the AIS broadcasting vessel it was matched
with.

    # Load AIS data from CSV into a dataframe
    ais_df = pd.read_csv(ais_file, delimiter=';')

    # Define time window for filtering AIS entries
    time_window = timedelta(minutes=delta_time_minutes)

    # Helper function to convert AIS coordinates (stored as strings
without decimals) to float
    def fix_coord(val):
        val = val.replace('.', '')  # Remove existing decimal if
any
        return float(val[:2] + '.' + val[2:])  # Insert decimal
after 2 digits
```

```
    # Apply coordinate fixer function to Latitude and Longitude
columns
    ais_df['Latitude'] =
ais_df['Latitude'].astype(str).apply(fix_coord)
    ais_df['Longitude'] =
ais_df['Longitude'].astype(str).apply(fix_coord)

    # Format AIS timestamps
    try:
        ais_df['Parsed Time'] = pd.to_datetime(ais_df['Time Of
Latest Position'], format="%d-%m-%Y %H:%M")
    except Exception as e:
        print(f"Error parsing AIS times: {e}")
        return []

    # Filter AIS records to those within the specified time window
around the satellite image acquisition time
    time_mask = abs(ais_df['Parsed Time'] - time_satellite) <=
time_window
    filtered_ais = ais_df[time_mask]

    # If no AIS entries found in the time window, return empty list
    if filtered_ais.empty:
        print(f"No AIS entries within +/- {delta_time_minutes}
minutes of satellite acquisition time.")
        return []

    # Dictionary to hold closest AIS match for each detection
    detection_best_matches = {}

    # For each AIS record, find the closest ship detection using
geodesic
    for ais_idx, ais_row in filtered_ais.iterrows():
        ais_coord = (ais_row['Latitude'], ais_row['Longitude'])

        closest_det_idx = None
        closest_dist = float('inf')

        for det_idx, det in enumerate(detections):
            det_coord = det['center']
            dist = geodesic(ais_coord, det_coord).meters

            # Keep track of closest detection to this AIS point
            if dist < closest_dist:
                closest_dist = dist
                closest_det_idx = det_idx
```

```
        # If this AIS point is closer to the detection than ANY
previous AIS point, store it
        if closest_det_idx is not None:
            if closest_det_idx in detection_best_matches:
                if closest_dist <
detection_best_matches[closest_det_idx]['distance']:
                    detection_best_matches[closest_det_idx] = {
                        'ais_idx': ais_idx,
                        'distance': closest_dist
                    }
            else:
                detection_best_matches[closest_det_idx] = {
                    'ais_idx': ais_idx,
                    'distance': closest_dist
                }

    # Assemble final list of matched detection and AIS pairs
    matched_results = []
    for det_idx, match_info in detection_best_matches.items():
        ais_idx = match_info['ais_idx']
        dist = match_info['distance']
        ais_row = filtered_ais.loc[ais_idx]
        det = detections[det_idx]

        # Calculate time difference and average speed estimate for
the match to be true
        time_diff_sec = abs((ais_row['Parsed Time'] -
time_satellite).total_seconds())
        time_diff_min = time_diff_sec / 60
        avg_speed = (dist / 1000) / (time_diff_min / 60) if
time_diff_min > 0 else 0

        # Classify whether the match seems realistic based on speed
        status = "Realistic" if avg_speed <= 45 else "Unlikely;
possible unauthorized vessel"

        # Store all information of the match
        matched_results.append({
            'sat': det['center'],
            'ais': (ais_row['Latitude'], ais_row['Longitude']),
            'name': ais_row['Vessel Name'],
            'length': round(det['length'], 1),
            'classification': ais_row['Vessel Type - Generic'],
            'distance': round(dist, 1),
            'delta_time': round(time_diff_min, 1),
            'speed': round(avg_speed, 1),
            'status': status,
```

```
            'time_satellite': time_satellite.strftime('%Y-%m-%d
%H:%M:%S'),
            'time_ais': ais_row['Parsed Time'].strftime('%Y-%m-%d
%H:%M:%S'),
            'tif_path': tif_path,
            'image_path': None  # To be filled in later
        })

    # Look for later AIS positions of matched vessels in order to
create a trajectory
    if all_ais_files is not None:
        for match in matched_results:
            vessel_name = match['name']
            later_positions = []

            for file_path in all_ais_files:
                if file_path == ais_file:
                    continue  # Skip current AIS file to avoid
double information

                # Try to load AIS file
                try:
                    ais_df_later = pd.read_csv(file_path,
delimiter=';')
                except Exception as e:
                    print(f"Error reading AIS file {file_path}:
{e}")
                    continue

                # Fix coordinates
                ais_df_later['Latitude'] =
ais_df_later['Latitude'].astype(str).apply(fix_coord)
                ais_df_later['Longitude'] =
ais_df_later['Longitude'].astype(str).apply(fix_coord)

                # Format time
                try:
                    ais_df_later['Parsed Time'] = pd.to_datetime(
                        ais_df_later['Time Of Latest Position'],
format="%d-%m-%Y %H:%M")
                except Exception as e:
                    print(f"Error parsing AIS times in {file_path}:
{e}")
                    continue

                # Filter entries matching this vessel and occurring
after the satellite time
                ais_ship_later = ais_df_later[
```

```
                    (ais_df_later['Vessel Name'] == vessel_name) &
                    (ais_df_later['Parsed Time'] > time_satellite)
                ]

                # Store each later position
                for _, row in ais_ship_later.iterrows():
                    later_positions.append({
                        'lat': row['Latitude'],
                        'lon': row['Longitude'],
                        'time': row['Parsed Time']
                    })

            # If later positions are found, add them to the
dictionary with the information on the matches.
            if later_positions:
                match['later_ais_positions'] = later_positions

    return matched_results


def add_cropped_images(detections_by_date):

    # This function crops satellite images around each detected
ship location and saves the cropped images.

    for date, detections in detections_by_date.items():
        print(f"Processing date: {date}, number of detections:
{len(detections)}")

        # Create output folder for this date's cropped images
        output_dir = os.path.join("detections", date)
        os.makedirs(output_dir, exist_ok=True)

        for i, det in enumerate(detections):
            tif_path = det['tif_path']
            print(f"  Detection {i}: Using tif {tif_path}")

            try:
                # Open satellite image using rasterio
                with rasterio.open(tif_path) as dataset:
                    sat_lat, sat_lon = det['sat']

                    # Transform detection lat/lon to raster CRS
                    raster_crs = dataset.crs
                    x, y = transform('EPSG:4326', raster_crs,
[sat_lon], [sat_lat])
                    x = x[0]
                    y = y[0]
```

```
                    # Convert coordinates to pixel row/col
                    row, col = dataset.index(x, y)

                    # Define a window of 200x200 pixels around the
center of the detected ship (100 in each direction)
                    window_size = 100
                    row_start = row - window_size
                    row_end = row + window_size
                    col_start = col - window_size
                    col_end = col + window_size

                    # Ensure window stays within image bounds
                    height, width = dataset.height, dataset.width
                    row_start = max(0, row_start)
                    row_end = min(height, row_end)
                    col_start = max(0, col_start)
                    col_end = min(width, col_end)

                    # Final window dimensions
                    win_height = row_end - row_start
                    win_width = col_end - col_start

                    # Skip invalid windows (in case they are
completely outside image)
                    if win_height <= 0 or win_width <= 0:
                        print(
                            f"    ERROR: Invalid window size
(height={win_height}, width={win_width}), skipping detection")
                        continue

                    # Define window object for reading image data
                    window = rasterio.windows.Window(col_start,
row_start, win_width, win_height)

                    # Read image data in window
                    cropped_img = dataset.read(window=window)

                    # Convert image to RGB format
                    if cropped_img.shape[0] >= 3:
                        img_array = np.stack([cropped_img[0],
cropped_img[1], cropped_img[2]], axis=2)
                    else:
                        img_array = np.repeat(cropped_img[0][:, :,
np.newaxis], 3, axis=2)

                    # Ensure pixel values are in [0, 255] range and
of type uint8
```

```
                    if img_array.dtype != np.uint8:
                        img_array = np.clip(img_array, 0, 255)
                        img_array = img_array.astype(np.uint8)

                    # Generate unique filename using uuid
                    unique_id = uuid.uuid4().hex
                    save_path = os.path.join(output_dir,
f"detection_{unique_id}.png")

                    # Save image
                    img = Image.fromarray(img_array)
                    img.save(save_path)

                    # Store path in dictionary with information on
all matches
                    det['image_path'] = save_path.replace("\\",
"/")

                    print(f"    Saved cropped image to
{save_path}")

            except Exception as e:
                print(f"Error cropping {tif_path} for date {date}:
{e}")

    return detections_by_date


def generate_ship_maps(data_by_date):

    # Generates interactive Folium maps showing satellite-detected
ships matched with AIS vessel broadcasts for
    # each date. Creates one HTML map per date with markers, ship
details, and trajectories, as well as an index.html
    # for date selection.

    base_detections_dir = "detections"

    # Generate a map for each date
    for date, ship_list in data_by_date.items():
        if not ship_list:
            continue  # Skip if no detections

        # Initialize Folium map centered on the first detection
        m = folium.Map(location=ship_list[0]['sat'], zoom_start=12)

        for info in ship_list:
            # Format satellite timestamp
```

```
            time_satellite = info['time_satellite']
            if not isinstance(time_satellite, datetime):
                time_satellite = datetime.strptime(time_satellite,
'%Y-%m-%d %H:%M:%S')

            # Get coordinates of matched AIS point and any later
positions
            candidates = [{'pos': info['ais'], 'time':
info['time_ais']}]
            for pos in info.get("later_ais_positions", []):
                candidates.append({'pos': (pos['lat'], pos['lon']),
'time': pos['time']})

            # Ensure all times are datetime objects
            for c in candidates:
                if not isinstance(c['time'], datetime):
                    c['time'] = datetime.strptime(c['time'],
'%Y-%m-%d %H:%M:%S')

            # Choose the AIS position closest in time to the
satellite detection
            best_match = min(candidates, key=lambda c:
abs((c['time'] - time_satellite).total_seconds()))

            # Calculate distance and estimated speed
            distance_m = int(geodesic(info['sat'],
best_match['pos']).meters)
            time_diff_seconds = abs((best_match['time'] -
time_satellite).total_seconds())
            time_diff_minutes = time_diff_seconds / 60 if
time_diff_seconds > 0 else 0.1
            speed_m_per_s = distance_m / time_diff_seconds if
time_diff_seconds > 0 else 0
            speed_kmh = speed_m_per_s * 3.6
            knots = speed_kmh * 0.5399

            # Classify vessel type based on length and determine
likeliness of match using speed
            classification = "Tanker/Cargo/Passenger/Other" if
info['length'] > 60 else "Fishing/Pleasure Craft"
            speed_label = "Realistic" if speed_kmh <= 45 else
"Unlikely; possible unauthorized vessel"

            # Format timestamps
            time_satellite_str = time_satellite.strftime('%Y-%m-%d
%H:%M:%S')
            ais_time_str = best_match['time'].strftime('%Y-%m-%d
%H:%M:%S')
```

```
            original_sat_coord = info['sat']
            ais_coord = best_match['pos']
            dist_m = geodesic(original_sat_coord, ais_coord).meters

            # Slightly shift red marker to avoid overlapping with
AIS marker if they are very close
            if dist_m < 50:
                offset = 0.0003  # around a 30 meters shift
                adjusted_sat_coord = (original_sat_coord[0] +
offset, original_sat_coord[1] + offset)
            else:
                adjusted_sat_coord = original_sat_coord

            # Create popup HTML for satellite detection (red
marker)
            popup_html = f"""
                    <div style='text-align:center;'>
                        <b>Detected Vessel</b><br>
                        <img src="{info['image_path']}"
alt="Detected Ship" style="width:300px; border-radius:8px;
margin-top:5px;"><br>
                        <div>Length: {info['length']} m<br>
                        <div>Classification: {classification}<br>
                        <div>Coordinates: {original_sat_coord}<br>
                        <div>Timestamp: {time_satellite_str}</div>
                    </div>
                """

            # Add red marker for satellite detection
            folium.Marker(
                adjusted_sat_coord,
                icon=folium.Icon(color='red'),
                popup=folium.Popup(popup_html, max_width=350)
            ).add_to(m)

            # Add blue marker for AIS position
            folium.Marker(
                ais_coord,
                tooltip=(f"<b>AIS Match</b><br>"
                         f"Name: {info['name']}<br>"
                         f"Location: {info['ais']}<br>"
                         f"Classification:
{info.get('classification', 'Unknown')}<br>"
                         f"Time: {ais_time_str}"),
                icon=folium.Icon(color='blue', icon='ship',
prefix='fa')
            ).add_to(m)
```

```
            # Add connecting line between satellite and AIS
            folium.PolyLine([adjusted_sat_coord, ais_coord],
color='gray', weight=2).add_to(m)

            # Invisible thick line with hover tooltip showing
distance/time/speed/status
            folium.PolyLine(
                [adjusted_sat_coord, ais_coord],
                color='gray',
                weight=20,
                opacity=0.0,
                tooltip=(f"Distance: {distance_m} m<br>"
                         f"Time difference: {time_diff_minutes:.1f}
min<br>"
                         f"Average Speed: {knots:.6f} knots
({speed_kmh:.2f} km/h)<br>"
                         f"<b>Status:</b> {speed_label}")
            ).add_to(m)

            # Plot later AIS positions (trajectory) if available
            later_positions = info.get("later_ais_positions", [])
            if later_positions:
                prev = best_match['pos']
                for pos in later_positions:
                    point = (pos['lat'], pos['lon'])
                    time_str = pos['time'].strftime('%Y-%m-%d
%H:%M:%S')
                    if point == best_match['pos']:
                        continue  # Skip duplicate

                    folium.CircleMarker(
                        location=point,
                        radius=4,
                        color='blue',
                        fill=True,
                        fill_color='blue',
                        fill_opacity=0.7,
                        tooltip=f"Later AIS position<br>Time:
{time_str}"
                    ).add_to(m)

                    folium.PolyLine([prev, point],
color='darkblue', weight=2, dash_array='5,5').add_to(m)
                    prev = point

        # Save map HTML for this date
        map_filename = f"map_{date}.html"
```

```
        m.save(map_filename)

        # Read saved HTML and inject custom UI (back button,
legend, help)
        with open(map_filename, "r", encoding="utf-8") as file:
            map_html = file.read()

        # Back button + legend HTML
        extra_ui_html = f"""
        <!-- Back Button -->
        <div style="
            position: fixed;
            bottom: 20px;
            left: 20px;
            z-index: 9999;
        ">
            <a href='index.html' style="
                background-color: #ffffff;
                padding: 10px 15px;
                border-radius: 6px;
                text-decoration: none;
                font-weight: bold;
                box-shadow: 0 2px 6px rgba(0,0,0,0.2);
                color: #333;
                font-family: Arial, sans-serif;
            ">← Choose a different date</a>
        </div>

        <!-- Legend -->
        <div style="
            position: fixed;
            bottom: 20px;
            right: 20px;
            z-index: 9999;
            background-color: #ffffff;
            border-radius: 8px;
            padding: 12px 16px;
            box-shadow: 0 2px 6px rgba(0,0,0,0.2);
            font-family: Arial, sans-serif;
            font-size: 14px;
            color: #333;
        ">
            <b>Legend</b><br>
            <div style="margin-top: 6px;">
                <span style="display: inline-block; width: 12px;
height: 12px; background-color: red; border-radius: 50%;
margin-right: 6px;"></span>
                Detected vessel
```

```
            </div>
            <div>
                <span style="display: inline-block; width: 12px;
height: 12px; background-color: #34cceb; border-radius: 50%;
margin-right: 6px;"></span>
                AIS broadcasting vessel
            </div>
            <div>
                <span style="display: inline-block; width: 12px;
height: 12px; background-color: darkblue; border-radius: 50%;
margin-right: 6px;"></span>
                AIS vessel trajectory
            </div>
        </div>
        """

        # Help button & tooltip HTML + JS
        help_tooltip_html = """
        <!-- Help Button -->
        <div style="
            position: fixed;
            bottom: 20px;
            left: 50%;
            transform: translateX(-50%);
            z-index: 9999;
        ">
            <button id="helpButton" style="
                background-color: #f1f1f1;
                color: #333;
                border: none;
                border-radius: 50%;
                width: 40px;
                height: 40px;
                font-size: 22px;
                font-weight: bold;
                cursor: pointer;
                box-shadow: 0 2px 6px rgba(0,0,0,0.2);
                font-family: Arial, sans-serif;
            ">?</button>
        </div>

        <!-- Help Tooltip -->
        <div id="helpTooltip" style="
            display: none;
            position: fixed;
            bottom: 70px;
            left: 50%;
            transform: translateX(-50%);
```

```
            background-color: rgba(0, 0, 0, 0.85);
            color: white;
            padding: 10px 15px;
            border-radius: 8px;
            z-index: 9999;
            font-family: Arial, sans-serif;
            font-size: 14px;
            max-width: 300px;
            text-align: center;
            box-shadow: 0 2px 6px rgba(0,0,0,0.3);
        ">
            Hover over lines and points for more info.<br>Click red
markers for detection details.
        </div>

        <script>
            const helpButton =
document.getElementById("helpButton");
            const helpTooltip =
document.getElementById("helpTooltip");

            helpButton.addEventListener("click", () => {
                if (helpTooltip.style.display === "none") {
                    helpTooltip.style.display = "block";
                } else {
                    helpTooltip.style.display = "none";
                }
            });
        </script>
        """

        # Insert extra UI before </body>
        map_html = map_html.replace("</body>", extra_ui_html +
help_tooltip_html + "\n</body>")

        # Save updated HTML with UI elements
        with open(map_filename, "w", encoding="utf-8") as file:
            file.write(map_html)

    # Create date picker UI
    available_dates = sorted(data_by_date.keys())
    background_css = """
        background-image: url('ship_background.png');
        background-size: cover;
        background-position: center;
        background-repeat: no-repeat;
    """
```

```
# Generate main UI with dropdown to select date
html = f"""
<!DOCTYPE html>
<html>
<head>
    <title>Ship Detection Calendar</title>
    <style>
        body {{
            font-family: Arial, sans-serif;
            {background_css}
            margin: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
        }}
        .container {{
            background: rgba(255, 255, 255, 0.9);
            padding: 40px;
            border-radius: 12px;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
            text-align: center;
            max-width: 400px;
        }}
        h1 {{
            margin-bottom: 20px;
            color: #333;
        }}
        p {{
            margin-bottom: 20px;
            font-size: 16px;
            color: #555;
        }}
        select {{
            padding: 10px;
            font-size: 16px;
            border-radius: 6px;
            border: 1px solid #ccc;
            width: 100%;
        }}
    </style>
</head>
<body>
    <div class="container">
        <h1>Maritime Monitoring System</h1>
        <p>Select a date:</p>
        <select id="dateDropdown">
            <option value="">-- Choose a date --</option>
```

```
            </select>
        </div>

        <script>
        const availableDates = {available_dates};

        const dropdown = document.getElementById("dateDropdown");

        // Populate dropdown with available dates
        availableDates.forEach(date => {{
            const option = document.createElement("option");
            option.value = date;
            option.textContent = date;
            dropdown.appendChild(option);
        }});

        dropdown.addEventListener("change", function() {{
            const selected = this.value;
            if (selected) {{
                window.location.href = "map_" + selected + ".html";
            }}
        }});
        </script>
    </body>
    </html>
    """

    # Save the main index page
    with open("index.html", "w", encoding="utf-8") as f:
        f.write(html)

    # Open index page in default web browser
    webbrowser.open("index.html")


# Set root folder containing satellite imagery and AIS folder
containing AIS data per date
root_folder = "Satellite_Data"
ais_folder = "AIS_Data"

all_results = {}  # Dictionary to collect detection results sorted
by date

# Loop through each subfolder in the satellite data root folder
for subfolder in os.listdir(root_folder):
    subfolder_path = os.path.join(root_folder, subfolder,
"PSScene")
    if not os.path.isdir(subfolder_path):
```

```
        # Skip if expected PSScene folder doesn't exist
        continue

    # Find TIF and metadata JSON files in the sub folder
    tif_files = glob.glob(os.path.join(subfolder_path,
"*_Visual_clip.tif"))
    meta_files = glob.glob(os.path.join(subfolder_path,
"*_metadata.json"))

    if not tif_files or not meta_files:
        print(f"Skipping {subfolder}: Required files not found.")
        continue

    tif_path = tif_files[0]        # Use the first TIF file found
    metadata_path = meta_files[0]  # Use the first metadata JSON
file found

    # Extract date from TIF filename (format: YYYYMMDD) and
reformat to YYYY-MM-DD
    tif_filename = os.path.basename(tif_path)
    raw_date = tif_filename.split('_')[0]  # e.g. "20250516"
    formatted_date =
f"{raw_date[:4]}-{raw_date[4:6]}-{raw_date[6:]}"  # changed to
format "2025-05-16"

    # Locate corresponding AIS folder for the date
    ais_date_folder = os.path.join(ais_folder, formatted_date)
    ais_csv_path = None
    all_ais_files = []

    if os.path.isdir(ais_date_folder):
        # List all CSV AIS files in the date folder
        csv_files = [f for f in os.listdir(ais_date_folder) if
f.endswith(".csv")]
        if csv_files:
            # Sort AIS files by time extracted from filename
(requires extract_time_from_filename function)
            csv_files.sort(key=extract_time_from_filename)
            all_ais_files = [os.path.join(ais_date_folder, f) for f
in csv_files]
            ais_csv_path = all_ais_files[0]  # Use the earliest AIS
file for initial matching
        else:
            print(f"No AIS CSV files found in folder:
{ais_date_folder}. Skipping.")
            continue
    else:
```

```
        print(f"AIS folder for date {formatted_date} not found.
Skipping.")
        continue

    # Print progress info
    print(f"Processing TIF: {tif_filename}")
    print(f"Using earliest AIS file:
{os.path.basename(ais_csv_path)}")
    print(f"Also found {len(all_ais_files)} AIS files for
trajectory building later")

    # Call main function
    # Pass the TIF file and its metadata directories and all AIS
files for a specific date to main function
    new_result = main(
        tif_path=tif_path,
        ais_file=ais_csv_path,
        meta_file=metadata_path,
        delta_time=30,  # Time window in minutes for matching
satellite and AIS data
        all_ais_files=all_ais_files,
        tile_size=(512, 512),   # Size of tiles to split image for
detection
        stride=(256, 256),      # Overlap stride for tiling
        dedup_threshold_m=50    # Distance threshold used to avoid
duplicate detections
    )

    # Sort results by date in all_results dictionary
    for date, entries in new_result.items():
        if date not in all_results:
            all_results[date] = entries
        else:
            all_results[date].extend(entries)

# If any detections found, generate cropped images and interactive
maps
if all_results:
    final_results = add_cropped_images(all_results)  # Crop
satellite images around detections, save images
    generate_ship_maps(final_results)                # Create
interactive maps with detections and AIS matches
else:
    print("No detection results to generate maps for.")
```

Appendix D: Evaluation code

```
from ultralytics import YOLO

# Load trained model
model = YOLO("YOLO_model.pt")


# Validate the model
metrics = model.val(
   data="dataset.yaml",
   imgsz=512,
   device="cpu"
)

# Get performance metrics
precision = metrics.box.mp
recall = metrics.box.mr
f1_score = sum(metrics.box.f1) / len(metrics.box.f1) if
metrics.box.f1 else 0
map5095 = metrics.box.map

print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1 Score:  {f1_score:.4f}")
print(f"mAP@0.5:   {map50:.4f}")
print(f"mAP@0.75:   {map75:.4f}")
print(f"mAP@0.5:0.95: {map5095:.4f}")
```

Appendix E: Questionnaire for evaluation of visualization (including answers)

Questions

How much experience do you have in the field of maritime monitoring? (Choose the option that best applies to you) *

- ○ No experience
- ○ Some familiarity (e.g. student, general interest)
- ○ Moderate experience (e.g. worked on related projects)
- ○ Professional experience (actively working in the field)

How intuitive did you find the visualization to use? *

	1	2	3	4	5	
Very confusing	○	○	○	○	○	Very intuitive

Was it clear what each element on the map represented? *

	1	2	3	4	5	
Not clear at all	○	○	○	○	○	Very clear

Do you think this type of visualization could be helpful in a real-world maritime monitoring context? *

- ○ Yes
- ○ No

If not, what do you think is missing in order for this service to become useful as a real-world maritime monitoring application?

Tekst lang antwoord

According to you, what features are missing? *

Tekst lang antwoord

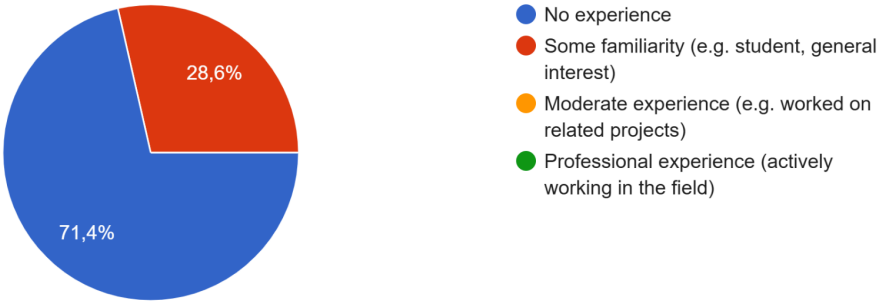
Do you have any other comments, suggestions, or feedback you would like to share?

Korte antwoordtekst

Answers

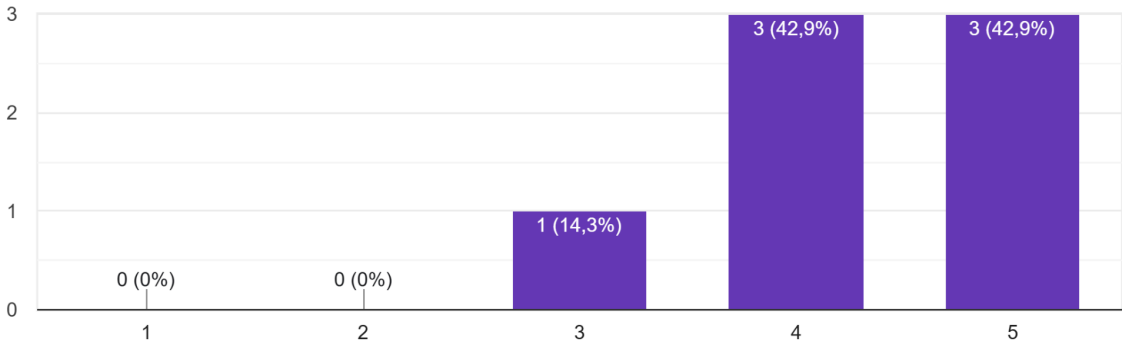
How much experience do you have in the field of maritime monitoring? (Choose the option that best applies to you)

7 antwoorden



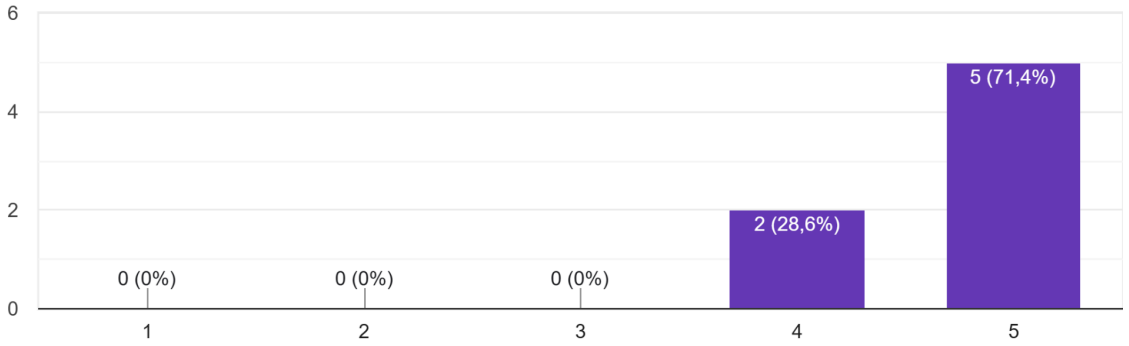
How intuitive did you find the visualization to use?

7 antwoorden



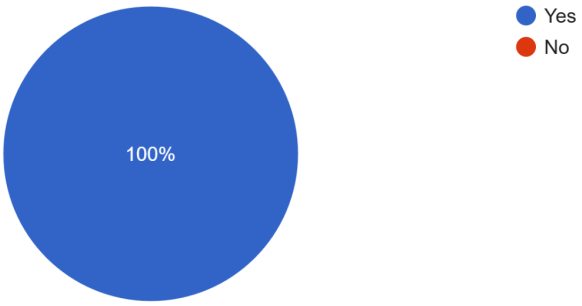
Was it clear what each element on the map represented?

7 antwoorden



Do you think this type of visualization could be helpful in a real-world maritime monitoring context?

7 antwoorden



If not, what do you think is missing in order for this service to become useful as a real-world maritime monitoring application?

2 antwoorden

It is useful - especially if you have frequent refreshes, and if it's easy to use / free (like it is). It is concerning that the AIS icon is far from detections especially in the third day. I am worrying that there is a lot of competition with the market and already advanced systems - targeting small vessels could differentiate the product.

The visualization was nice. I'm also interested in seeing an actual image of the ship along with AIS match information in a pop-up tab, similar to the Detected Vessel info pop-up.

According to you, what features are missing?

7 antwoorden

it can be improved, e.g. not my field but ISAR to detect during night/clouds/bad weather? also advanced processing to be able to flag AIS-off targets, create density heatmaps, track routes, detect smaller vessels which are responsible usually for illegal fishing and other (e.g. hot topic of migrations). It would be nice for users to be able to select grid areas or vessels to download csv or geo file for import to GIS platforms and other / models for risks/impacts. You can adjust it to be used for managers; e.g. managers to keep track of the compliance with anchoring location, managers to send real-time alarms in projects where mammals/turtles are detected, risk maps of noise pollution, animal collisions etc.

To me, all the relevant points are captured.

As I mentioned, I mean an actual image, something a ship monitoring agency would use. I want to compare the detected ship with the actual one, as part of a human-in-the-loop verification process.

Historical data about a vessel's route

I have no experience in this field but this looks a little empty like addition information might help or the trajectories or the path of the ships. count of ships something like that should be added.

If possible, more vessel information (i.e Fure Viken is an oil/chemical tanker, under the flag of Sweden).

More information on each vessel, more data for all vessels.

Do you have any other comments, suggestions, or feedback you would like to share?

3 antwoorden

The results are interesting. I think you should exclude smaller boats, as I see more errors in those categories.

home page visulization is good.but i would suggest that the second tab where the maps is displayed , should have some addition visualization like tabs or something like that

N/A

Appendix F: Use of AI assistance

While all core code, design choices, and implementation ideas in this project were developed independently, OpenAI's ChatGPT [39] was occasionally used to help troubleshoot specific errors or clarify certain aspects of Python behavior. All AI-generated content was carefully reviewed, modified where necessary, and fully validated to make sure it aligned with the intended functionality and goals of the project. The overall methodology, structure, and logic of the system were fully conceived and implemented by myself.