Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Sakib Ahmed

Maritime Small Object Detection from UAVs for Autonomous SAR Missions using Deep Learning-based Approaches

Master's thesis in Master in Marine and Maritime Intelligent Robotics (MIR)
Supervisor: Asgeir Johan Sørensen
Co-supervisor: Oscar Pizarro
June 2023

MIR MARINE & MARITIME INTELLIGENT ROBOTICS

NTNU
Norwegian University of Science and Technology

Co-funded by the Erasmus+ Programme of the European Union

Sakib Ahmed

Maritime Small Object Detection from UAVs for Autonomous SAR Missions using Deep Learning-based Approaches

Master's thesis in Master in Marine and Maritime Intelligent Robotics (MIR)
Supervisor: Asgeir Johan Sørensen
Co-supervisor: Oscar Pizarro
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

Abstract

In the maritime context, to cover large area for surveillance, search and recovery missions, there is no better alternative to onboard autonomous aerial visual intelligence. However most computer vision algorithms struggles with small object detection which is a key challenge to solve for aerial visual autonomy. This thesis explores the development of techniques for detecting small objects in maritime context using deep learning. The primary objective is to improve surveillance and monitoring capabilities in maritime activities without huge computational overhead, where small objects play a significant role. Two approaches are presented: altitude aware dynamic spatial tiling for improving accuracy and reducing memory demand. Additionally, an anomaly based detector for semi supervised object detection.

The dynamic tiling approach involves spatially scaling the input images and then dividing the input image into smaller tiles, allowing for more focused detection and tracking of small objects. The proposed system architectures and the findings from field experiments demonstrate significant effectiveness of this approach for small object detection and faster inference.

In the second approach, a VAE (Variational Autoencoder) anomaly detector is employed to identify anomalous regions within the maritime scene. This method helps identify regions of interest (ROIs), aided with another input tiling technique we developed, this significantly enhance the efficiency of small object detection. The training process and performance evaluation of this approach are thoroughly discussed, including a comparison with traditional methods such as OpenCV.

The results obtained from both approaches showcase their potential in accurately detecting small objects in maritime context.

Sammendrag

I maritim sammenheng finnes det ikke noe bedre alternativ enn autonom visuell intelligens om bord for å dekke store områder for overvåkings-, søke- og gjenopprettingsoppdrag. De fleste datasynalgoritmer sliter imidlertid med deteksjon av små objekter, noe som er en viktig utfordring å løse for visuell autonomi fra luften. Denne avhandlingen utforsker utviklingen av teknikker for deteksjon av små objekter i maritim kontekst ved hjelp av dyp læring. Det primære målet er å forbedre overvåkings- og kontrollfunksjonene i maritime aktiviteter uten store beregningsutgifter, der små objekter spiller en viktig rolle. To tilnærminger presenteres: høydebevisst dynamisk romlig flislegging for å forbedre nøyaktigheten og redusere minnebehovet. I tillegg presenteres en anomalibasert detektor for semiovervåket objektdeteksjon.

Den dynamiske fliseleggingstilnærmingen innebærer romlig skalering av inngangsbildene og deretter inndeling av inngangsbildet i mindre fliser, noe som muliggjør mer fokusert deteksjon og sporing av små objekter. De foreslåtte systemarkitekturene og resultatene fra felteksperimenter viser at denne tilnærmingen er svært effektiv for deteksjon av små objekter og raskere inferens.

I den andre tilnærmingen brukes en VAE-anomalidetektor (Variational Autoencoder) til å identifisere avvikende regioner i den maritime scenen. Denne metoden bidrar til å identifisere interessante regioner (ROI-er), og ved hjelp av en annen flisleggingsteknikk som vi har utviklet, forbedrer dette effektiviteten ved deteksjon av små objekter betydelig. Opplæringsprosessen og ytelsesevalueringen av denne tilnærmingen diskuteres grundig, inkludert en sammenligning med tradisjonelle metoder som OpenCV.

Resultatene fra begge metodene viser at de har potensial til å detektere små objekter i maritim kontekst.

(*Translated with www.DeepL.com/Translator*)

Preface

This master's thesis represents the culmination of my two years of studies in the Master in Marine and Maritime Intelligent Robotics (MIR) program at the University of Toulon and the Norwegian University of Science and Technology, supported by the ERASMUS MUNDUS Scholarship.

I extend my sincere gratitude to my supervisor, Dr. Oscar Pizarro, for his invaluable guidance, unwavering support, and insightful contributions throughout the research process. Special thanks to my mentor, M M Shaifur Rahman, for their feedback and motivation. I also acknowledge Dr. Ricard Marxer for inspiring my pursuit of deep learning.

I am thankful to the professors, researchers, and fellow students who have contributed to my academic journey. Lastly, I appreciate the support of my family and friends, who have been instrumental in my success.

This thesis is dedicated to all those who have shaped my academic and personal growth.

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

1.1 Background and Motivation

Autonomous launch and recovery of small underwater vehicles is an important capability to enable one form of large-scale, scalable systems in the ocean. One possible design would consider an autonomous surface vessel (ASV) that services small vehicles by recharging them and transferring data. This ASV would directly launch and recover small vehicles. Alternatively, an unmanned aerial vehicle (UAV) could be used as an intermediate vehicle to launch and recover the underwater asset. Both cases could use visual detection and homing onto a floating payload waiting for recovery.

While the initial inspiration stems from this, this thesis explores the fundamental challenges pertaining to aerial small object detection in the maritime context and investigates methods that can be applied to many scenarios requiring autonomous aerial visual intelligence.

1.2 Problem Statement

In UAV-based SAR (search and recovery), object detection(reliably fast) is the main challenge. SOT deep learning models suffer in this task mainly for two reasons,

1. From High altitudes, the “pixel-print”(relative pixel area) of objects is very small compared to the whole image.
2. A specialized model like in [1] (YOLO-fine) which is optimized for small objects will start suffering when the UAV will descend for recovering the target as it will get bigger (larger “pixel print”).

Most modern object detection algorithms are based on CNNs. In deeper layers, the characteristics of smaller objects vanish, thereby making it harder for the detector to pick them out. For Example, A 640p image is scaled down to about 20p in the hidden layers of YOLOv5’s backbone[2]. Small objects will have few

features, which means their features will vanish in a deeper network and never be recognized.

Methods	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster-RCNN (Baseline)	ResNet-50-C4	36.5	57.3	39.3	18.4	40.6	50.6
SOD-MTGAN (Ours)	ResNet-50	**37.2**	**57.7**	**40.2**	**19.9**	**41.1**	**51.2**
Mask-RCNN (Baseline)	ResNet-101-FPN	40.9	61.9	44.8	23.5	44.2	53.9
SOD-MTGAN (Ours)	ResNet-101	**41.5**	**62.5**	**45.4**	**25.1**	**44.6**	**54.1**

Figure 1.1: Small Object Detection Benchmark, adopted from [3].

As shown in [3], the SOTA performance of small objects is significantly less than medium and large objects. If an object has dimensions 32 x 32px in an image of 1024 x 1024px, and if we divide the image into 4 tiles of 512 x 512px, then if we compare the area of the objects in both cases, in 1st case (original image), the object occupies 0.098% of the original image area, whereas in the 2nd case, the object occupies 0.39% of the split image patch.

In field robotics, If we consider practical implementation, for example in a UAV, we have to find out a method of inputting high-resolution images to the DL models without exceeding memory and computational constraints.

1.3 Main Contributions

This master thesis presents a comprehensive investigation into maritime aerial small object detection. Through extensive research, data collection, analysis, and experimentation, this study has made several significant contributions to the field.

This thesis is the continuation of the work from the specialization project where we investigated the challenges with maritime autonomous visual intelligence and found that there is a necessity in terms of datasets for maritime vision-based learning. We also reached the conclusion that altitude-aware tiled-based object detection can benefit especially in small object detection.

Based on the leanings from the specialization project, we conducted several experiments to build upon our hypothesis and two small field trials which confirmed the hypothesis.

Later we investigated two deep-learning-based methods for small object detection/localization and classification in combination with a novel altitude-aware dynamic scaling and tiling that addresses the issue of memory constraints in practical implementation.

We also propose a two-stage object detection framework that loosely decouples the object localization and classification problem and allows for adaption into domain/application-specific combinations of multiple deep-learning models or heuristic methodologies.

1.4 Thesis Outline

This thesis is structured as follows:

Chapter 2: Background
Covers related work, and introduces relevant theory for this thesis.

Chapter 3: Methodology
Describes the proposed system architecture, datasets, and experimental setup and explains training and testing of the system.

Chapter 4: Results
Describes the results of the experiments.

Chapter 5: Discussion
Contains a discussion of the choice of training data, and the system architecture. The experimental results, consistency with related work, and fulfillment of the research objectives are also discussed.

Chapter 6: Conclusion and Further Work
Concludes the work, and proposes areas that could be further explored.

Chapter 2

Background

This chapter covers relevant theoretical aspects related to the thesis.

2.1 Relevant Theories and Definitions

2.1.1 Object Detection and Tracking

Object Detection

Object detection is one of the most common machine vision tasks. It is the process of both localizing and identification of any object in an image or video frame. When an object is recognized, bounding boxes are drawn around it, allowing us to determine where it is in (or how it moves through) a scene. Image classification involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all of these problems are referred to as object recognition. Empirically it is done by computing a mathematical filter to remove the background or isolate the target object with its geometric properties(edges, colors, textures, etc). In machine learning, it is done by training a neural network or model to learn these features from example/training data. Traditionally in machine learning, object detection was solved using two-stage detectors like **rcnn**s for example, the first stage was used to detect regions of Interest and the second stage was used to classify them([4]). That approach was accurate but quite slow. Currently, the most popular solutions are single-stage detectors like *YOLO: You Only Look Once* [5] or *SSD: Single shot multibox detector* [6], they are the perfect compromise between speed and accuracy.

Object Tracking

Tracking is essentially looking at subsequent frames of the video and determining whether the object that is seen in the first frame is the same that is seen on the next

frame. There are different types of trackers. Some of them use neural networks, and some of them are based on calculating the intersection over the union (IoU) between the frames. Regardless of the type, the result is usually the same, a unique object ID assigned to every bounding box [7].

2.1.2 Maritime Aerial Object detection

Unmanned Aerial Vehicles (UAV) are a common choice for moderately high-fidelity surveillance of sea surface with high-resolution cameras. But the problem arises for autonomous operation in object detection algorithms as images/videos taken from high altitude has a significantly low subject-to-background ratio. Due to computational constraints, input images for typical machine learning models are scaled down (ranging from 600 to often as low as 32 pixels). In that case, the target objects are spread across only several pixels in the overall image losing most of the features necessary for detection.

Additionally, there is the problem of varying object sizes depending on the altitude of the UAV, i.e., locating from a high altitude and then approaching to recovery. In that case, the object size dramatically increases in subsequent frames which also poses a problem for the detection algorithms.

Another problem with marine scenarios is that the environment is highly dynamic and there is high variance in color and sea state(wave patterns, etc). This is often the motivation for using thermal cameras instead of RGB(visual light) imaging. However, they come with their own set of limitations such as low fidelity, losing important features based on object colors(in RGB spectrum), covariance with environmental temperature, and reflective objects, etc.

Ground Sample Distance (GSD)

The term "ground sampling distance" describes how much a single photograph covers ground or surface area while it is in flight. This distance is the amount of ground the drone covers per image while flying a mapping flight with the camera pointed downward (nadir position). GSD is the amount of facade surface area covered by a single image taken when flying vertically and mapping a skyscraper or facade. GSD is commonly expressed as cm/pixel. This is a reference to how much ground or surface area a drone image covers.

$$GSD = \frac{\text{Sensor Width} \times \text{Working Distance}}{\text{Focal Length} \times \text{Image Width}} \tag{2.1}$$

The greater the image resolution and the more details that can be seen in the photos, the lower the GSD.

In simple terms, it is another interpretation of FOV in the context of aerial or satellite imaging.

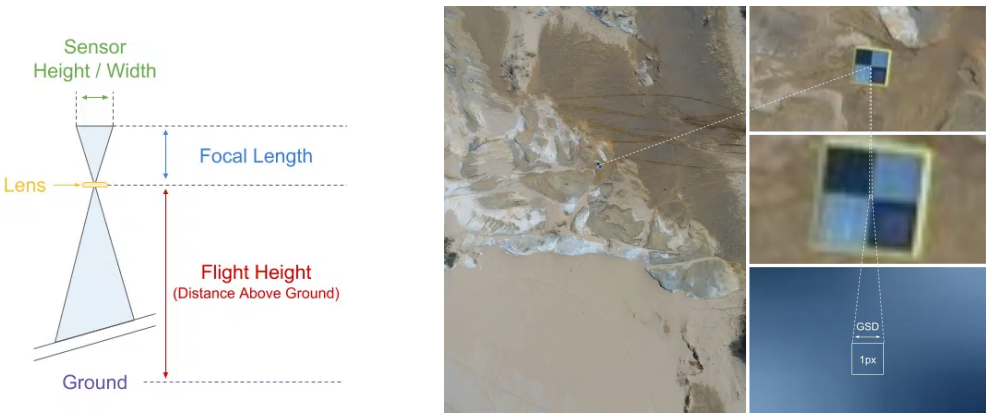


Figure 2.1: Ground Sampling Distance taken from [8]

Tiling

Tiling is a generic preprocessing block that clips rasters into rectangle-shaped tiles. Tiling is especially important for large raster datasets that should be broken up into more manageable pieces to fit into memory and to improve performance.

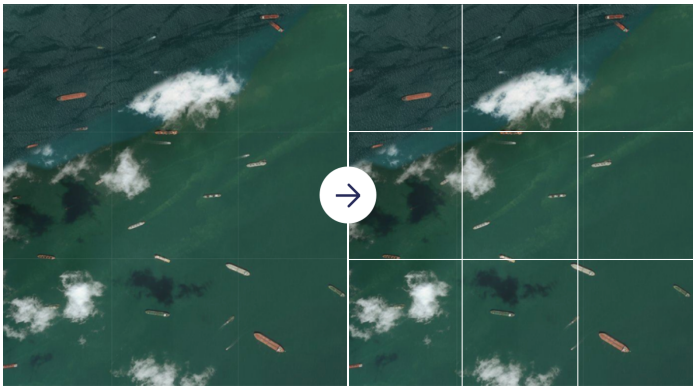


Figure 2.2: Image broken into tiles, taken from [9]

YOLOv5

YOLOv5 is an unofficial release of the YOLO family. It is essentially a Pytorch implementation of the official YOLOv4 [10] model which was built on the darknet framework. YOLOv5 is not considered part of the official YOLO series but was heavily inspired by the original one-stage YOLO architecture. Compared to previous versions, the YOLOv5 has a CSP backbone and PA-NET neck [11]. The major improvements include mosaic data augmentation and auto-learning bounding box anchors.

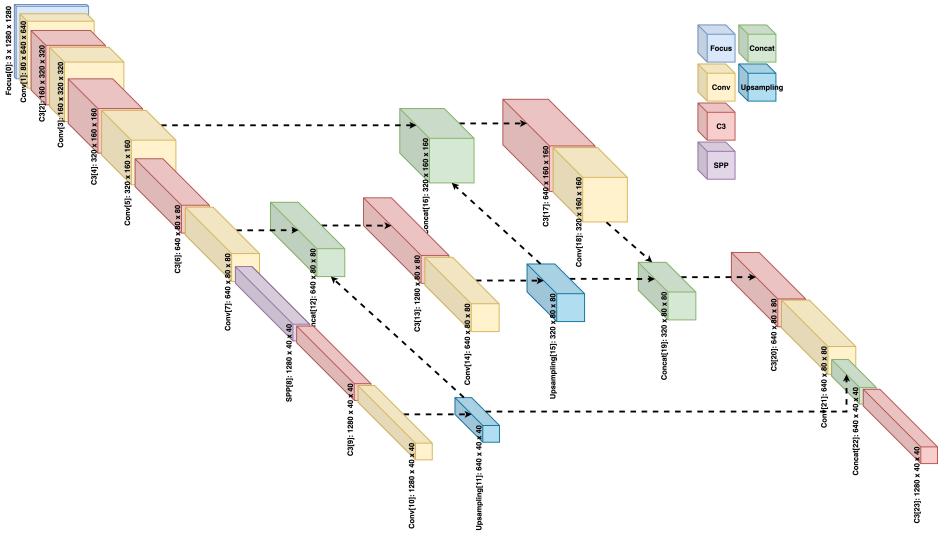


Figure 2.3: YOLOv5x architecture taken from [12]

Bounding Box Anchors

State-of-the-art object detection models like EfficientDet and YOLO employ anchor boxes as a starting point to predict and locate objects within an image frame. The process involves creating many anchor boxes, usually rectangles of different scales and shapes that might fit the target objects. They may start from the centre or around the image space and then the model predicts offsets from these boxes to generate candidate boxes. A loss function is calculated based on a ground truth example, and the probability of overlap between a predicted offset box and a real object is determined determined by IoU. If the probability exceeds set threshold, the prediction is factored into the loss function. Through a gradual process of rewarding and penalizing predicted boxes, the models are trained to accurately localize true objects in the image. [13]

2.2 Literature Review

2.3 Literature Review

Different studies have been done over the years to investigate object detection at tracking at the sea surface. [14] proposed an unmanned autonomous deployment and recovery mechanism using UAV and USV. They experimented with active payload (the MUGs latch onto the UAV with EPM), GPS data, and are marked with Aruco Tags with specific colors for better detection and pose-distance estimation for recovery. They also conduct the experiment under the constraint that the weather is favorable and adequate visibility is ensured. While this systems-of-systems can be further enhanced with smarter payload design, i.e. short-range wireless communication between the UAV and the payload for precise recovery/latching, this approach will not work in the case of passive payloads i.e, in the case of MUGs that are out of battery or no GPS, unreadable Tags due to waves or weather conditions, or in case of generalized floats with no active marker or homing device. Additionally, to make their landing and launching of the UAV from the USV more robust, a gimbal-stabilized landing dock can be incorporated similarly to [15].

In ([16]), they proposed a complete framework for UAV-based object detection and tracking using thermal cameras and filter-based image segmentation for object detection. They achieved this by extending their previous work on automatic object detection and tracking ([17]) and georeferenced object detection using onboard thermal cameras on small fixed-wing UAVs ([18]). For object detection, they took a similar approach as [19] which is essentially a low pass gradient filter with a threshold. For classification, they used a feature vector containing observed object area, perceived average object temperature, and the first invariant Hu moment([20]) for 5 classes. They used Kalman filter-based tracking and a global nearest-neighbor approach for data association. They achieve very good results using only classical approaches.

in[21], they conducted a comparison study of object detectability under different circumstances between Visible camera image vs thermal camera image. They used a convolutional filter(Sobel filter) and showed how different pre-processing i.e. RGB to greyscale conversion method using all channels or a single channel, noise reduction can significantly affect the result as well as the filters (kernel size) depending on the size and color of the objects. As the authors suggested, it's reasonable to believe that a Deep Learning model or CNN-based classifiers could be used in determining detectability.

Due to the low fidelity of thermal images, although they often work better in detecting, pose estimation for tracking becomes difficult. This is only exacerbated by the fast dynamics of the UAV motion which leads to higher degrees of uncertainty in its pose estimates and consequently the tracking result ([22]). They propose a Schmidt-Kalman filter-based tracking system for UAV with a thermal camera where they try to mitigate the negative effects originating from the uncer-

tainty in sensor pose in tracking.

In [23], they proposed a Deep Learning based fusion technique between thermal and visible light images and showed that it achieves state-of-the-art performance in object assessment visual quality. This technique can be evaluated in the context of Marine applications if a suitable data set is available.

In ([24]), they proposed an update to YOLOv3([25]) with a multi-dilated module between the convolution unit and the receptive field for improving the detectability of very tiny objects for real-time object detection in UAV scenarios. They showed that their implementation is 2.69% faster than the original YOLOv3. They implemented a path aggregation module for fusing the semantic information in a deeper layer with detailed information in earlier layers. YOLO(You Only Look Once) models are single-shot CNN models, they do classification and bounding box regression at the same time based on the current input. Since they are computationally lightweight, they do it on every input frame and achieve the effect of real-time tracking.
Although most segmentation models like YOLO were initially created for RGB images, recently a lot of studies have been adopting them for thermal infrared (TIR) images. In ([26]), they trained 15 different configurations of YOLO models on thermal images and then used the best-performing model for object detection on TIR videos from UAVs. The authors concluded in favor of qualitative and quantitative evaluation of objection detection from TIR images and videos using deep-learning models.
Many recent studies have attempted to do fusion between visible light camera(RGB) and Thermal camera images to capture both low details and high-contrast features from the thermal camera and high details low contrast RGB images. This should potentially help the DL models to classify and segment better.
In ([27]), they proposed MFGNet, a novel dynamic modality-aware filter generation module to boost communication between visible and thermal data by adaptively adjusting the convolutional kernels online for various input images in practical tracking. They also introduced a direction-aware target-driven attention network for global search which can further improve the final tracking performance. While typical trackers are developed based on RGB cameras, including binary classification-based, siamese matching-based, and correlation filter-based tracking which all have static kernels. According to ([27]), the static kernels limits the typical RGB-T models in their final tracking performance. In ([28]), they looked into existing Deep learning based RGB-T trackers and made a survey with which they are aiming to provide as a look-up-table for other researchers in the domain. They made a comparative study among the existing RGB-T architectures, on several challenging benchmarks with statistics. They recommend encoding temporal information in the tracking/detection architectures plays a major role in performance.
In [29], they developed several ways of generating synthetic IR images using the AIR-Sim simulator engine and CycleGAN. They also propose an Illumination

Awareness Network (IAN) which connects two YOLOv4 models trained respectively on RGB and IR datasets and fused together/filtered out in the decision layers. But their experiment does not provide satisfactory results in fusion, and not in synthetic IR-only benchmarks. They conclude that this is due to the lack of features in the synthetic IR images as they are generated from RGB images.
In ([30]), they propose a novel attention-based deep fusion network for RGBT salient object detection. They adopt a strategy of multi-layer feature fusion where, they use two VGG16 networks to extract RGB and thermal infrared features respectively, which can preserve RGB and thermal infrared features before upsampling. Their comparison experiments demonstrate that their method performs best over all the state-of-the-art methods with the most evaluation metrics. However, in development and real application, there are not enough curated(aligned and annotated) RGB-T datasets. In practice, capturing images from two spatially separated cameras with different FOVs will always have alignment as well. In ([31]), they address this issue by proposing a novel deep correlation network, which builds the correspondences between modalities from the spatial, feature, and semantic levels, for weakly alignment-free RGBT SOD which should reduce human labor and save time and cost. They introduced Modality Alignment Module (MAM) to handle the problem of spatial misalignment of two modalities. By implicitly learning spatial affine transform and dynamically generating intermediate representation, MAM can robustly capture the spatial correlation of two modalities in challenging scenarios and a novel bi-directional decoder to enable the model to have a great ability for information selection and suppression.

[32] conducted research on physics based image correction (scaling) for benthic image image segmentation task and showed performance gain on transfer learning. They also showed that fidelity loss due to image scaling has less performance penalty compared to the performance gain achieved by the scale correction. This is specially interesting for us since we are also going to explore the premise of sacrificing fidelity to reduce computational overhead in one of our approaches.

However, in [33], they proposed a Variational Auto Encoder(VAE) based anomaly detector for a semi supervised object detection pipeline for underwater image segmentation. They used VAE to detect objects in images as an anomaly and used latent space clustering to automate thresholding and finally used heuristic filters to suppress noise and extract ROI mask for segmentation. This can be beneficial in some maritime object detection applications as there is a stark scarcity of annotated dataset for developing supervised machine learning algorithms. We shall explore this approach as well in this thesis.

Chapter 3

Methodology

Amongst all methods for the detection of small objects, tiling [34] seems to be the simplest and most affordable (computationally) way to solve this problem of object detection in UAV-based SAR.

One drawback of the tiling is that it will increase the inference/detection time (reduce the fps). For n tiles, at least n^2 tiled patches (more if we need overlapping tiles) need to be passed through the model. But, at high altitudes, an UAV's camera with a fixed FOV will cover a larger area,[35] resulting in the object being in the frame for a longer duration, therefore, compensating for the low fps.

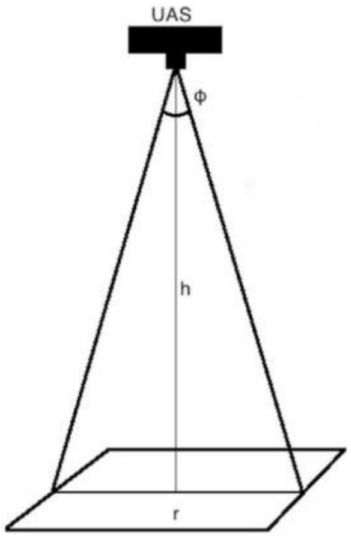


Figure 3.1: UAS field of view geometry, adopted from [35].

Whereas in low altitudes (i.e., while trying to hone onto the target), the object is already closer (higher object-to-background ratio), we can reduce the tiling factor (number of patches) and increase the fps which is more important now.

Therefore, the number of tiles, tiling factor n at a given moment would be a function of the UAV's altitude, camera characteristics, and target object size.

$$n = f(altitude, camera\ FOV, target\ object\ size) \tag{3.1}$$

With an RGBT setup, we can even upgrade the tiling-based system using dynamic patching. The low-fidelity but high contrast Thermal images can help us determine the potential location of objects in the image. Instead of tiling the whole image, we can select patches from these Regions of interest (RoI). Although it makes the detector technically a two-stage OD. Depending on the ROI extraction methode used, this approach can potentially run faster than the 1st method as we are now saving a lot of inference time in the 2nd stage by only looking into the potential portion of the high-resolution image.

3.1 Field experiment

On March 24, 2023, we conducted a small field trial to collect some aerial data in the Trondheim Fjord in front of the NTNU Trondheim Biologiske Stasjon. The weather was cloudy with light wind, hence a relatively calm water surface.

We used a DJI Phantom 4 to capture video clips at varying altitudes keeping the target objects (two A0-sized red buoys) in the frame. The video was captured in 4K (3840×2160, 30fps), camera facing downwards.

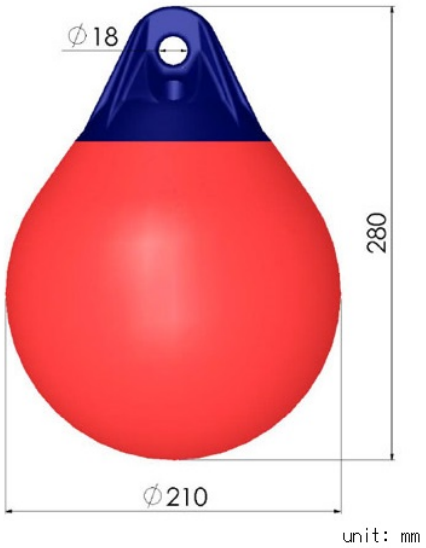


Figure 3.2: Target Object (A0 Buoy) dimension.

3.1.1 Experiment setup

For data pre-processing and annotation, we used [36] and extracted 2916 frames from the video. For this experiment, only 224 images were cherry-picked (includ-

ing augmentations) and annotated with bounding boxes. Figure 3.3 shows the overview of the dataset. We used *ROBOFLOW 2.0 OBJECT DETECTION (FAST)*

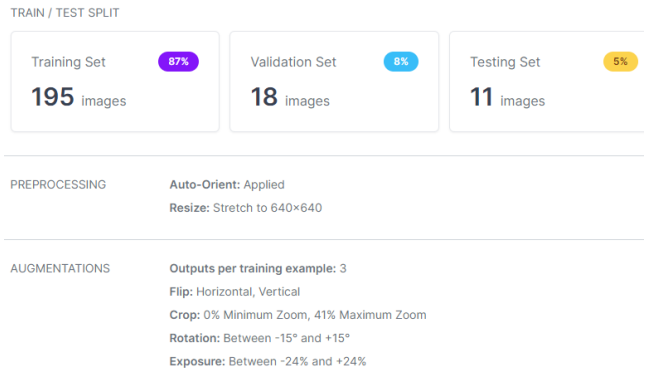


Figure 3.3: Dataset overview.

(pre-trained on COCO) [36] to train on our small dataset for 203 epochs. Concerns regarding over-fitting were ignored since we were only interested in detectability vs spatial scale of the image performance.

3.1.2 Key Observations

We can see that the size of the object with respect to the background matters significantly regardless of the resolution/fidelity (high altitude = low fidelity) of the image. Below is the example with an image taken roughly at 52m altitude and the detection performance (model input size 640x640) with a varying factor of crop ratio.

Figure 3.4: Original image 3840p at 52m altitude: No Detection.

For an input frame of original 4k resolution, the model cannot detect anything. The object occupies only 18x18p which is ~0.004% of the image area (Figure 3.4).

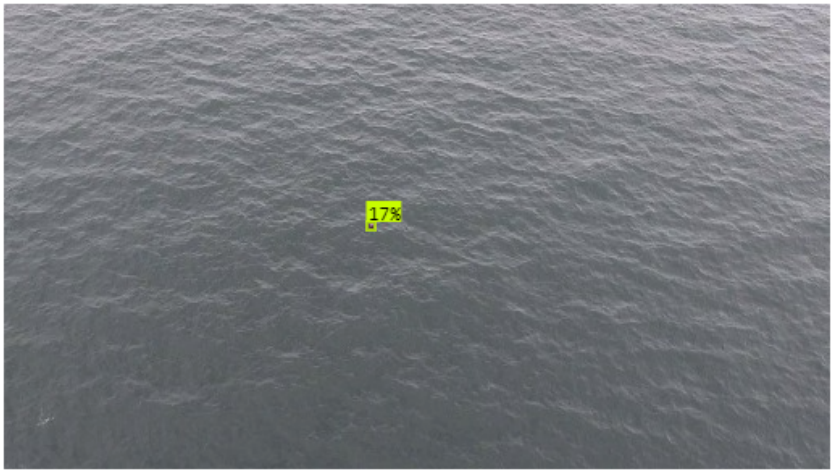


Figure 3.5: Original image cropped at 1920p. Class: Buoys, confidence: 17%.

Similarly, on Figure 3.5, the object-to-background ratio is ∼0.02%, and the model can barely detect it with a confidence of 17%.

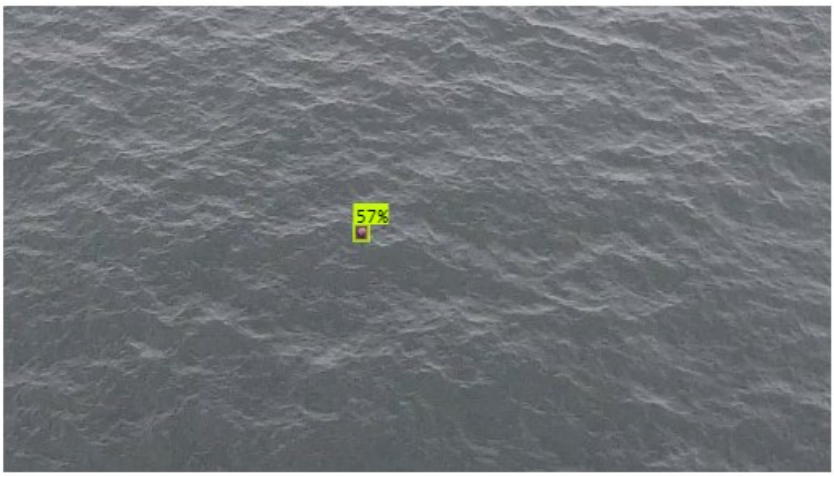
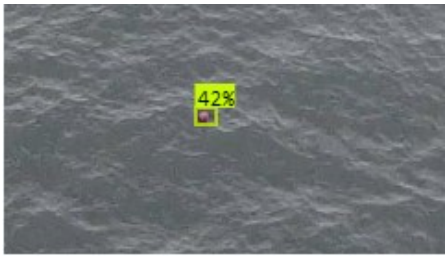


Figure 3.6: Original image cropped at 960p. Class: Buoys, confidence: 57%.

The model detects with its highest confidence in Figure 3.6 where the object-to-background ratio is ∼0.06%.

We can also see a decrease in the confidence score when the object-to-background ratio starts getting bigger. However, in Figure 3.7, we also see that the effect of 50% low fidelity is not that significant compared to the effect of spatial scaling. Interestingly, in Figure 3.8(a) where the object-to-background ratio is ∼1% the model classifies it as a boat. In Figure 3.8(b), full resolution 4K image, boat size 306x150p, the object-to-background ratio is similar, ∼0.55%.

(a) Original image cropped at 480p: Class: Buoys, confidence: 49%



(b) Cropped at 480p and resized to 240p: Class: Buoys, confidence: 42%

Figure 3.7: Same spatial scaling: Original fidelity (left) vs Low fidelity(right).

(a) Original image cropped at 240p: Detected Class: **Boat**, confidence: 38%

(b) Original image 3840p at 50m altitude: Class: Boat, confidence: 32%

Figure 3.8: Same spatial scaling: Buoy (left) vs Boat(right).

3.1.3 Field Experiment Takeaways

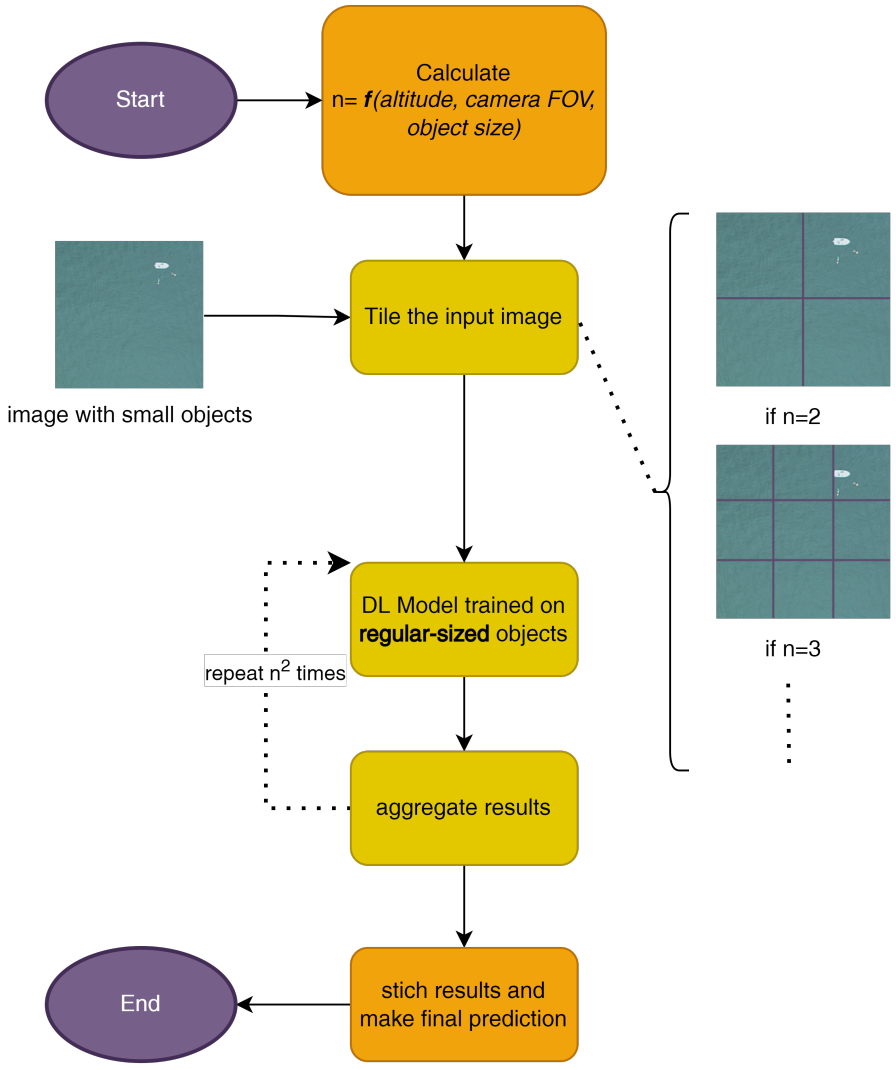
This simple test confirms the hypothesis about benefiting from a two-stage object detection where the first stage would produce potential ROI and the 2nd stage will work on those "zoomed/cropped" sections of the image.

It's also important to note that from a high-altitude aerial image, a lot of the object fidelity is lost and presumably, the model is only learning to identify "anomaly" on the water surface and deciding a label based on the size(background-to-object ratio) of it since it does not have enough information to accurately discern a class label. This also makes sense if we look at the model architecture of most modern OD (Object Detection) models, for example in [2] uses predefined anchor boxes of different scales.

Therefore, it would be necessary to train and infer the model(2nd stage OD) with the roughly same size(background-to-object ratio), in other words, spatial scaling.

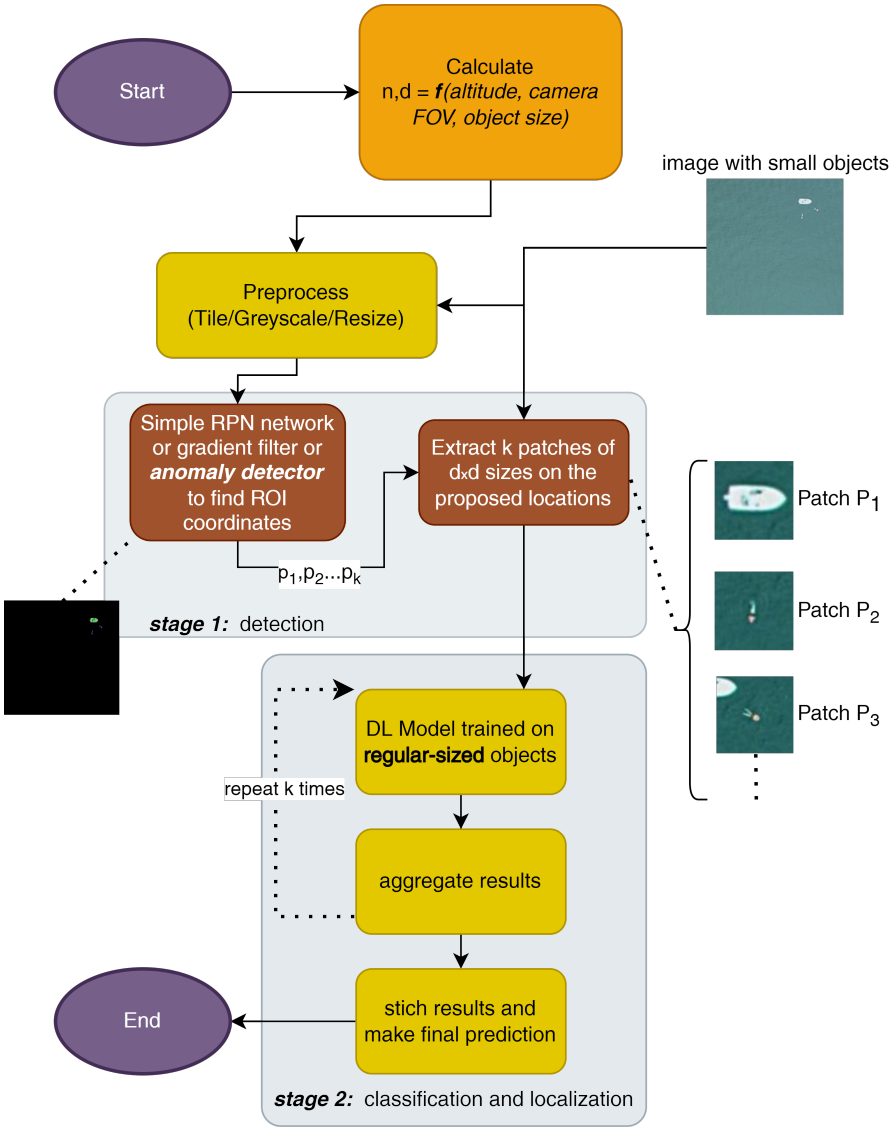
3.2 Proposed System Architectures

Figure 3.9 shows the two proposed pipelines. Figure 3.9a presents the workflow of a single-stage OD with the proposed Dynamic Tiling(Spatial scaling and tiling). Figure 3.9b proposes an alternative to the first approach where we decouple the detection and recognition tasks into two stages.



(a) Dynamic Tiling (one stage detector)

Figure 3.9: Proposed high level System Architectures/pipelines.



(b) Dynamic patching (two stage detector)

Figure 3.9: Proposed high level System Architectures/pipelines.

3.3 Dataset

For experimenting and development, we selected the ***SeaDronesSee*** Dataset [37]. It is the first large-scale annotated (including camera metadata like altitude and gimbal angles/camera orientations) UAV-based dataset of swimmers in open waters. The class labels are swimmers (swimmers with and without life jackets), life saving appliances, boat(speed boat), buoy(green and red), and jet-ski. The dataset offers several challenges, i.e., object detection, single-object tracking, and multi-object tracking. We only used the ***Object Detection v2*** for our purposes. This set contains around 8.9K Training, 1.5K validation and 3.7K test images and annotations in COCO-json format [38]. A detailed overview of the dataset is also attached at Appendix A.

The authors have a brief description in their website about the data acquisition process. We got further insight from the annotation files which also contains image meta data in for most of the images. Presumably, the images were captured using a DJI Mavic series quad copter (4k resolution image frames extracted from video) from altitude ranging between 9-140 meters, and a Trinity F90 series fixed wing drone which produced (in the training and validation sets) two types (mostly high altitude) images: **i.** 3172 images of 5456x3632 pixels (20M) resolution without altitude information, **ii.** 500 RGB images from a Multi-spectral camera of size around 1230x 930 pixels(altitude range from 20-259 meters, but mostly 200+). See Figure 3.10.

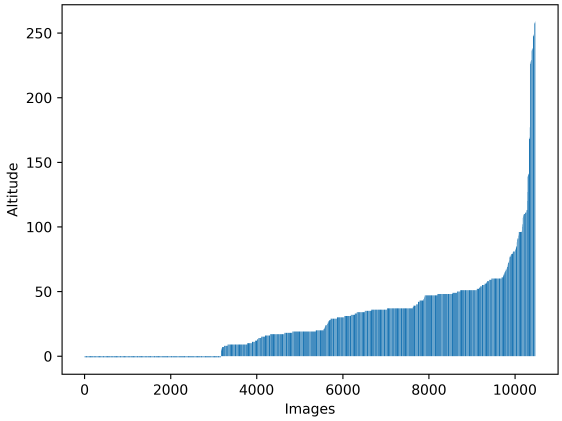
Upon careful visual inspection of apparent object size(dimensions in pixels) of (i.e., the boats) known objects, and comparing them between other images, using Equation (3.2) which is based on simplified pinhole camera model [39], we estimated that the 20M images are mostly take from altitude beyond 200 meters.

$$Real\ Height = \frac{Image\ height \times Object\ distance}{focal\ length} \tag{3.2}$$

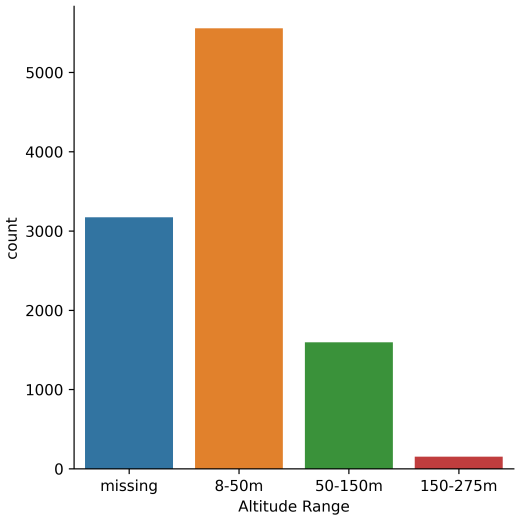
3.4 Approach 1: Dynamic Tiling

Figure 3.9a illustrates the basic premise of the dynamic tiling concept in a easy to understand manner. However, the actual implementation is a little different. Because of the way GPU's micro architectures and memory busses are designed, most CNN based deep-learning models on GPU work faster (optimally) at fixed(specific) sizes of input images. For example, in case of [40], the authors recommend input sizes that are multiple of 32 in their paper presentation. Furthermore, with a varying size of input image, it increases the chance of run-time GPU crashing due to memory overflow.

In [35], they showed an optimal/minimum image resolution $\hat{p}$ can be calculated based on altitude h, camera apparatus: FOV $\hat{\varphi}$ (degrees), target objects dimension obj, recognition criterion (minimum pixel print) rec, and proposed



(a)



(b)

Figure 3.10: Dataset Image Altitudes

the following equation:

$$\hat{p} = 2h \tan\left(\frac{\hat{\varphi}}{2}\right)\sqrt{\frac{rec}{obj}} \tag{3.3}$$

Now, using Equation (3.3), instead of directly calculating n, we can first down scale the UAV image frame to the minimum/appropriate dimension (based on DORI, Jonson's[41], [42] or some other criterion based on experiments) at any particular altitude(or altitude range). And then perform a predefined fixed-sized (based on UAV's onboard computer's memory capacity) sliding window/sliced inference. In our implementation, we are using SAHI (Slicing Aided Hyper Inference) [43] with YOLOv5 [2] backend. The same authors published SAHI [44], a vision library for large-scale object detection and instance segmentation which provides most of the tools for segmented inference and result aggregation out of the box. Figure 3.11 shows a block diagram of the implemented pipeline. Here $P_1, P_2', ..P_l$ are the tiles/patches generated by SAHI during inference.

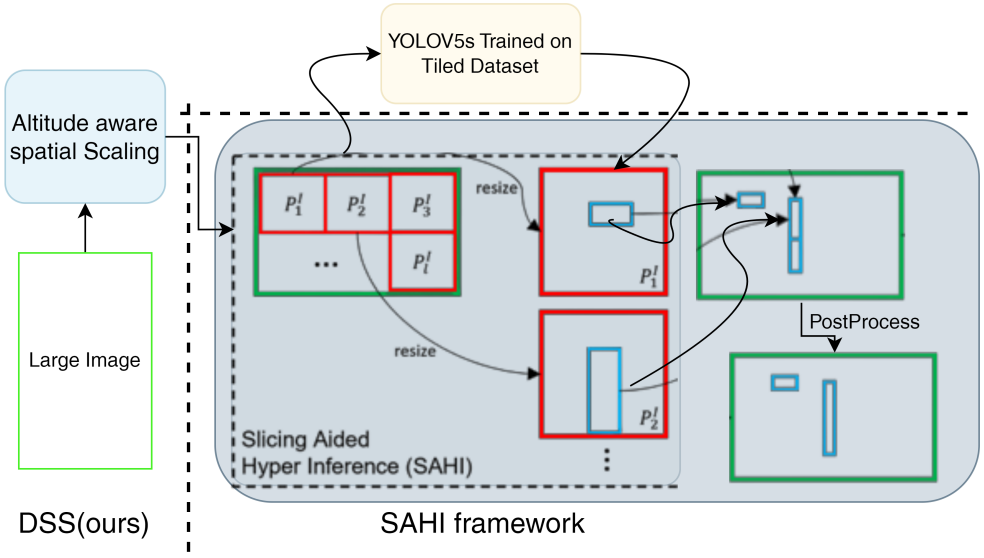


Figure 3.11: Implemented System architecture: Altitude Aware Tiled Inference. Modified from [43]

How SAHI works: The input image I is divided into NxN overlapping patches $P_1, P_2', ..P_l$. Object detection is then performed on each patch separately. The predictions from the overlapping patches are merged back into the original size using Non-Maximum Suppression (NMS). During NMS, boxes with Intersection-over-Union (IoU) ratios higher than a specified threshold T_m are matched, and detections with detection probability lower than T_d are discarded.

The authors of SAHI also maintained that, when the patch size is decreases(more tiles in same image area), it also increases the chance of larger objects not fitting entirely within a single patch, which hurts performance. We can interpret this in our context that, when an UAV is decreasing altitude, for example, homing onto

a floating asset for recovery, the apparent pixel-print of the will be increasing, therefore, simple fixed number tiling will hurt performance.

To mitigate this problem, in this experiment, we will investigate the efficacy of our proposed altitude aware Dynamic Spatial Scaling(DSS) using the SAHI framework.

3.4.1 Dataset Preprocessing

As the ground truth annotations for the *test set* is withheld by the organizers, we skipped the original *test* set, merged the train and validation sets and created a new train-val-test split (70%-20%-10%) using Roboflow [36]. A detailed overview of the dataset used is attached in Appendix A.

Training sets Preprocessing

For the training and validation sets, the images were tiled in 4(2x2) images without applying any other augmentations using Roboflow. Then we filtered the generated images so that at least 80% of the image segments contains objects. Figure 3.12 shows the result of the described steps. Note that this process also generated tiled test sets which was discarded.

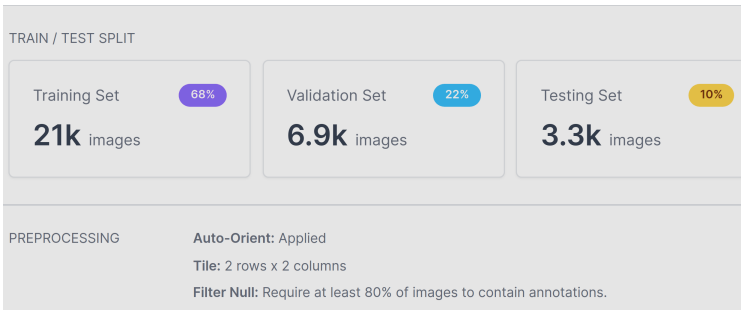


Figure 3.12: Preprocessing of Training sets

Test set Preprocessing

First we generate Table 3.1 using Equation (3.3) for an object of size 0.48 m^2 (typical human size, 0.3m x 1.6m) and considering the detection criterion to minimum 80 $pixels^2$ as suggested on [35]. Although, later we will see why this is not a universal criterion and depends on application domain and model architecture, for this experiment, we will continue using this as the base line.

Since our dataset is not uniform and contains images of different sizes and qualities, and to simplify things, we scale the original images (of our test set) into multiple fixed bins based on Table 3.1 but constrained by actual image quality/resolution and generate two test sets. Table 3.2 shows the details of the produced two test set's after image scaling.

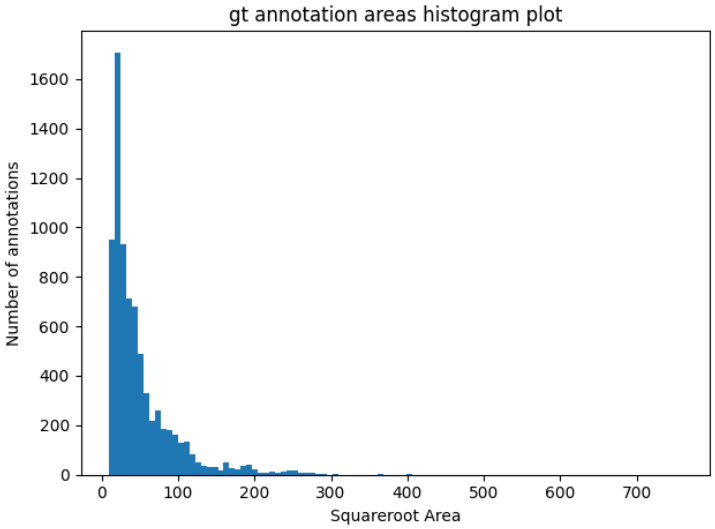
Table 3.1: Minimum image sizes (in pixels) at different altitudes

Altitude (meters)	$\hat{p}$ (trinity 20M) FOV 73°	$\hat{p}$ (DJI) (MAVIC 4K) FOV 65°	$\hat{p}$ (trinity) (Multispectral RGB) FOV 50°
10	191	164	95
20	382	328	190
30	573	493	285
.	.	.	.
.	.	.	.
240	4585	3947	2284
250	4776	4112	2379

Table 3.2: Test Set Image Size Scaling (in pixels)

Image source	Image Size	Altitude	Scaled V1	Altitude	Scaled V2
Trinity	1330x930	0-50	640x480		
Multispectral		50-150	1152x864	0-150	1152x864
RGB		150+	1664x1248	150+	1664x1248
DJI Mavic	3840x2160	0-50	640x360	0-30	640x360
4K		50-90	1152x648	30-60	1152x648
		90-110	1664x936	60-90	1664x936
		110-130	2176x1224	90-120	2176x1224
		130-150	2688x1512	120-150	2688x1512
Trinity (20M)	5456x3632	missing	3712x2470	missing	4224x2816

Figure 3.13 shows the object area(pixel-print) comparison between the original and the two scaled versions of the test set. From Figure A.1, we know that most of our annotated objects are from *swimmer* class, hence small objects. However, from the annotation histogram in Figure 3.13b, we can see that around 40% of the annotations in the original dataset (test set) are in medium group. This indicate high scale variance of apparent object sizes in the pictures. After scaling (DSS), We can see that most of the objects are now small objects and the *small-medium-large* distribution did not change much between our two test sets, and the pixel-print distribution is more inline with the physical size of the objects.



(a) Original

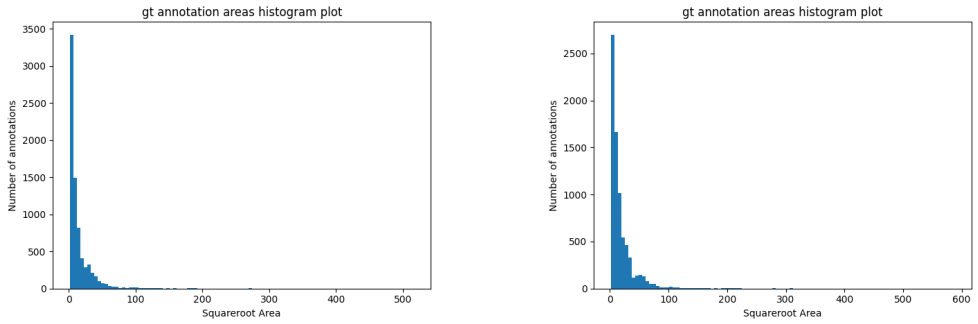
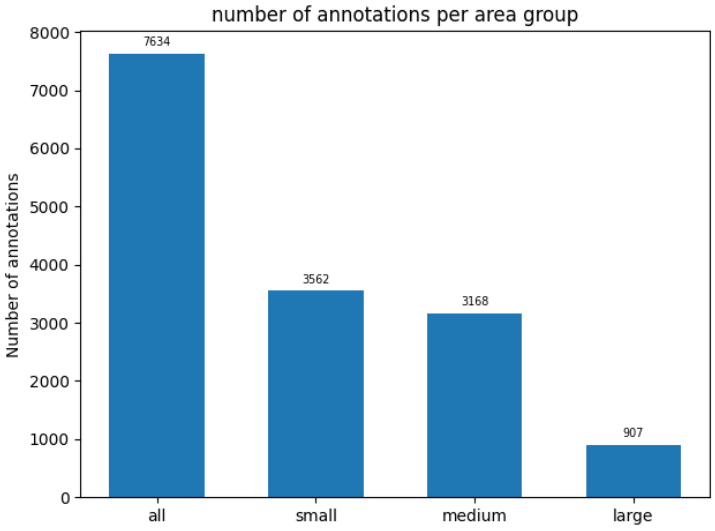


Figure 3.13: Test set's annotation area. Left: Scaled v1, Right: Scaled v2.



(b) Original

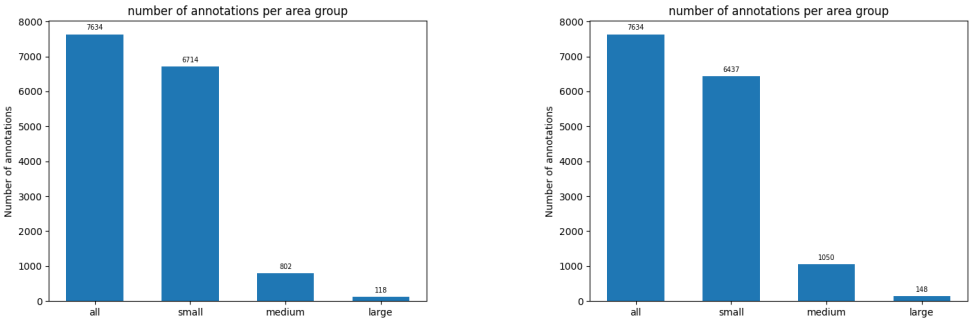


Figure 3.13: Test set's annotation area. Left: Scaled v1, Right: Scaled v2.

3.4.2 Model Selection

We selected YOLOv5 [2] as our object detection model for this experiment. This could be done using any of the SOTA object detection models, but we decided to use YOLOv5 as it is one of the best OD models. It is actively maintained, easy to implement, supports a variety of OD tasks including Bounding box detection, classification, segmentation and most importantly, has built-in support for SBCs like Raspberry Pi and NVIDIA Jetson for field deployment in the future.

3.4.3 Training

We are using the s (small) variant of YOLOv5 model, with the official Checkpoints (predefined weights with 300 epochs on COCO dataset) named "YOLOv5s". Figure 3.14 shows benchmarks for some of the official models.

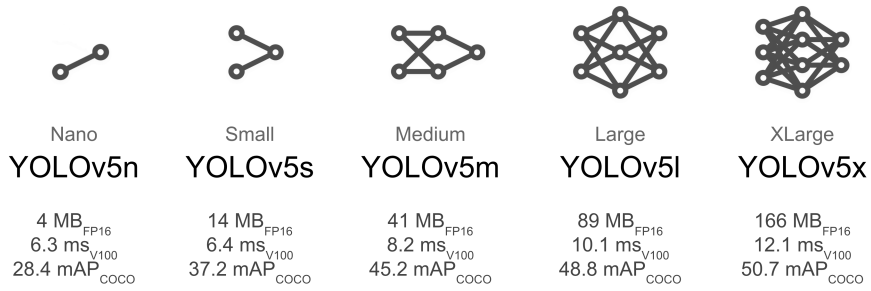


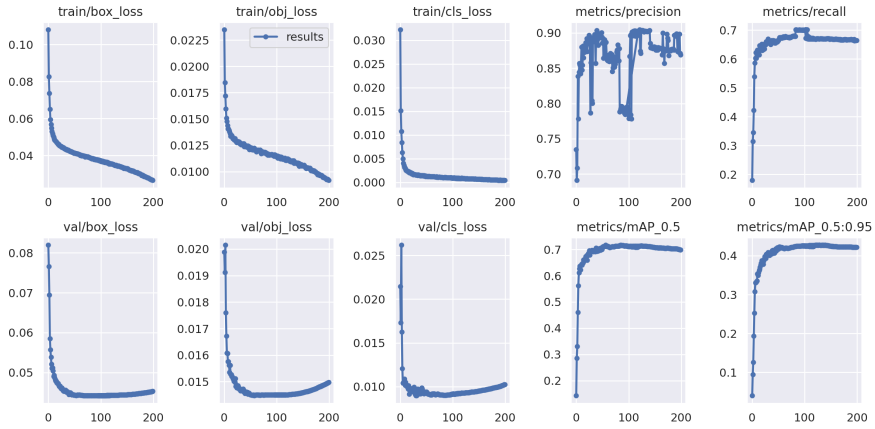
Figure 3.14: YOLOv5 official (pre-trained) models, figure taken from [2]

We trained the pre-trained yolov5s model for 200 epochs using the default configuration at 640 pixels resolution (image's longest side resized at 640 keeping aspect ratio). Since our training datasets are 2x2 tiled, essentially, we trained the model at 1280p resolution of the input images(regardless of their source) and 2x scale of their original pixel-print in the source images. Although, tiled training does not directly translate to resolution scaling, it at least enabled the model to see much finer details at the training resolution(640) which would have been lost due to down sampling, i.e., 5456x3632 images from 200+meters altitude, down sampled to 640x426 will not have any discernible features to detect the swimmers. We did not strictly (Dynamically) scale the training sets as YOLOv5 is to some extent scale invariant by design thanks to its normalized auto-anchor algorithm and mosaic augmentation. Figure 3.15d shows how the training input is augmented into mosaics of random scales. The authors claims, this helps the model to learn to recognise same objects at different scales [12].

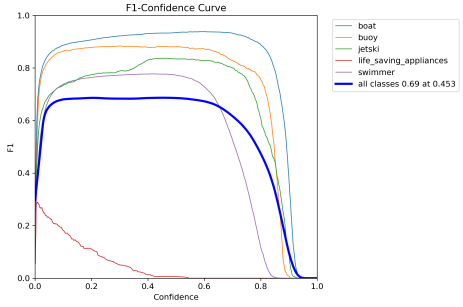
Mosaic Augmentation (YOLOv5): According to the YOLOv5 developers [2], each time an image is loaded for training, ***online imagespace and colorspace augmentations*** are applied in the trainloader (training dataloader) to present a new and unique augmented Mosaic (original image + 3 random images). Mosaics are created randomly ensuring that images are never presented twice in the same

way.

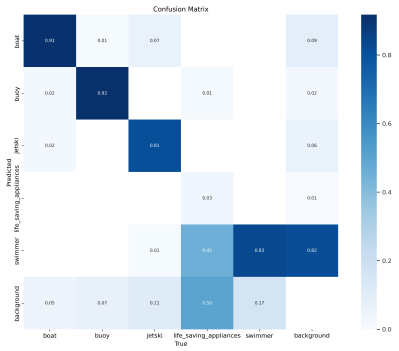
Training statistics: Figure 3.15 shows the taring matrices over 200 epochs. The F1-confidence graph in Figure 3.15b looks good for all classes except "life-saving-appliances" and we will discuss about it further in the discussion section. However, if we look at the confusion matrix Figure 3.15c, we see that our model is also detecting a lot of false positives for "swimmer" class, one of the smallest classes in terms of object size. The consequences of this will be apparent in Section 4.1.



(a) Training epochs

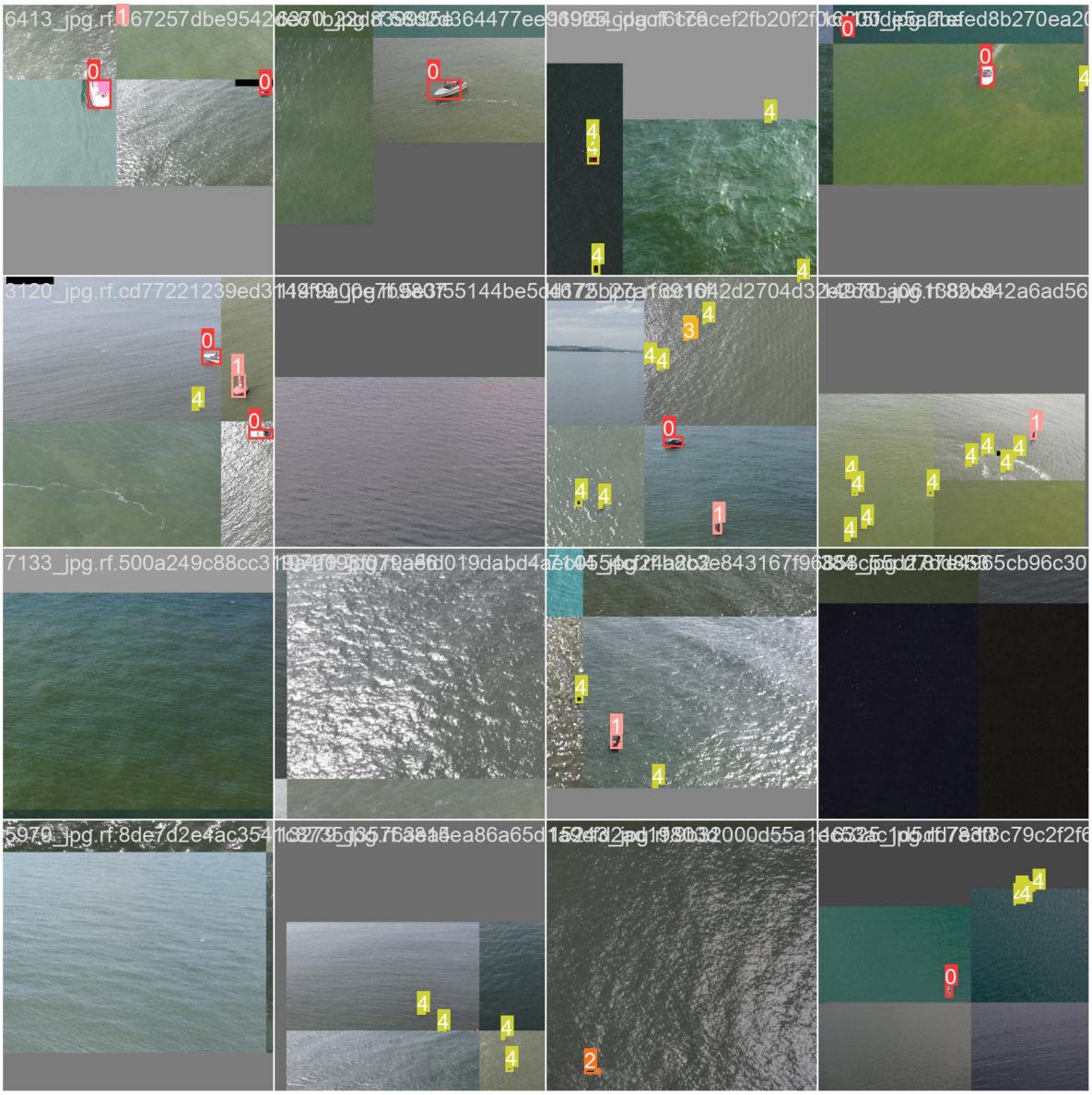


(b) F1-confidence score(training)



(c) Confusion Matrix(training)

Figure 3.15: YOLOv5s Training statistics (200 epochs). $mAP50 = 0.69, mAP_0.5 : 0.95 = 0.45$ on validation set



(d) Training time Mosaic augmentation(Exhibit: 16 samples from training batch 1)

Figure 3.15: YOLOv5s Training statistics (200 epochs)

3.5 Approach 2: Dynamic Patching using Anomaly Detector

This experiment will focus on the first stage/part of the proposed 2-stage small object detection pipeline (Figure 3.16).

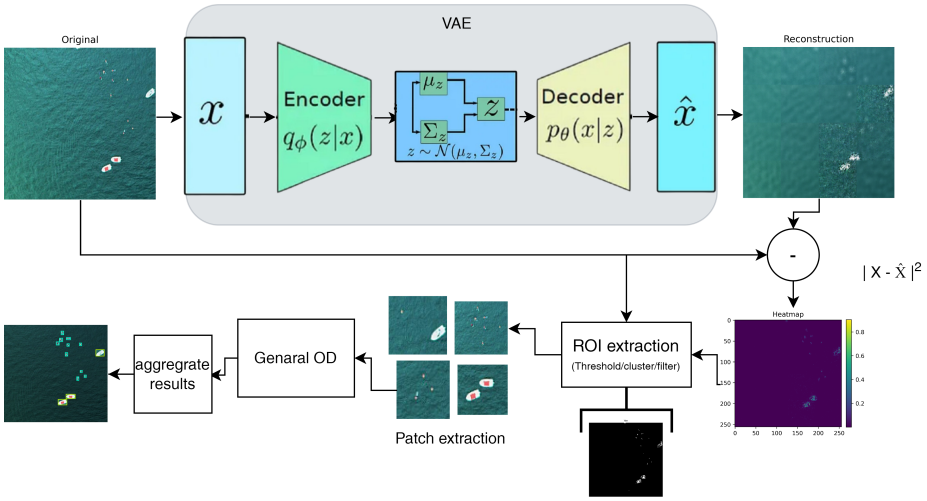


Figure 3.16: VAE Anomaly detector based two stage Detection.

The aim of this experiment is to train an anomaly detector with "background" images of marine aerial view and use it to identify objects as "anomalies" in the scene. The intuition is that since we have a scarcity of maritime aerial image datasets and in those datasets, objects are sparse in the spatial dimension, meaning, we have (relatively) a lot more information about the water surface than the target objects. Due to the nature of this outlier detection algorithm, the semi-supervised approach can also solve or at least ameliorate one of the most hindering challenge of Maritime Deep-learning application, not having annotated dataset.

3.5.1 Experiment Setup

The original dataset is in COCO format.

Dataset Pre-Processing

First, the dataset was converted into YOLOv5 format using a Python script and then uploaded into Roboflow. All the images contain at least one object in them, but sparsely distributed (most of the image area is background). Therefore it was faster and easy to just slice/cut the images into pieces to get some images with no objects. The images were tiled into 2x2 blocks (same as Section 3.4.1). Then we divided the tiled images into three sets: **i.** Inlier set (Background Images containing only water surface), **ii.** Outlier set (Images containing at least on object) and **iii.** Mixed set (containing both background and images with objects).

3.5.2 Model Description(VAE anomaly detector)

Alibi Detect library [45] in TensorFlow was used to streamline the training and for its dynamic thresholding features.

We trained our VAE model with only **Inlier set** so that the encoder part of the model learns how to compress the information into the latent vector which is commonly referred as "code" without loosing the most essential features and at the same time, the Decoder part learn to reconstruct the original image based on the information from the compressed latent vector. The difference between the input image and the reconstructed image is the loss of the model and the model learns to minimize it. Since we are only feeding it inlier images(background), it optimizes, or we can say learns the non-linear mapping from image space to latent space and back to image space but only for background images.

Therefore, when we feed any image containing objects, the model can not properly encode and therefore also decode that region of the image and the reconstruction loss around that portion of the image becomes significantly higher. After thresholding and a little bit post processing(smoothing/blurring/max-pooling), this reconstruction loss tensor gives us our 2D mask for object localization or segmentation depending on the application.

We can potentially feed the VAE network with the **mixed set** where we know the inlier to outlier ratio and calculate the threshold value online. Although, in our experiment, the builtin threshold function of alibi-detect did not perform that well. Presumably, this is due to the fact that it computes the threshold based on per pixel value but we provided inlier to outlier ratio based on image counts.

VAE Network

After several trials, we decided to use the following CNN based encoder-decoder network (Figure 3.17) for our VAE model considering the memory limitation on our development computer. Please note that, this may not be the best/optimal configuration for this task. An optimal network size could be determined, for example, through a grid search on the model and tuning the input size and latent size as hyper-parameters as shown in [33].

Dataloader with Input Scaling

Unlike fully convolutional networks (i.e., yolov5, FCNN, RCNN etc), the VAE has dense/fully connected layers at the end of its CNN encoder and beginning of its Decoder. The downside is that, The VAE network can not be scaled at runtime like FCNN or YOLO and the input size has to be fixed. This can be a problem if the images are from multiple source like we have in our dataset or if we want to optimally train/run our model on different computers with different memory capacity. To streamline our development process, across multiple work stations (i.e., different lab PCs with different GPU or Google Colab) we decided to go with a smaller VAE Network, only 64x64 input size. Normally, this would be too small

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 64)        3136
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 128)       131200
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 512)         1049088
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              33555456
=================================================================
Total params: 34,738,880
Trainable params: 34,738,880
Non-trainable params: 0
_________________________________________________________________
```

(a) Encoder Network

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32768)             33587200
_________________________________________________________________
reshape (Reshape)            (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 16, 16, 256)       2097408
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 32, 32, 64)        262208
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 64, 64, 3)         3075
=================================================================
Total params: 35,949,891
Trainable params: 35,949,891
Non-trainable params: 0
_________________________________________________________________
```

(b) Decoder Network

Figure 3.17: VAE model details

for the images to have any features for small objects. However, in our custom dataloader, we are chopping the images into predefined tiles/patches i.e., 2x2, 3x3, and 4x4 patches similar to the first approach (Section 3.4) but based on image resolution instead of altitude. We are then feeding the pieces to the network as a batch. On different machines, one would only change the batch size to fully utilize the GPU rams. Table 3.3 shows the scaling bins and resulting number of patches generated.

Table 3.3: Dataloader Image Scaling and patching

Input Size	Scaled Size	No. of Patches
(shorter side in Pixels)		(w/64 x h/64)
128-256	128	4
256-512	256	16
512+	512	64

This allowed us to use the same network on all devices and achieve a pseudo scalability. Figure 3.18 illustrates the input scaling and patching pipeline. On the output side, we essentially do the same in reverse and *unpatchify* the results. Unlike OD models like YOLOv5, here, we can create any (reasonable) number of patches without thinking about cutting the objects. Since our VAE model is object agnostic, it can still detect parts of the object in the patches as anomaly without performance penalty. This also allowed us to avoid overlapping patches which in turns also helps performance (inference speed).

We could also employ the altitude aware spatial scaling approach (Section 3.4) here and check whether it makes much difference. This shall be explored in a future project.

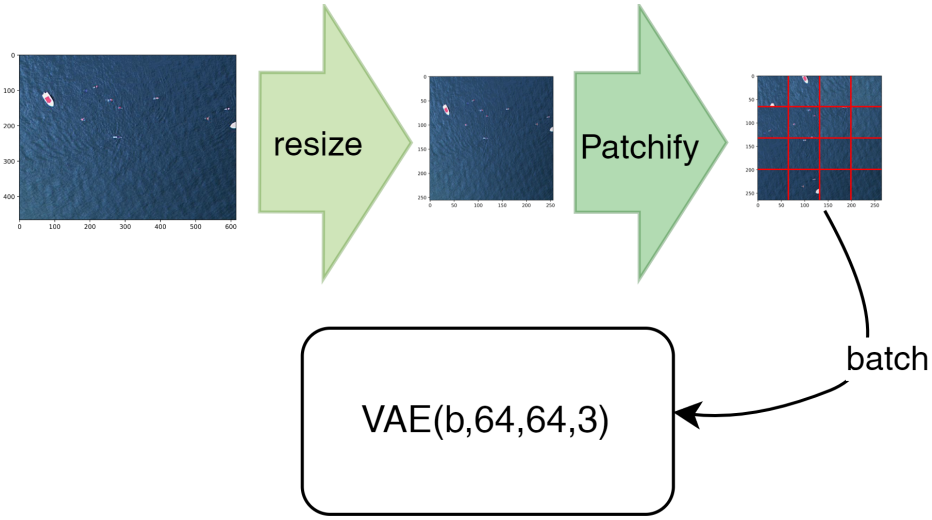


Figure 3.18: Input scaling and patched batching

3.5.3 Training

For training the model, we took 999 images from the **inlier dataset** and this generated a training set of ∼16K patches of shape (64,64,3). The images were sampled manually to make sure that the training set contains background images over a varied weather and lighting condition (i.e., water color, reflections, wave). Similarly, we sampled 125 **outlier** images for our test set and for thresholding, 275 from the **mixed** set with a 200:75 inlier to outlier ratio (∼30% images with at least one object).

We have trained the network for 50 and 100 epochs with our proposed scaling algorithm. We are also going to present some test examples without the scaling mechanism.

Test Case 1

20 epochs training with 128x128 pixels input and 1024 latent size.

Inlier samples(train+val set): 999 images of clean water of different colors(resized to input shape) Outlier sample(test set): 100 images with objects.

Test Case 2 and 3

50 and 100 epochs training with 64x64 pixels input and 1024 latent size. Patching enabled (∼16K patches).

Thresholding and filtering (ROI Extraction)

Compared max pooling, knn and different heuristic filters on the VAE results using [46] for ROI mask extraction.

Chapter 4

Results

4.1 Approach 1: Dynamic Tiling (DSS)

For benchmarking, we first ran our YOLOv5s model with the test set's original images in default model resolution (640px). This is our baseline 1 for evaluation. To compare how the model would perform in best case (without resource constraints), we also ran the model in full image resolution. Then we ran our model on the two test sets (Scaledv1 AND Scaledv2) with SAHI enabled (shown as DSS+SAHI). SAHI was configured with default parameters (640x640 pixels tiles/slices, 20% overlap). We also ran a test on the original images with only SAHI to compare inference time (average fps was calculated for all 1075 test set images) and this is our baseline 2.

Figure 4.1 shows the evaluation results for full resolution, resized(640pixels) and spatially scaled(ours) inputs. These are standard coco evaluation plots generated by coco error analysis tool. The left column shows the precision-vs-recall curve, where the area under the curves are mean average precision (mAP) over different IoU thresholds. For example, c75 means mAP at IoU=0.75. *Sim*(supercategory false positives (fps) ignored.) score can be ignored here as all our target classes were annotated under the same supercategory in the annotation files (happened during conversion from original to yolov5 format in roboflow). Or we can simply consider it as a detection score here(all target classes considered same). On the right column, we can see the same scores in a bar-chart and detailed for small, medium and large objects (according to ground truth annotation pixel areas). Table 4.1 shows the scores for all the five test set experiments and their average fps.

Figure 4.2 shows a FPS vs mAP comparison for all the test cases.

From Table 4.1, we can see that our proposed method brings significant performance gain compared to our baseline 1 score which performs poorly as expected. Most notably, for small objects, it also out performs the SAHI baseline which has the overall best performance in terms of inaccuracies. Large and medium size annotations are only a small portion of our dataset, and all models did relatively well on them. The greatest improvement our proposed method brings is in terms of in-

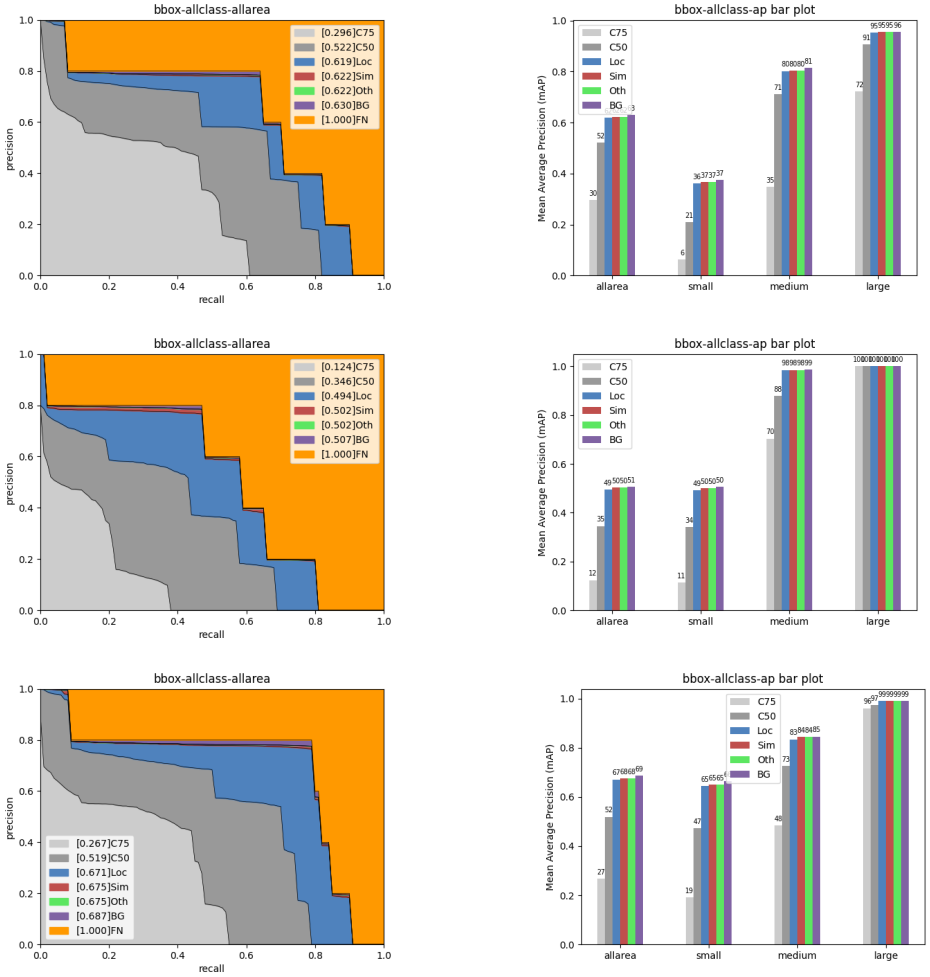
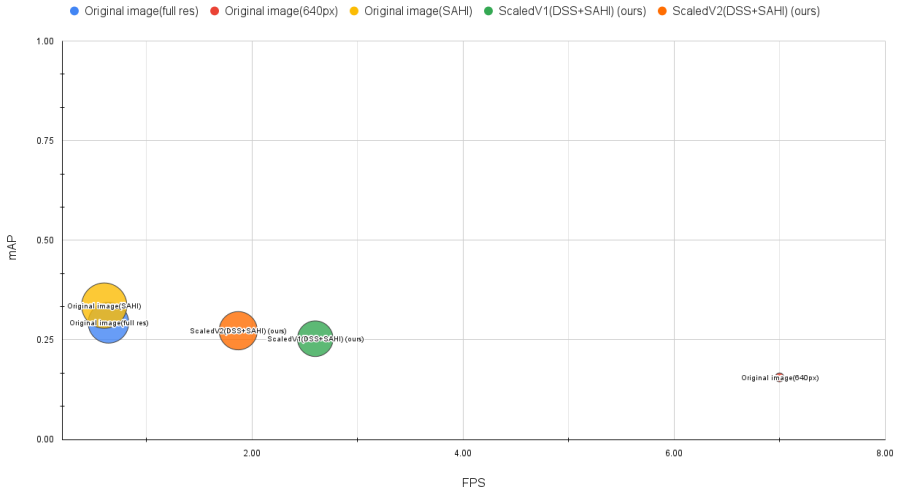


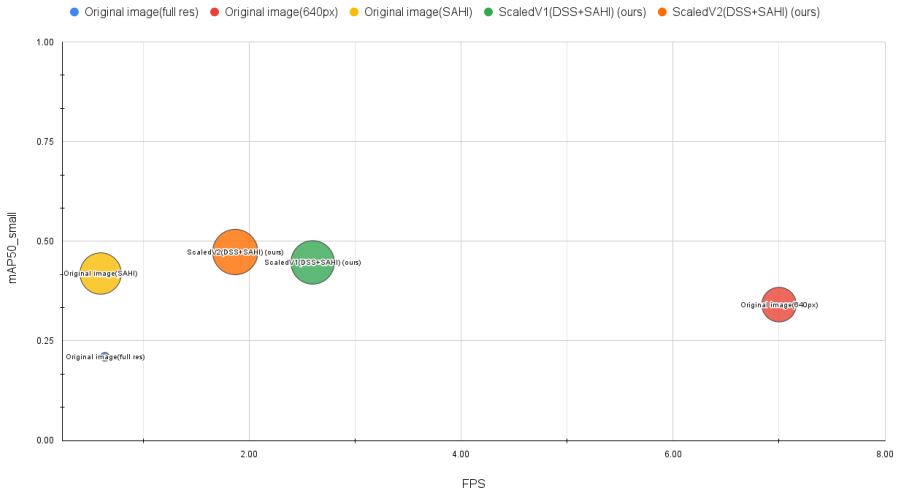
Figure 4.1: Test set performance Matrices; Original image: full resolution(top), Original image: resized @640(mid), Scaledv2: DSS+SAHI@640 (**ours**)(bot).

Table 4.1: Mean average precision values and (average)fps calculated on out test sets. The bottom two rows are scores for our proposed method. Red colored are indicative of bad performance while green remarks improvement

Setup	**mAP**	**mAP50**	**mAP50**	**mAP50**	**mAP50**	**FPS**
			small	medium	large	(avg)
Original image(full res)	0.29	0.52	0.21	0.71	0.91	0.64
Original image(640px)	0.16	0.35	0.34	0.88	1.00	**7.00**
Original image(SAHI)	**0.34**	**0.62**	0.42	0.75	0.91	0.60
ScaledV1(DSS+SAHI)	0.25	0.48	**0.45**	0.68	0.97	**2.60**
ScaledV2(DSS+SAHI)	0.27	0.52	**0.47**	0.73	0.97	1.87



(a) all sizes



(b) small targets

Figure 4.2: FPS vs mAP comparison

ference time. We can see from Figure 4.2 and Table 4.1 that while we are within comparable range in terms of mAP, or even out performing every other methods for Small objects, we are doing it in 269% and 374% faster than SAHI. We also see a positive co-relation with larger recognition criterion (minimum pixel print) and greater accuracy, albeit at the cost of slower speed. ***ScaledV2*** performs slightly better than ***ScaledV1*** but also 27% slower.

4.2 Results: Anomaly Detector

For evaluating the anomaly detector, we test our model with a hand-picked set of test images from the test set ensuring a wide variety of lighting, different image source and object sizes are present. Figure 4.3 shows the preliminary result of the VAE before implementing input scaling. We can see that model is only able to reconstruct the average color of the input image and the outlier score is not that good for small objects.

4.2.1 VAE Outputs

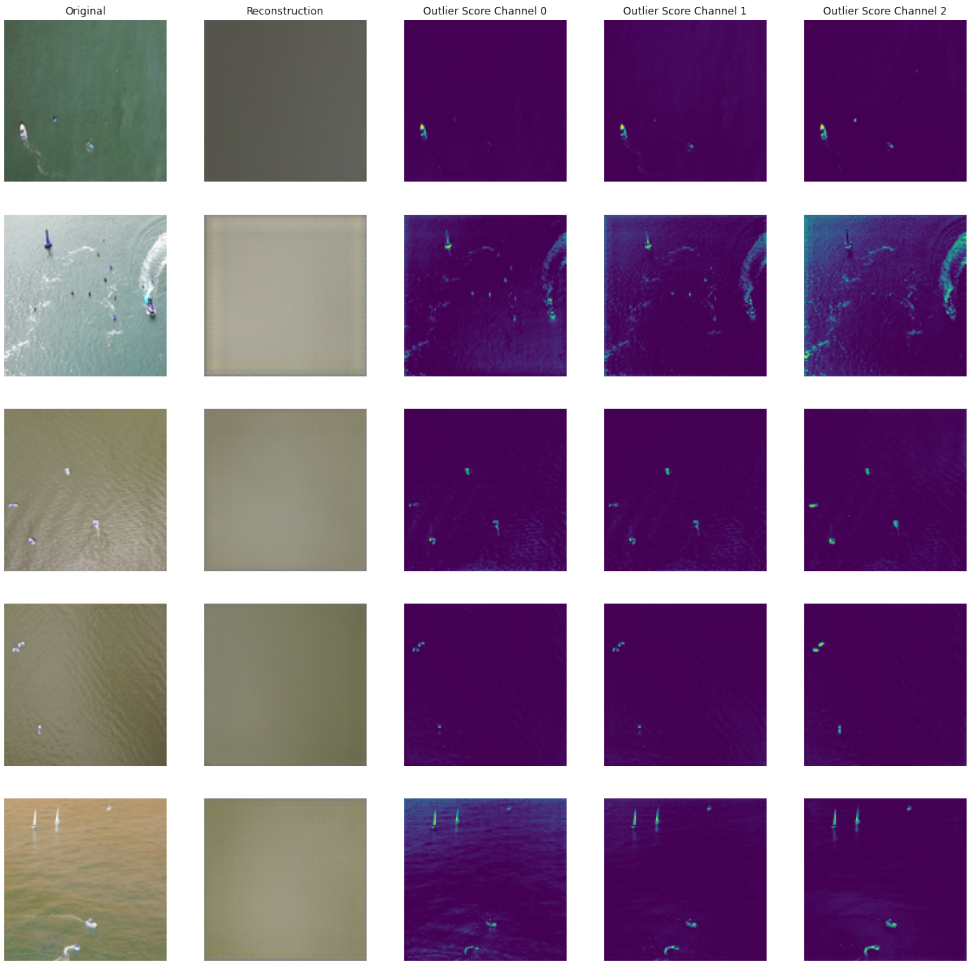


Figure 4.3: anomaly detection result for Test Case 1 (input scaling disabled, input size 128,128,3 pixels)

Figure 4.4 and Figure 4.5 shows the VAE outputs on our test images after 50 and 100 epochs of training with input scaling and patched batching enabled. we can see that the reconstruction image looks much more realistic for the inlier

part. However, we did not observe any performance difference between these two models. Both were reliably able to detect small objects or even very small part of an object (Figure 4.7 first row).

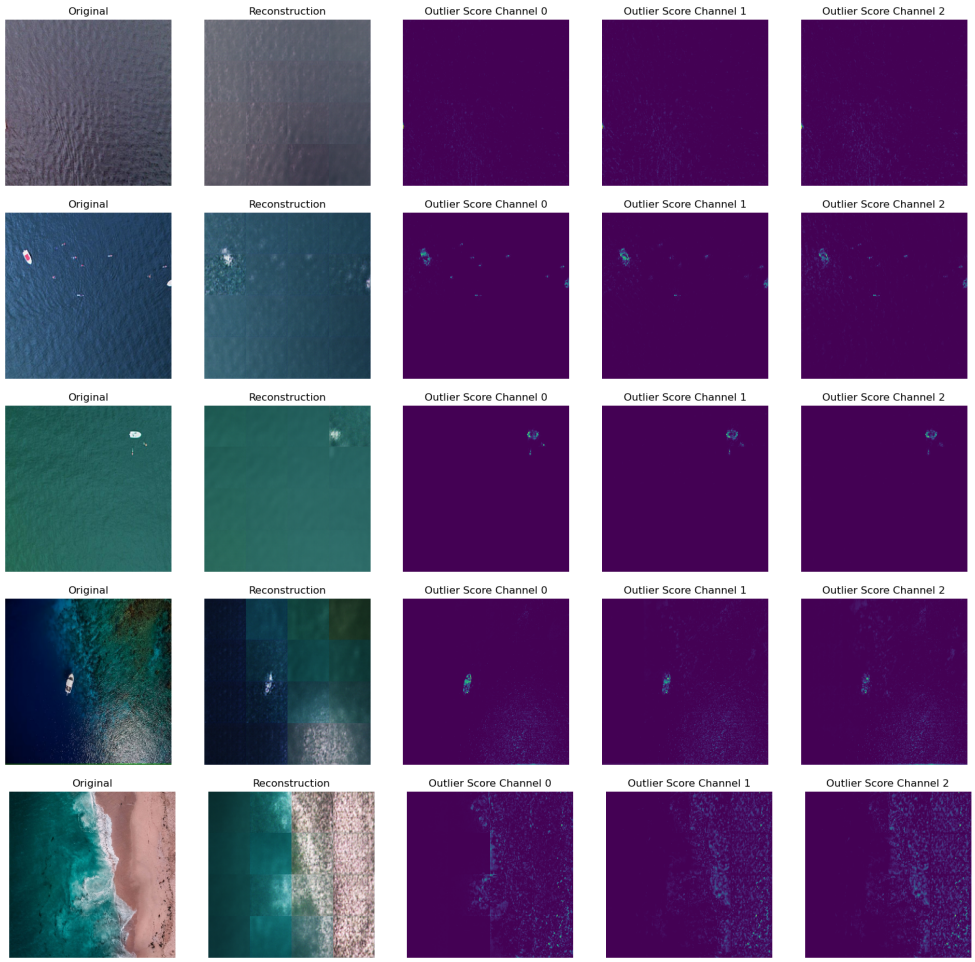


Figure 4.4: Test case2 VAE(50 epochs) results. The last two rows are images from outside the dataset to show robustness

We can also see the model was robust enough to perform relatively well on two random images from the internet.

4.2.2 Comparison with OpenCV

Figure 4.6 shows an example on two images and compares with simple OpenCV edge detection. Here we can see the justification of a computationally expensive deep learning model instead of normal heuristic filters. While our model performed consistently in two images with a very different lighting and water condition, the openCV filter(ostu) fails miserably in one of the images while being

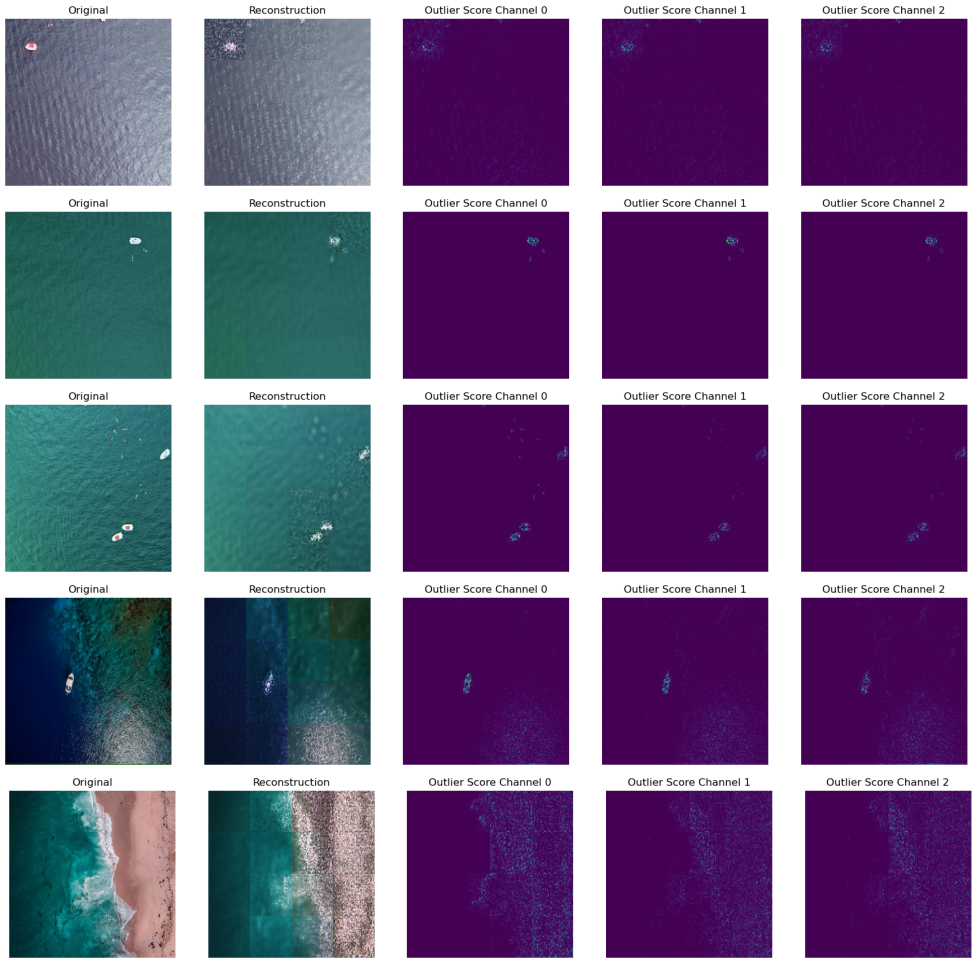


Figure 4.5: Test case3 VAE(100 epochs) results. The last two rows are images from outside the dataset to show robustness

almost perfect on the 2nd image. This goes to show that, huristic approaches are cheaper and faster but does not adopt well in dynamic conditions.

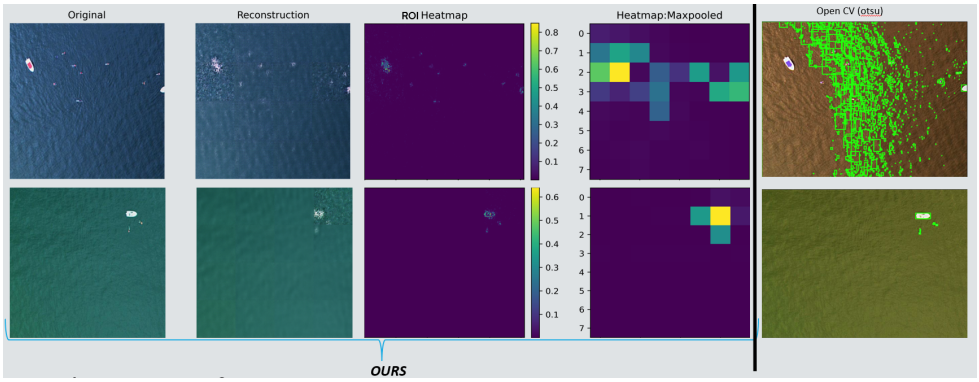


Figure 4.6: Comparison of our VAE model(100epochs) with OpenCV(otsu) filter(right-most column)

4.2.3 ROI extraction

We did not implement the complete pipeline for the two stage object detection, however propose a very simple and inexpensive max-pooling operation for extracting ROI coordinates. We show the result of our simple ROI extraction method using Maxpooling in Figure 4.7. For an object detection task, this should be enough to extract image coordinates of high probability locations or hot-spots. We can later decide an optimal patch size based on our altitude aware scaling functions depending on target object size and camera specifications.

Thresholding and Filters for post-processing

In Figure 4.8 we show a comparison for Maxpooling vs k-NN clustering and Median blurring. Notice that while k-nn is bringing in some noise suppression, median blurring is suppressing a lot of noises, but it is also pulling the outlier peak down.

Binary Mask extraction Using OpenCv filters

Figure 4.9 shows sample outputs of different OpenCV thresholding filters which can be useful in case of segmentation based tasks. In our tests, we see that ***Yen*** and ***Triangle*** performs most consistently across different examples. However, Triangle method often generates more false positives than Yen and Yen often misses small objects.

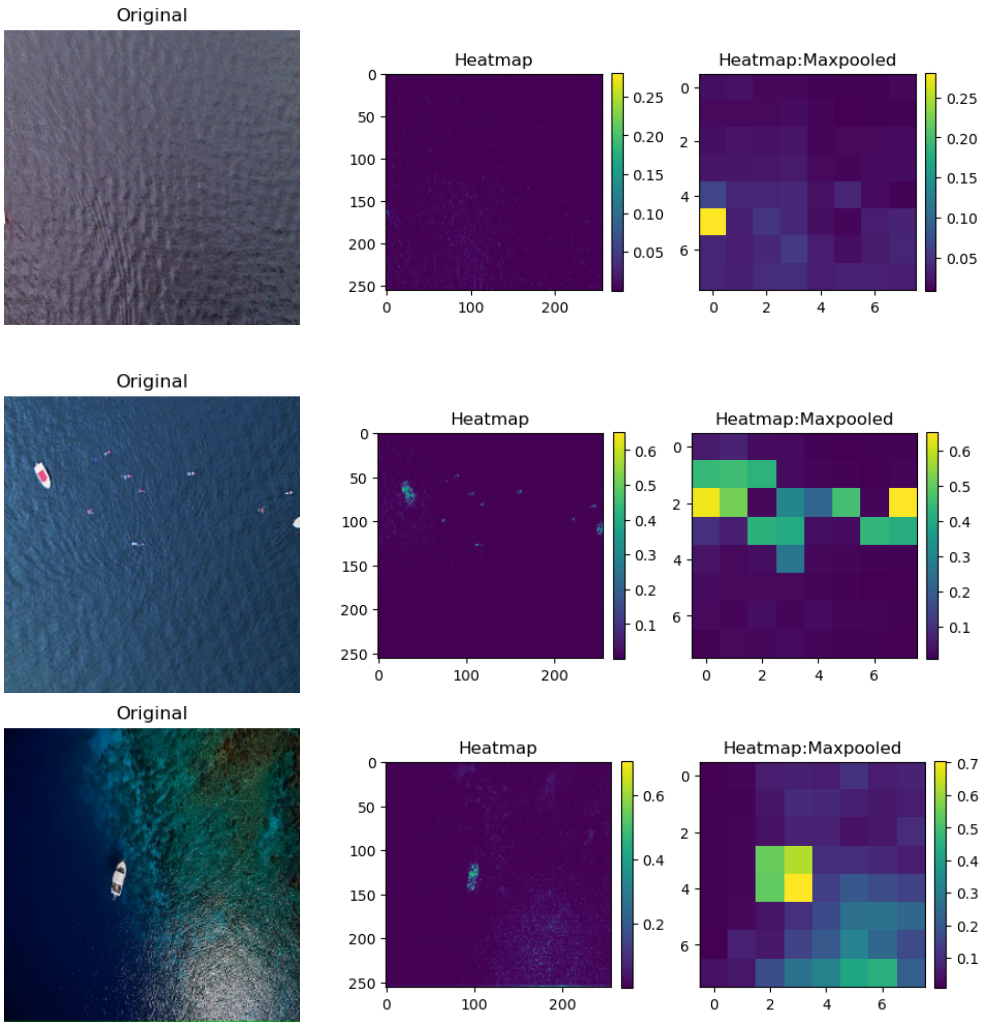


Figure 4.7: ROI heat-map generation using Max-pooling

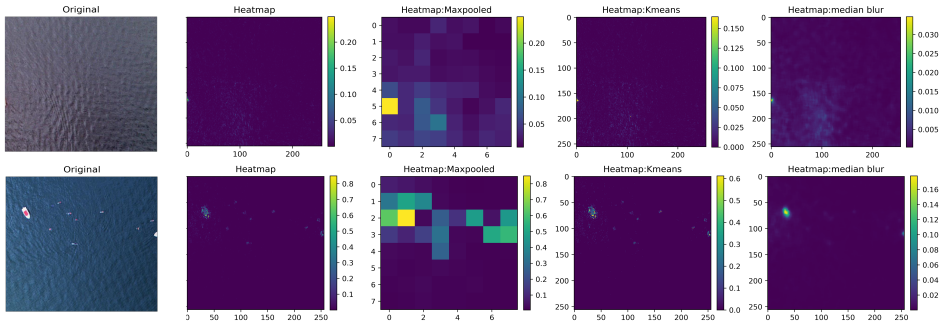


Figure 4.8: Comparing max-pooling vs k-NN clustering vs Median blur(used in [33]) for noise/artifact suppression

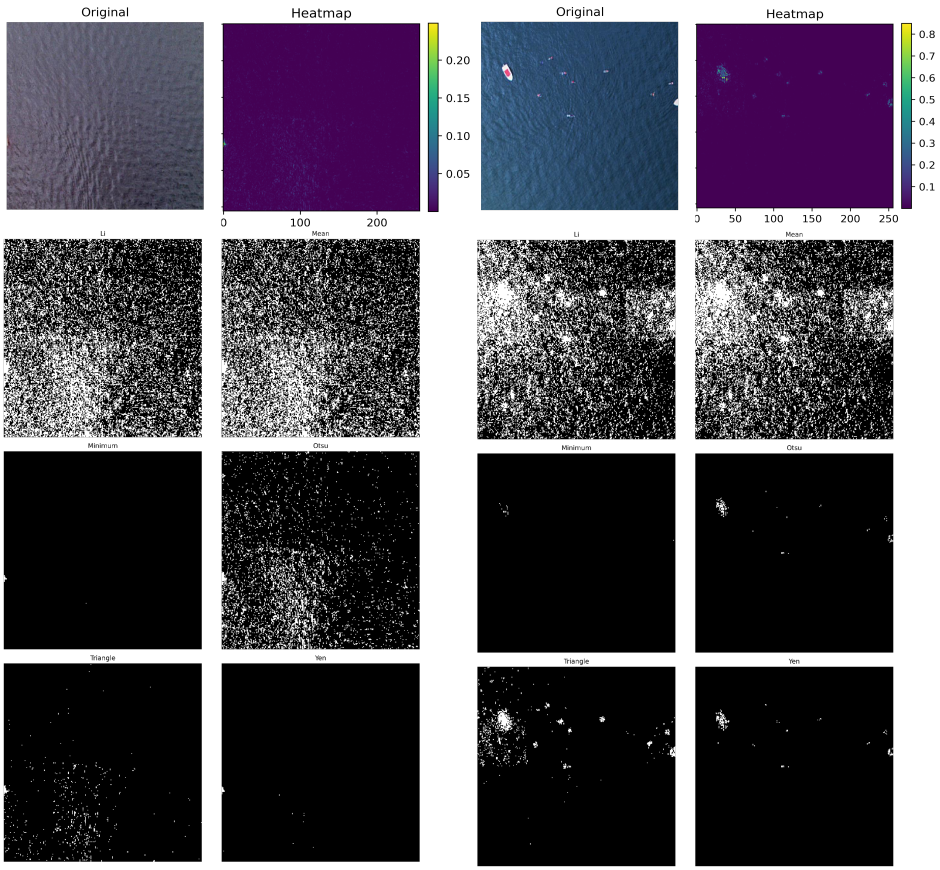


Figure 4.9: Visual comparison of different thresholding filters in OpenCV for binary mask extraction

Chapter 5

Discussion

5.1 Approach 1: Result analysis

In our experiment we were able to clearly demonstrate the efficacy of our proposed altitude aware dynamic spatial scaling and the benchmark results were in per with our hypothesis. However, we could not compare them against other papers as we have customized our dataset and re-balanced the test-val-train split. Therefore we only highlight the relative performance gain of our approach rather than comparing with SOTA benchmarks. This is also because of the fact that a good deep learning model needs careful hyper parameter tuning to achieve best performance. Since we were dealing with several architecture, it would simply be too time and resource consuming to individually tune the detection models for each approach and then benchmark. To save time and resource exhaustion, we trained the YOLOv5s model with default configuration and relatively small input resolution. This has lead to the overall low mAP score for all the benchmarks. Revisiting the training statistics and training confusion matrix, we know that our detection model had bad score for *swimmers* and *life saving appliances* classes. This is the reason our best test scores are still far below SOTA benchmarks which one might expect from a SOTA model. However, as our intention was to not give any particular algorithm advantage over the others, conduct a fair apple to apple comparison, and show a algorithmic supremacy under similar constraints, we believe it was a fair experiment. Another point to note that, our dataset, although one of the best (of the very few) maritime object detection datasets out their, the mixture of multiple image sources of very different image quality, class imbalance, missing metadata, samples with redacted pixels, altitude imbalance all has contributed adversely in the development process and goes to show that there is a huge lack of dataset choices for Maritime computer vision development.

5.2 Approach 2: Result analysis

In this experiment, we wanted to explore an alternative way for maritime object detection. Our experiments shows promising results and a clear indication of potentials. Although this might not be applicable for real-time missions on UAVs due to high computational requirement, in some offline or ground-linked applications, this approach can mitigate the lack of supervised learning's biggest hurdle, annotated data.

The authors of our dataset did not intend it for segmentation based applications and we did not have ground truth annotations for such application. Therefore, we did only manual benchmarking in the result analysis and did not provide any typical benchmarking matrices. Furthermore, our pseudo input scaling mechanism, which steamed from the development hurdle, proved to be a simple and effective tool. This shall be further investigated and we should also combine our Dynamic scaling technique with this approach to see if any potential benefit comes out of it. We kept the VAE model relatively simple, and did not do any hyper parameter search for optimal latent code size. This can potentially increase performance and reduce memory overhead. As this was an additional exploration on top of the first approach, we only intended for this experiment to be a preliminary investigation for a future work. However, the promising results from this relatively non-exhaustive experiments warrants a through investigation in the future.

Chapter 6

Conclusion

The overall goal of this thesis was to research the fundamental challenges with maritime small object detection and investigate possible solution with deep learning based approaches. We proposed two high level object detection pipelines and through extensive experiment, proposed a novel altitude aware dynamic spatial scaling technique that allows us to use SOTA deep learning framework for maritime aerial object detection. In our experiment including one small field trial, we were able to show significant performance gain while solving the constraint on memory requirement for field applications. Our proposed algorithm was able to beat SOTA frameworks by a clear margin in our benchmarks. We were also able to increase detection accuracy for small objects while reducing detection time (fps gain) and memory overhead.

Additionally we also investigated a semi-supervised anomaly detection based object detection approach which addresses the lack of annotated dataset in maritime context. We were again able to show positive outcome. As a byproduct, we also developed a input patching/tiling mechanism that gives us a pseudo scalability on rigid deep learning models like the VAE used in this thesis. This also improved our models ability to detect anomaly(object) even for very small objects on aerial images.

6.1 Future Work

The first natural continuation of this thesis would be to develop the architecture further and conduct field experiments with SBCs. In this thesis, to maintain consistency and comparability, we only used one model (YOLOv5), we should also explore other deep-learning models. As our proposed framework is architecture agnostic, there is an opportunity to integrate with SOTA vision libraries or develop a standalone platform like SAHI. We should also investigate this technique to general aerial vision application as this solves or at least paves a way to handle a fundamental challenge regarding small object detection on real-time application.

Furthermore, in the maritime context, we need to explore our anomaly based object detection approach with thermal and hyper-spectral images as most field

application in the recent time are equipped with these advanced cameras but deep-learning based research is still lacking behind other areas. Although this was part of our intended area of investigation of this thesis to explore deep learning algorithm with thermal+visual light imagery, due to unavailability of a dataset in due time, we could not explore this area. This would be an important investigation to do qualitative analysis whether sophisticated deep learning algorithm brings significant performance gain in thermal maritime images or heuristic approaches are still superior.

Furthermore, a lot of practical challenges needs to be addressed before using these methods in field applications.

Bibliography

[1] M.-T. Pham, L. Courtrai, C. Friguet, S. Lefèvre, and A. Baussard, "Yolo-fine: One-stage detector of small objects under various backgrounds in remote sensing images," *Remote Sensing*, vol. 12, no. 15, 2020, ISSN: 2072-4292. DOI: 10.3390/rs12152501. [Online]. Available: https://www.mdpi.com/2072-4292/12/15/2501.

[2] G. Jocher, *Yolov5 by ultralytics*, version 7.0, 2020. DOI: 10.5281/zenodo.3908559. [Online]. Available: https://github.com/ultralytics/yolov5.

[3] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "Sod-mtgan: Small object detection via multi-task generative adversarial network," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 210–226, ISBN: 978-3-030-01261-8.

[4] J. Brownlee, *A gentle introduction to object recognition with deep learning*, en, May 2019. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[7] S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans, *Fundamentals of object tracking*. Cambridge University Press, 2011.

[8] D. Wawrzyn, *What is ground sample distance and how does it affect your drone data?* en, Mar. 2023. [Online]. Available: https://www.propelleraero.com/blog/ground-sample-distance-gsd-calculate-drone-data/.

[9] en. [Online]. Available: https://up42.com/marketplace/blocks/processing/tiling.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[11] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, *Cspnet: A new backbone that can enhance learning capability of cnn*, 2019. arXiv: 1911.11929 [cs.CV].

[12] F. Dadboud, V. Patel, V. Mehta, M. Bolic, and I. Mantegh, "Single-stage uav detection and classification with yolov5: Mosaic data augmentation and panet," in *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2021, pp. 1–8. DOI: 10.1109/AVSS52988.2021.9663841.

[13] J. Solawetz, *What are anchor boxes in object detection?* Jul. 2020. [Online]. Available: https://blog.roboflow.com/what-is-an-anchor-box/.

[14] A. Zolich, T. A. Johansen, M. Elkolali, A. Al-Tawil, and A. Alcocer, "Unmanned aerial system for deployment and recovery of research equipment at sea," in *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*, 2021, pp. 1–8.

[15] G. Caze. "Tidewise e repsol sinopec brasil se unem para desenvolver sistemas autônomos para aplicação offshore - making marine operations sustainable." pt. (Sep. 2022), [Online]. Available: https://www.tidewise.io/2022/09/30/tidewise-e-repsol-sinopec-brasil-se-unem-para-desenvolver-sistemas-autonomos-para-aplicacao-offshore/ (visited on 10/10/2022).

[16] F. S. Leira, H. H. Helgesen, T. A. Johansen, and T. I. Fossen, "Object detection, recognition, and tracking from UAVs using a thermal camera," en, *J. field robot.*, vol. 38, no. 2, pp. 242–267, 2021.

[17] F. S. Leira, T. A. Johansen, and T. I. Fossen, "Automatic detection, classification and tracking of objects in the ocean surface from uavs using a thermal camera," in *2015 IEEE Aerospace Conference*, IEEE, 2015, ISBN: 9781479953790.

[18] F. S. Leira, K. Trnka, T. I. Fossen, and T. A. Johansen, "A ligth-weight thermal camera payload with georeferencing capabilities for small fixed-wing uavs," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, ISBN: 9781479960101.

[19] J. Canny, "A computational approach to edge detection," en, *IEEE transactions on pattern analysis and machine intelligence*, vol. 8, no. 6, pp. 679–698, 1986, ISSN: 0162-8828.

[20] M.-K. Hu, "Visual pattern recognition by moment invariants," en, *IEEE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962, ISSN: 0018-9448. DOI: 10.1109/tit.1962.1057692. [Online]. Available: http://dx.doi.org/10.1109/tit.1962.1057692.

[21] C. D. Rodin and T. A. Johansen, "Detectability of objects at the sea surface in visible light and thermal camera images," in *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, 2018, pp. 1–10.

[22] H. H. Helgesen, T. H. Bryne, E. F. Wilthil, and T. A. Johansen, "Camera-based tracking of floating objects using fixed-wing uavs," en, *Journal of intelligent & robotic systems*, vol. 102, no. 4, 2021, ISSN: 0921-0296. DOI: 10.1007/s10846-021-01432-z. [Online]. Available: http://dx.doi.org/10.1007/s10846-021-01432-z.

[23] H. Li, X.-J. Wu, and J. Kittler, "Infrared and visible image fusion using a deep learning framework," in *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 2018, pp. 2705–2710. DOI: 10.1109/ICPR.2018.8546006.

[24] N. Kumar, A. K. Jilani, P. Kumar, and A. Nikiforova, "Improved yolov3-tiny object detector with dilated cnn for drone-captured images," *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pp. 89–94, 2022.

[25] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. DOI: 10.48550/ARXIV.1804.02767. [Online]. Available: https://arxiv.org/abs/1804.02767.

[26] C. Jiang, H. Ren, X. Ye, J. Zhu, H. Zeng, Y. Nan, M. Sun, X. Ren, and H. Huo, "Object detection from uav thermal infrared images and videos using yolo models," *International Journal of Applied Earth Observation and Geoinformation*, vol. 112, p. 102 912, 2022, ISSN: 1569-8432. DOI: https://doi.org/10.1016/j.jag.2022.102912. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569843222001145.

[27] X. Wang, X. Shu, S. Zhang, B. Jiang, Y. Wang, Y. Tian, and F. Wu, "Mfgnet: Dynamic modality-aware filter generation for rgb-t tracking," *IEEE Transactions on Multimedia*, 2022.

[28] Z. Tang, T. Xu, and X.-J. Wu, "A survey for deep rgbt tracking," 2022. [Online]. Available: http://arxiv.org/abs/2201.09296.

[29] L. Yang, R. Ma, and A. Zakhor, *Drone object detection using rgb/ir fusion*, 2022. DOI: 10.48550/ARXIV.2201.03786. [Online]. Available: https://arxiv.org/abs/2201.03786.

[30] Z. Tu, Y. Ma, Z. Li, C. Li, J. Xu, and Y. Liu, *Rgbt salient object detection: A large-scale dataset and benchmark*, 2020. DOI: 10.48550/ARXIV.2007.03262. [Online]. Available: https://arxiv.org/abs/2007.03262.

[31] Z. Tu, Z. Li, C. Li, and J. Tang, "Weakly alignment-free rgbt salient object detection with deep correlation network," *IEEE Transactions on Image Processing*, vol. 31, pp. 3752–3764, 2022. DOI: 10.1109/TIP.2022.3176540.

[32] J. Walker, T. Yamada, A. Prugel-Bennett, and B. Thornton, "The effect of physics-based corrections and data augmentation on transfer learning for segmentation of benthic imagery," in *2019 IEEE Underwater Technology (UT)*, 2019, pp. 1–8. DOI: 10.1109/UT.2019.8734463.

[33] S. Bijjahalli, O. Pizarro, and S. B. Williams, *A semi-supervised object detection algorithm for underwater imagery*, 2023. arXiv: 2306.04834 [cs.CV].

[34] F. Ö. Ünel, B. O. Özkalayci, and C. Çiğla, "The power of tiling for small object detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 582–591. DOI: 10.1109/CVPRW.2019.00084.

[35] P. Petrides, C. Kyrkou, P. Kolios, T. Theocharides, and C. Panayiotou, "Towards a holistic performance evaluation framework for drone-based object detection," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 1785–1793. DOI: 10.1109/ICUAS.2017.7991444.

[36] B. Dwyer, J. Nelson, and J. Solawetz, *Roboflow*, 2022. [Online]. Available: https://roboflow.com.

[37] L. A. Varga, B. Kiefer, M. Messmer, and A. Zell, "Seadronessee: A maritime benchmark for detecting humans in open water," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2260–2270.

[38] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[39] P. J. A. Alphonse and K. V. Sriharsha, "Depth perception in single rgb camera system using lens aperture and object size: A geometrical approach for depth estimation," *SN Applied Sciences*, vol. 3, no. 6, p. 595, May 2021, ISSN: 2523-3971.

[40] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016. arXiv: 1612.08242 [cs.CV].

[41] J. Johnson, "Analysis of image forming systems," *Selected papers on infrared design. Part I and II*, vol. 513, p. 761, 1985.

[42] *Dori (detection, observation, recognition, identification) how to measure how far a surveillance camera can see?* [Online]. Available: https://www.infinitioptics.com/whitepapers/dori-detection-observation-recognition-identification.

[43] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 966–970, 2022. DOI: 10.1109/ICIP46576.2022.9897990.

[44] F. C. Akyon, C. Cengiz, S. O. Altinuc, D. Cavusoglu, K. Sahin, and O. Eryuksel, *SAHI: A lightweight vision library for performing large scale object detection and instance segmentation*, Nov. 2021. DOI: 10.5281/zenodo.5718950. [Online]. Available: https://doi.org/10.5281/zenodo.5718950.

[45] A. Van Looveren, J. Klaise, G. Vacanti, O. Cobb, A. Scillitoe, R. Samoilescu, and A. Athorne, *Alibi detect: Algorithms for outlier, adversarial and drift detection*, version 0.11.2, Apr. 28, 2023. [Online]. Available: `https://github.com/SeldonIO/alibi-detect`.

[46] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[47] R. Ribeiro, G. Cruz, J. Matos, and A. Bernardino, *A dataset for airborne maritime surveillance environments*, 2017.

[48] B. Bovcon, J. Muhovič, J. Perš, and M. Kristan, "The mastr1325 dataset for training deep usv obstacle detection models," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019.

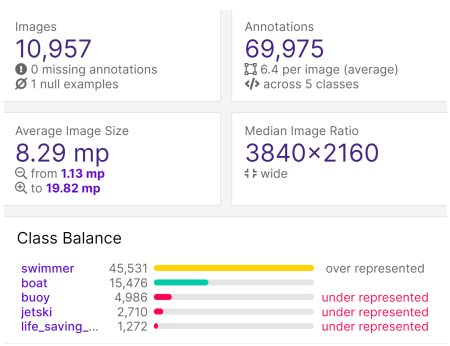
[49] D. Cafarelli, L. Ciampi, L. Vadicamo, C. Gennaro, A. Berton, M. Paterni, C. Benvenuti, M. Passera, and F. Falchi, *Mobdrone: A drone video dataset for man overboard rescue*, 2022. arXiv: `2203.07973 [cs.CV]`.

[50] D. D. Bloisi, L. Iocchi, A. Pennisi, and L. Tombolini, "Advanced video and signal based surveillance (AVSS)," in *12th IEEE International Conference on*, 2015, pp. 1–6.

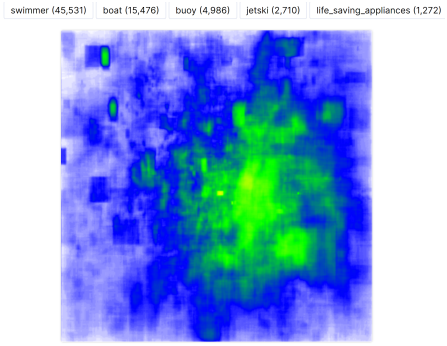
[51] S. Fefilatyev, V. Smarodzinava, L. O. Hall, and D. B. Goldgof, "Horizon detection using machine learning techniques," in *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*, Orlando, FL: IEEE, Dec. 2006.

Appendix A

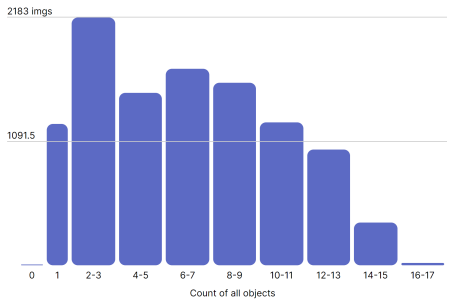
Dataset Insights



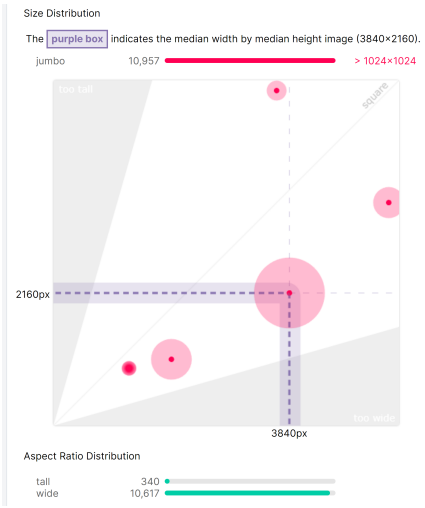
(a) Dataset Overview



(b) Annotation Heatmap (all classes)

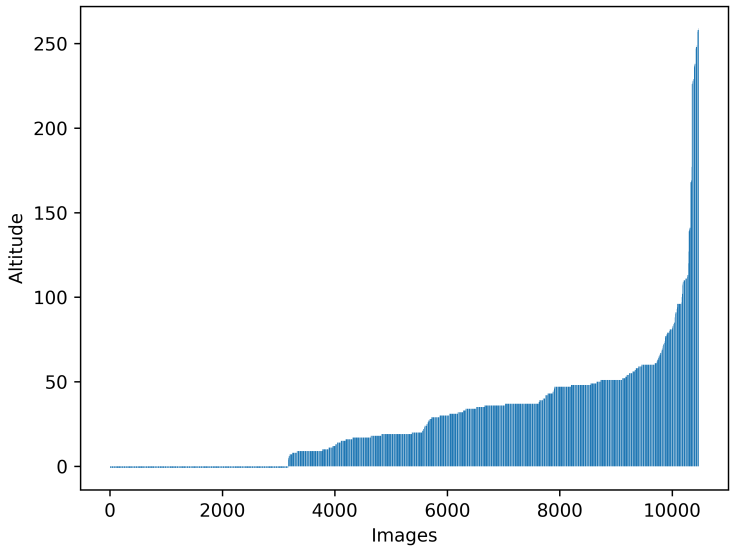


(c) Histogram of Object Count by Image



(d) Image dimensions

Figure A.1: Dataset Insights (generated by [36])



(e) Image Altitudes

Figure A.1: Dataset Insights (generated by [36])

Appendix B

Link to project Code

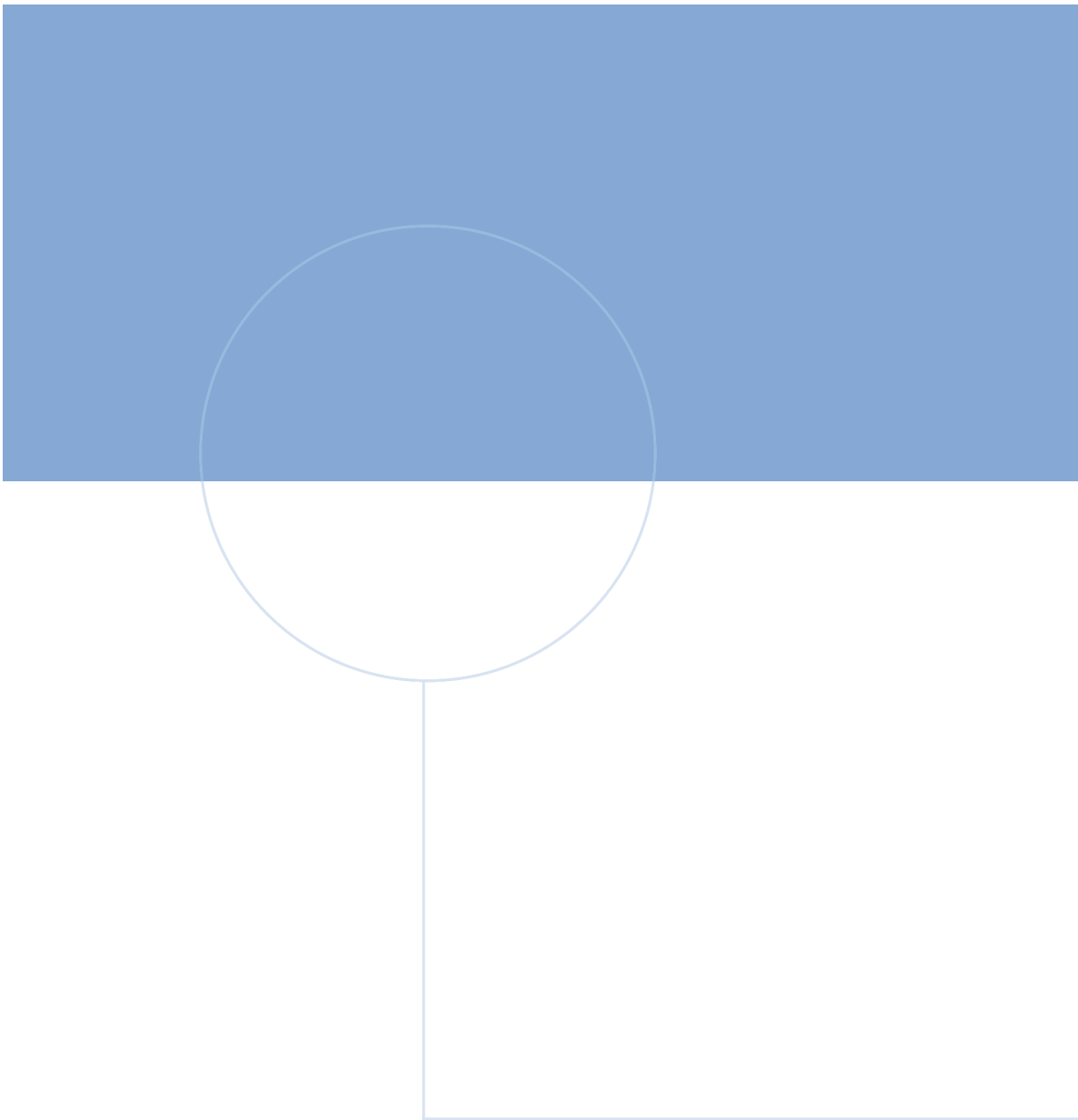
https://github.com/ahmadSum1/NTNU_mscThesis

Appendix C

Maritime Datasets Review

Over the course of this thesis and the specialization project we explored many of the contemporary computer vision datasets intended for maritime applications. Below is a list of some of the most prominent maritime datasets:

- SeaDronesSee [37]
  - Object Detection V1 and V2(Used in this thesis)
  - Single-Object Tracking
  - Multi-Object Tracking
  - Multi-Spectral Object Detection Dataset
  - Synthetic SeaDronesSee dataset
- The Seagull dataset [47]
- MODS Obstacle Detection and Segmentation Benchmark [48]
- MOBDrone: a Drone Video Dataset for Man OverBoard Rescue [49]
- Other maritime dataset
  - Horizon detection Ground Truth and videos for Mar-DCT dataset [50]
  - Horizon detection Ground Truth and videos for Buoy dataset [51]



NTNU
Norwegian University of
Science and Technology

Co-funded by the
Erasmus+ Programme
of the European Union