UAV–Based Maritime Search and Rescue Missions: An End-to-End Approach

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Martin Meßmer
aus Wangen im Allgäu

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	10.12.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Andreas Zell
2. Berichterstatter:	Prof. Dr. Thomas Linkugel

Of course it is happening inside your head, Harry,
but why on earth should that mean that it is not real?

ALBUS DUMBLEDORE

Abstract

Unmanned Aerial Vehicles (UAVs) have become important tools in various vision-based fields due to their comparably low cost, cheap maintenance, versatility, and ease of use, one of which being search and rescue (SAR) missions. Despite these advancements, automating aerial robots to enable them to detect distressed humans below or even find the most promising flight trajectory to maximize detection probability remains an open challenge.

This dissertation presents various approaches to enhancing UAV-based maritime search and rescue (mSAR) operations in the areas of computer vision, UAV path planning, and the technical implementation of SAR missions.

In the field of computer vision, we studied how domain variations, such as flight altitude or capture angle, affect the performance of object detectors. Building on that, we developed object detectors capable of overcoming these challenges. By identifying that the bird's eye view poses significant difficulties and creating a dedicated strategy to address this issue, we showed that object detectors can benefit from engineering tailored to specific domains. Addressing the shortage of data in the maritime UAV computer vision community, we have recorded and released a large-scale data set, which includes dense meta-data labels like capture-altitude and capture-angle. It features single-object tracking, multi-object tracking, and object detection. In the context of UAV flight trajectory planning we explored algorithms incorporating environmental data, such as water current and wind flow, as well as some which do not incorporate this knowledge. Based on these finding, we developed a trajectory planning algorithm based on branch-and-bound algorithms. Additionally, we investigated various techniques for efficiently predicting regions of interest onboard the drone. This enables our newly developed and published mSAR software framework to selectively stream video footage, thereby conserving bandwidth.

The experiments presented in this dissertation show that the proposed methods effectively improve the capabilities of maritime search and rescue drones, both in the field of flight trajectory planning and the detection of humans or vessels in distress. The promising nature of these results suggest interesting future research endeavors. These include, but are not limited to, exploring coordinated multi-UAV missions, incorporating capture-angle information deterministically into the detection pipeline, and expanding data sets to include more diverse scenarios or sensors.

Kurzfassung

Drohnen (UAVs) sind aufgrund ihrer vergleichsweise geringen Kosten, ihrer billigen Wartung, ihrer Vielseitigkeit und ihres einfachen Einsatzes zu wichtigen Werkzeugen in verschiedenen kamerabasierten Bereichen geworden, darunter auch bei Such- und Rettungseinsätzen (SAR). Trotz dieser Fortschritte ist es weiterhin ein offenes Problem, Flugroboter so zu automatisieren, dass sie in Not geratene Menschen erkennen oder sogar die beste Flugbahn zur Optimierung der Auffindewahrscheinlichkeit selbst planen können. In dieser Dissertation werden verschiedene Ansätze zur Verbesserung von UAV-basierten Seenotrettungseinsätzen (mSAR) in den Bereichen Computer Vision, UAV-Pfadplanung, und ihrer technische Umsetzung vorgestellt und diskutiert.

Auf dem Gebiet der Mustererkennung haben wir untersucht, wie sich Variationen in einem bestimmten Bereich, z. B. die Flughöhe oder der Aufnahmewinkel, auf die Leistung von Objektdetektoren auswirken. Aufbauend darauf haben wir Objektdetektoren entwickelt, für die diese Variationen keine Probleme darstellen. Da wir dabei festgestellt haben, dass die Vogelperspektive erhebliche Schwierigkeiten mit sich bringt, entwickelten wir eine spezielle Strategie, um dieses Problem zu lösen. Um gegen den Datenmangel in der Gemeinschaft der Forschenden in maritimer Drohnenmustererkennung zu arbeiten, haben wir einen umfangreichen Datensatz aufgezeichnet und veröffentlicht, der vollständige Metadaten wie Erfassungshöhe und Erfassungswinkel für jedes Bild enthält. Er stellt Einzelobjektverfolgung, Multiobjektverfolgung, und Objekterkennung zur Verfügung. Im Zusammenhang mit der Flugpfadplanung von UAVs untersuchten wir Algorithmen, die Umweltdaten wie Wasserströmung und Windfluss berücksichtigen, sowie solche, die dieses Wissen nicht einbeziehen. Auf der Grundlage dieser Erkenntnisse haben wir einen Algorithmus zur Flugbahnplanung entwickelt, der auf Verzweigen-und-Begrenzen-Algorithmen basiert. Außerdem untersuchten wir verschiedene Techniken zur effizienten Detektion interessanter Regionen an Bord der Drohne. Dies ermöglicht unserem neu entwickelten und veröffentlichten mSAR-Softwareframework, Videomaterial selektiv zu streamen und so Bandbreite einzusparen.

Die in dieser Dissertation vorgestellten Experimente zeigen, dass die vorgeschlagenen Methoden die Fähigkeiten maritimer Such- und Rettungsdrohnen sowohl im Bereich der Flugpfadplanung als auch bei der Erkennung von Menschen in Not oder havarierten Schiffen verbessern. Die vielversprechenden Ergebnisse zeigen interessante zukünftige Forschungsvorhaben auf. Dazu gehören unter anderem die Erforschung koordinierter Multi-UAV-Einsätze, die deterministische Einbeziehung von Aufnahmewinkelinformationen in Erkennungsalgorithmen, und die Erweiterung von Datensätzen, um mehr unterschiedliche Szenarien oder Sensoren abzubilden.

Contents

Chapter 1

Introduction

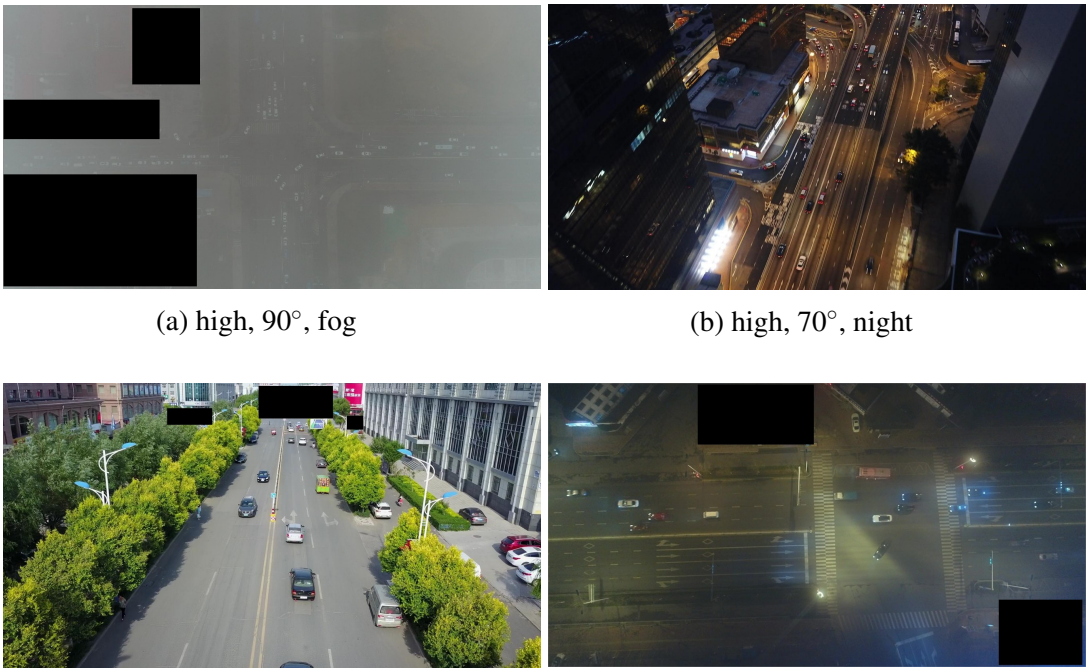
Unmanned Aerial Vehicles (UAVs) equipped with cameras have become invaluable assets in various fields, including agriculture, delivery, surveillance, and search and rescue (SAR) missions (Adão *et al.* (2017); San *et al.* (2018); Geraldes *et al.* (2019)). UAVs are particularly effective in SAR missions due to their rapid deployment and versatility, providing comprehensive scene overviews (Mishra *et al.* (2020); Karaca *et al.* (2018); Albanese *et al.* (2020)). In maritime scenarios, where vast areas must be explored and searched quickly, the efficient use of autonomous UAVs is crucial (Yeong *et al.* (2015)) for various reasons; they provide the beneficial bird's eye view similar to that of aircraft and helicopters, while causing significantly lower cost to maintain and operate. The most significant challenges in this application include the detection and localization of people in open water (Gallego *et al.* (2019); Nasr *et al.* (2019)), as well as the path planning of the UAV to maximize the probability of detecting all distressed humans (Sato (2008)).

Additionally, over the last decade, the field of deep learning research has undergone a revolution regarding its capabilities, the problems it can solve, and the applications it is applied to. This so-called 'third wave' (Emmert-Streib *et al.* (2020)) of neural networks, initiated by the introduction of AlexNet (Krizhevsky *et al.* (2012)) in 2012 led to various deep learning based breakthroughs in diverse fields, like image generation (Goodfellow *et al.* (2020); Rombach *et al.* (2022)), strategic two player games (Silver *et al.* (2016)), or natural language processing (Vaswani *et al.* (2017)), to only name a few. Most relevant to this dissertation are the advancements in computer vision, in particular object detection (Girshick (2015); Jiang *et al.* (2022)), which will be further discussed in Section 1.1 later in this chapter.

In parallel to these developments, since the beginning of the so-called refugee crisis in 2014, we have seen a steep increase of refugee movements from Africa to Europe across the Mediterranean sea (Frontex (2024)). For a variety of reasons, many of these journeys take place under perilous conditions, often resulting in tragic fatalities as boats capsize (IOM (2024)). On the other hand, the sea has become increasingly crowded due to the ongoing rise in maritime trade (UN Trade and Development (2023)), potentially also leading to more incidents at sea requiring maritime search and rescue (mSAR) for humans in distress at sea. We argue that these developments call for intelligent strategies to improve sea-monitoring efforts.

In this dissertation, we will address these developments by presenting approaches to tackle the challenges posed by the increased demand for maritime search and rescue. We will demonstrate that UAVs utilizing deep learning models on mobile GPUs can be an integral part of an mSAR pipeline. The remainder of this chapter is dedicated to giving an introduction to (UAV) computer vision and an outline of the rest of this work.

1.1 UAV Computer Vision



(a) high, 90°, fog

(b) high, 70°, night

(c) low, 30°, day

(d) medium, 90°, night

Figure 1.1: This Figure illustrates the different ways a picture from a UAV data set could look like. Below the images, the sensor and lighting conditions are displayed. The categories for the flight altitude are broad categories, as for these data sets there is no dense meta information per image available. Images are taken from UAVDT (Du *et al.* (2018)) and VisDrone (Du *et al.* (2019)).

Since computer vision in general – particularly deep learning-based computer vision – is such a heavily researched field (Papers with Code (2024b)), the question arises as to why further investigation into the topic is necessary. UAV computer vision faces distinct challenges and advantages compared to generic computer vision tasks.

One of the primary challenges in UAV object detection is the need to detect and recognize objects that can vary significantly in size due to the drone's flight altitude and viewing angle. Unlike generic computer vision tasks, which mostly operate at relatively

consistent distances and angles, UAVs can encounter objects that appear very small when viewed from high altitudes. Even when flying close to the ground, these objects will merely reach the size of the smallest objects in generic object detection data sets like COCO (Common Objects in Context) (Lin *et al.* (2014)). Figure 1.1 and 1.2 illustrate these appearance differences. The former shows typical examples from UAV object detection data sets, while the later shows examples sampled from COCO.

Another distinct challenge in UAV object detection is the gap in data set sizes. Generic object detection tasks have benefited immensely from large-scale datasets such as COCO (Common Objects in Context), ImageNet (Deng *et al.* (2009)), and Open Images Dataset (Kuznetsova *et al.* (2020)), to only name a few. They consist of roughly 200 thousand, 14 million, and 1.9 million labeled images, respectively. In contrast, UAV-specific data sets are significantly smaller. For example, UAVDT (Du *et al.* (2018)) and VisDrone (Du *et al.* (2019)) contain roughly $41,000$ and $7,000$ labeled UAV images, respectively.

Figure 1.2: Some example images from the data set common objects in context Lin *et al.* (2014), mostly containing humans. Clearly, the objects present in the images are a lot larger than for Figure 1.1, even for the humans with a comparably high distance to the camera.

These two issues amplify each other. When a lot of different scales are scattered across fewer images, the detector has fewer examples to learn from per domain. This hampers detection accuracy, as deep learning based detectors are well known for their data hunger. Additionally, the techniques developed to address the problem of differently scaled objects in generic computer vision, most notably Feature Pyramid Networks (FPNs) (Lin et. al. (2017)), unfortunately, fail for UAV object detection. This is primarily due to two reasons: First, FPNs introduce new weights to the detector network, each dedicated to specific object scales. Intuitively, when there is less data to learn from, these newly introduced weights will not perform as well. Second, the scales most frequently occurring in everyday objects, and therefore present in generic object detection data sets are hardcoded as biases in FPNs. Consequently, FPNs are inherently not designed for UAV imagery.

An additional obstacle for UAV object detection is given by the weight and energy consumption constraints on the payload. Instead of full-fledged Graphics Processing Units (GPUs), UAVs usually utilize small, embedded GPUs such as the NVIDIA Jetson Series (Orin (2020)). This constraint limits the size and capability of the neural networks that can be deployed, making it challenging to achieve high performance with computationally intensive state-of-the-art deep learning models.

Despite these challenges, UAV computer vision also has one unique advantage compared to generic computer vision: UAVs are equipped with additional sensors such as GPS, IMU, and barometers. These sensors provide contextual information about the drone's position, most valuable its flight altitude, and its orientation. This additional data can be leveraged to improve the accuracy and robustness of computer vision algorithms. For example, knowing the UAV's altitude and angle can help in estimating the scale of objects more accurately and adjusting the processing algorithms accordingly.

These distinct challenges and advantages give rise to an extensive discussion in Chapters 3, 4, and 5. Computer vision can determine whether there are humans in distress at the current location. However, to conduct a meaningful search, the next section will discuss the field of path planning and its importance to mSAR missions.

1.2 mSAR Path Planning

Path planning is a well-established problem in robotics and navigation. Traditional path planning algorithms, such as A* or Dijkstra (Russell (2010)), are designed to find the cheapest path between two points within a defined space according to some cost function. Importantly, these algorithms are designed for path finding settings where the start and goal are fixed positions, and obstacles are static. However, the trajectory planning problem in the context of maritime search and rescue missions diverges from these strong assumptions by being highly dynamic, requiring specialized planners.

In mSAR operations, the objective is to maximize the probability of detecting distressed individuals or vessels within a vast and non-static maritime environment. Here,

the location of the search targets (the distressed individuals or vessels) is unknown and can vary significantly over time due to confounders such as water currents and wind flow, the two main factors for the search targets' movement. This uncertainty introduces a layer of complexity that traditional path planning algorithms are not equipped to handle. Unlike common planners, there is a lack of a meaningful heuristics like, for example, the euclidean norm to the goal node most frequently used in A*. Additionally, the search area in mSAR missions is typically much larger and less structured compared to the ones in classical robotics or navigation. Mostly, it is open sea, which has virtually neither limits nor obstacles. These two factors cause the search graph to grow exponentially with the number of steps taken. Figure 1.3 showcases the search problem at hand.

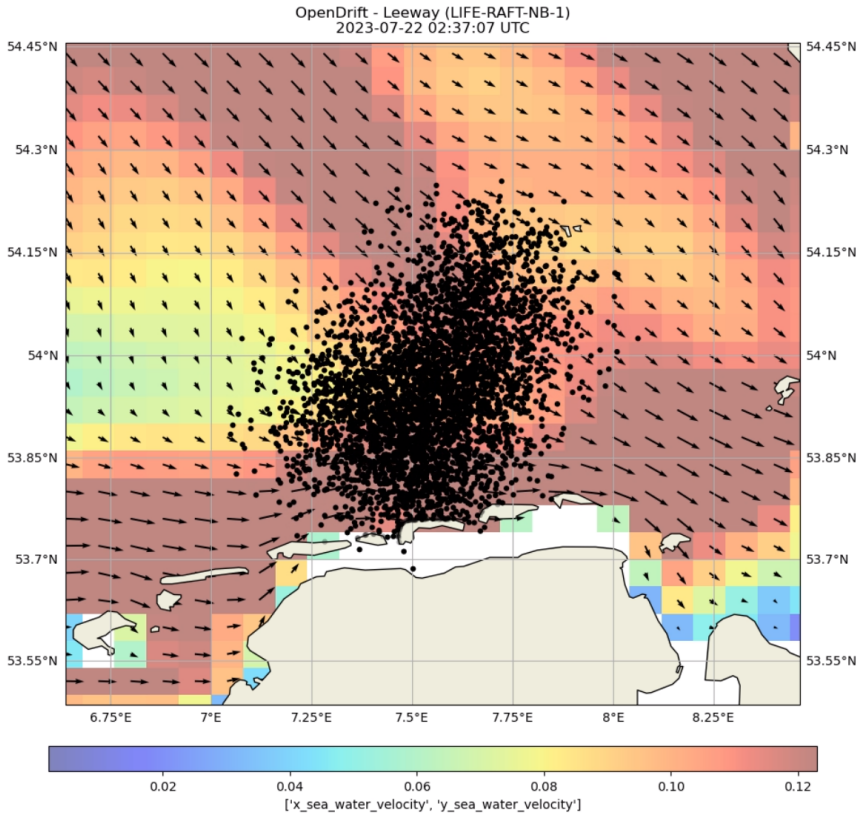


Figure 1.3: Example of a distribution modeling the estimated position of multiple search targets. It results from a drift simulation modeling the situation several hours after the distress signal. See Chapter 6 for further details.

To address the mentioned challenges, we employ the well-known branch-and-bound (B&B) algorithm to meet the specific needs of mSAR scenarios. We incorporate meteorological data, such as water currents and wind flow, to model the drift of search targets as realistically as possible, thereby improving the likelihood of detecting distressed individuals or vessels. We then compare the performance of our modified branch-and-bound

algorithms with environment-agnostic algorithms commonly used in practice, showcasing the need to employ environmental data into mSAR trajectory planning algorithms.

Our discussion of UAV trajectory planning for mSAR missions can be found in Chapter 6. The next section will summarize the contribution and outline of this dissertation and conclude this introductory chapter.

1.3 Contribution and Outline

This work offers a thorough overview of the use of UAVs in maritime search and rescue operations, primarily focusing on the search part. This is based on five first-author publications presented at highly ranked conferences in computer vision and robotics. These are listed according to their order of appearance in this dissertation, with an asterisk (*) indicating equal contributions:

1. Martin Messmer, Andreas Zell. "**UAV-Assisted Maritime Search and Rescue: A Holistic Approach**". In: 2024 International Conference on Unmanned Aircraft Systems (ICUAS), 2024.

2. Benjamin Kiefer*, Martin Messmer*, Andreas Zell. "**Diminishing Domain Bias by Leveraging Domain Labels in Object Detection on UAVs**". In: Proceedings of the IEEE 20th International Conference on Advanced Robotics (ICAR), 2021.

3. Martin Messmer*, Benjamin Kiefer*, Andreas Zell. "**Gaining Scale Invariance in UAV Bird's Eye View Object Detection by Adaptive Resizing**". In: Proceedings of the IEEE 26th International Conference on Pattern Recognition (ICPR), 2022.

4. Leon Amadeus Varga*, Benjamin Kiefer*, Martin Messmer*, Andreas Zell. "**SeaDronesSee: A Maritime Benchmark for Detecting Humans in Open Water**". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022.

5. Martin Messmer, Benjamin Kiefer, Leon Amadeus Varga, Andreas Zell. "**Evaluating UAV Path Planning Algorithms for Realistic Maritime Search and Rescue Missions**". In: 2024 International Conference on Unmanned Aircraft Systems (ICUAS), 2024.

Additionally, these 3 papers will not be covered in this dissertation, since they do not fit the scope of this work:

6. Leon Amadeus Varga, Martin Messmer, Nuri Benbarka, Andreas Zell. "**Wavelength-Aware 2D Convolutions for Hyperspectral Imaging**". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023.

7. Timon Höfer, Benjamin Kiefer, Martin Messmer, Andreas Zell. "**HyperPosePDF – Hypernetworks Predicting the Probability Distribution on** $\mathrm{SO}(3)$". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023.

8. Benjamin Kiefer, Martin Messmer, et al. "**2nd Workshop on Maritime Computer Vision (MaCVi) 2024: Challenge Results**". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024.

After this introduction, the work is structured as follows:

Chapter 2 The second chapter discusses the integration of UAVs into maritime search and rescue operations and the usefulness of various types of drones for different applications. Additionally, we introduce a specialized software framework designed to aid the detection pipeline of maritime search and rescue operations consisting of an onboard-part running on the UAV and a powerful GPU-server based on the ground as its counter piece. This framework runs a region of interest proposer onboard the drone and streams its detections along with some metadata to the ground station. Due to the disparity in research efforts in between conventional object detection and region of interest proposal systems, we also explore various methods for predicting regions of interest within maritime images in this chapter. Notably, we test these methods under simulated bandwidth constraints of the radio link, which is a realistic scenario given the remoteness of maritime environments. Chapter 2 is based on Messmer *et al.* (2024).

Chapter 3 This chapter focuses on the impact of domain variations in the UAV imagery such as altitude and camera angle and especially on their distribution across various well known UAV data sets. We employ these observations, introducing a novel approach to improve detection performance by incorporating domain knowledge into the detection models, leveraging sensor data available from the UAVs like altitude, camera angle, and time. The chapter also introduces a new data set, PeopleOnGrass (POG), specifically created to provide detailed annotations and domain labels for further research. Experiments demonstrate significant improvements in detection performance across various data sets and metrics, suggesting that domain-aware models can effectively reduce bias introduced by domain imbalances. This chapter is based on our publication Kiefer *et al.* (2021), number two from the enumeration.

Chapter 4 In this chapter we introduce a novel method called *Adaptive Resizer* designed for bird's eye view object detection. As this method is a preprocessing step, it is essentially applicable to any object detector. By resizing images based on their capture altitude, which is obtained from the UAV's onboard sensors, it makes the detection results invariant under scale variation, a problem inherent to UAV imagery. By reducing images' sizes it furthermore reduces the computational burden imposed by

the object detector. Additionally, the method allows for effective transfer learning, capable of generalizing across different capture-altitudes not present during training. Finally, building on Chapter 3, we present a detector working on general UAV images, expanding beyond the bird's eye view imagery. The basis for this chapter is Messmer *et al.* (2022), or the third item in above's enumeration.

Chapter 5 Here, we introduce *SeaDronesSee*, a new benchmark data set aimed at improving UAV-based object detection in maritime settings. The data set consists of over $54,000$ frames annotated with roughly $400,000$ instances of humans in the water and boats captured from different UAVs and cameras across a variety of altitudes and angles. The data set is fully public. By providing detailed meta-data for each frame, we enable the development and testing of meta-data aware computer vision techniques. We provide details about the data set's creation, including image capture conditions and annotation process, as well as the initial evaluations of state-of-the-art models on this data to establish baselines and their performance on the new data set. Chapter 5 is based on our published work Varga *et al.* (2022), the fourth item in the enumeration above.

Chapter 6 The final chapter discusses the development and evaluation of path planning algorithms for UAVs in maritime search and rescue missions. We propose a path planning algorithm based on branch-and-bound techniques, designed to take into account real-world factors like water current and wind flow, which influence the distressed search targets' trajectories over the course of the search. It aims at bridging the gap between existing methods that already account for environmental influences but are computationally infeasible, and those that do not consider such factors. Moreover, it provides an extensive comparison of various path planning strategies, highlighting their strengths and limitations under simulated conditions reflective of real-world scenarios. The framework developed for this comparison is publicly available. This chapter is based on Messmer and Zell (2024), the last item in the previously mentioned enumeration.

In the following, the technical set up of drones in maritime search and rescue missions will be discussed.

Chapter 2

A Holistic Approach to UAV-Assisted Maritime Search and Rescue

The vast and unpredictable nature of maritime environments presents a significant challenge for search and rescue (SAR) operations. Traditional methods, often reliant on manned vessels and aircraft, face limitations in speed, accessibility, and risk to human life (CCG Manual (2023)). With the advent of Unmanned Aerial Vehicles (UAVs), new possibilities have emerged, offering enhanced efficiency, safety, and endurance in maritime SAR (mSAR) missions. This chapter delves into the application of UAVs in mSAR, specifically medium-sized fixed-wing drones and quadcopters, focusing predominantly on their utility in search operations due to their physical constraints.

In particular, we discuss various classes of drones, including quadcopters and fixed-wing drones of different sizes, along with their respective advantages and disadvantages. We delve into how each type of drone could be effectively utilized at different stages of a mSAR mission, and address the potential challenges and limitations when operating them in such scenarios.

One key aspect of our research is the development of a comprehensive software framework that enables the prediction of preliminary detection onboard the UAVs, followed by a more capable but more compute intensive object detector on the interesting regions of the image on a more powerful ground station (Carion *et al.* (2020); Girshick (2015)).

The initial detections, called Regions of Interest (RoI) (Kiefer and Zell (2023)), are intended as the primary focus for both the detection system and the operator at the ground station as they are the most likely to contain humans or vessels in distress. Additionally, the software is designed to stream these regions of interest in their full quality, in contrast to the rest of the image, which is transmitted in lower quality and gray-scale (see figure 2.6). This approach accounts for potential constraints of limited bandwidth, which can be a significant factor when operating UAVs in remote locations or far off-shore. Therefore, the proposals for regions of interest serve multiple purposes and are a vital component of the overall pipeline. Hence we present an evaluation of various methods for generating these proposals in Section 2.4 of this work. In particular, we put emphasis on the bandwidth constraint in these experiments, evaluating the performance of each

© 2024 IEEE. Reprinted, with permission, from Messmer *et al.* (2024).

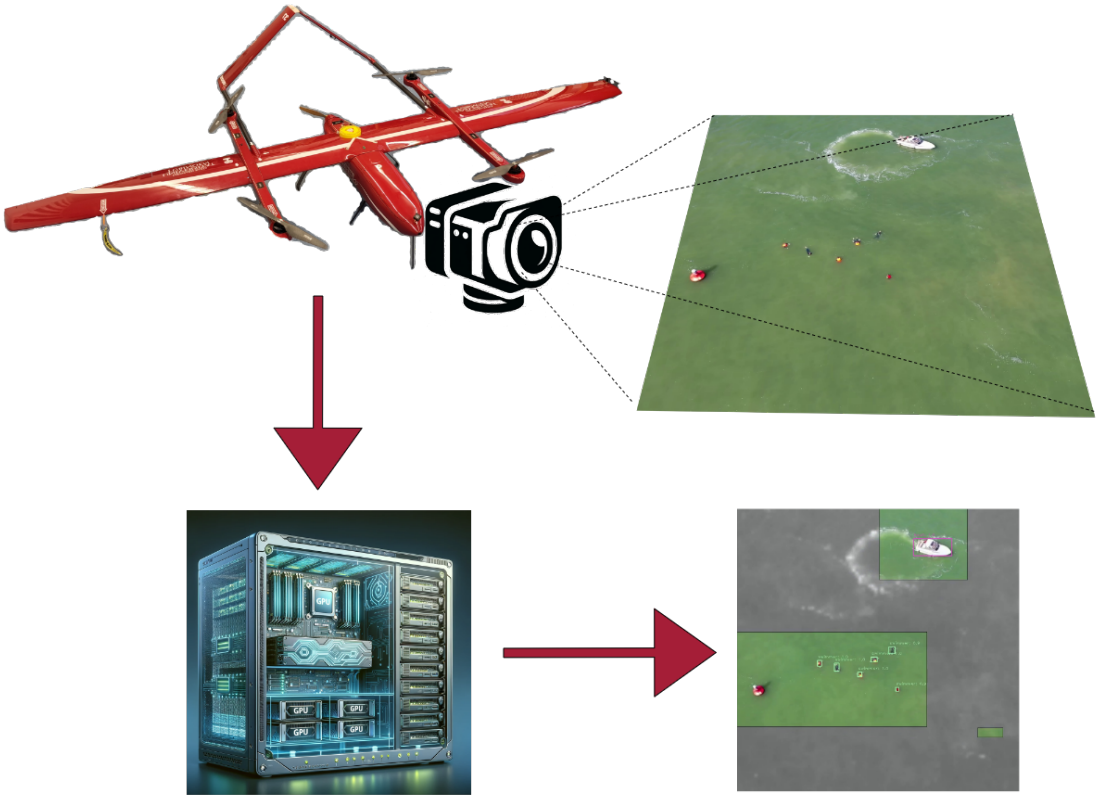


Figure 2.1: The maritime surface is captured by the drone and then, together with the preliminary detections (RoIs) transmitted to a GPU server on the ground. This then performs fine-grained detection on the RoIs and displays the whole scene to the SAR operator.

algorithm under various transmission restrictions.

By advancing the integration of UAVs in mSAR missions, our research aims to significantly enhance the efficiency and effectiveness of maritime search operations, ultimately contributing to faster response times and increased chances of successful rescue in maritime emergencies. Moreover, this software design aims to reduce the visual strain on the drone operator on the ground by automating a portion of the decision-making process. This aids decision making, as watching a screen extensively might lead to 'eye fatigue' (Coles-Brennan *et al.* (2019); Council (2016); Sullivan (2008)), which might potentially lead to bad decisions. Our hope is that this automation will long term lead to operators making more accurate decisions by minimizing the chances of overlooking important details.

In Lygouras *et al.* (2019) the authors propose a full system for UAV-based mSAR missions. In contrast to the paper at hand they perform regular object detection fully onboard the drone on a NVIDIA Jetson TX1. The work Stabernack and Steinert (2021)

concerns itself with the stream of a low-quality video with embedded high-quality regions of interests on FPGAs. In Mayer *et al.* (2019) the authors discuss the advantages of using drones in SAR missions in general which lays the foundations for works like ours. An interesting RoI proposal method is given in Kiefer and Zell (2023). However, it requires working with video streams, which is an adequate requirement on RGB data. Our approach, on the other hand, may also be applied to multispectral data, which can usually only be captured at 1 Hertz, see e.g. Chapter 5. In Roberts *et al.* (2016) the authors present a framework to analyze mSAR missions with UAVs. Since we were not able to find their framework we can only estimate from their description, that its goal is not to conduct the actual search operation but to judge whether or not the use of UAVs make sense for the case at hand. This goal differs largely from the framework developed (and published) in this chapter.

The remainder of the chapter is structured as follows: Section 2.1 provides an overview of different drone classes, discussing their advantages and disadvantages. Section 2.2 discusses the software framework we propose, including both, UAV and ground station components, while Section 2.3 discusses the RoI proposal methods and Section 2.4 details our experimental setup and the results and observations obtained. Finally, Section 2.5 concludes with a discussion of our findings and their implications.

2.1 Choosing the Right Drone for mSAR Missions

Table 2.1 shows the technical details of the DJI Matrice M210, the Quantum Systems Trinity F90+, and the ElevonX SkyEye Sierra VTOL. All of these were used in various capacities in our experiments regarding SAR research. Their advantages and disadvantages are meant to be exemplarily for their respective classes of drones.
The DJI Matrice 210 (figure 2.2), as a quadcopter, offers excellent maneuverability and the ability to hover in place, which is crucial for precise, targeted searches and for capturing specific areas of interest. However, its flight time and speed are less than the other two, which may limit its range and efficiency in covering large maritime areas.

The Quantum Systems Trinity F90+ (figure 2.3) is a fixed-wing VTOL (vertical takeoff and landing) that provides a longer flight time of roughly 90 minutes and a large coverage area, making it suitable for initial wide-area searches. Its ability to carry various sensors is advantageous for generating a diverse dataset necessary for training and refining deep learning algorithms. The fixed-wing design offers higher speed and greater efficiency over long distances compared to quadcopters, but it lacks the ability to hover, which can be a limitation for close inspections. Additionally, it significantly complicates the operation of the UAV compared to quadcopters, placing a considerable demand on the operator to maintain a flawless overview of the situation at all times and to think ahead. Furthermore, the maneuverability is limited compared to quadcopters, resulting in an inability to be operated in confined spaces.

One of the key characteristics of the Trinity F90+ is its limited adjustability, especially

	DJI Matrice 210	Quantum Systems Trinity F90+	ElevonX Skyeye Sierra VTOL
System	Quadcopter	VTOL Fixed Wing	VTOL Fixed Wing
Weight	5 *kg*	5 *kg*	12.5 *kg*
Max. Payload Weight	1.3 *kg*	✓	3 *kg*
Air Speed (Range)	$0-12$ m/s	$17-21$ m/s	$17-21$ m/s
Min. Flight Altitude	0 *m*	30 *m*	50 *m*
Max. Flight Time	30 *min.*	90 *min.*	3 *h* (electric) 5 *h* (gas)
Command & Control Range	8 *km*	7.5 *km*	20 *km*
Wingspan	89 *cm*	239 *cm*	300 *cm*
Number of Operators	1	1	2

Table 2.1: Technical data of the UAVs used in this research (DJI (2018); Quantum (2020b); ElevonX (2023)). As described in the text, possible payloads for the Trinity F90+ are limited to those offered by the manufacturer. Therefore, weight restrictions for the payload are not a concern, as indicated by the ✓.

regarding payload integration. Users can choose from a range of cameras supported by the manufacturer, including multiple different RGB and Multispectral cameras, a LiDAR scanner, and an Oblique RGB camera (Quantum (2020a)). The latter has five lenses, each oriented slightly differently, designed for 3D mapping. This selection caters to various data collection needs, from detailed imagery to topographical mapping. However, the inability to integrate custom payloads or sensors outside the manufacturer's offerings restricts the drone's versatility. In particular the inability to employ onboard processing in the form of an NVIDIA Orin (Orin (2020)) or similar.

Furthermore, the Trinity F90+ operates with a proprietary communication system that lacks openness or user adjustability. The absence of a customizable down-link connection means that users are confined to the data transmission and control options provided by the manufacturer. This could pose challenges in integrating the drone into a broader system that employs deep learning algorithms. More precisely, it is impossible to stream captured data instantaneously. This rules out online post-processing on a more powerful GPU server on the ground.

In summary, the Trinity F90+ can play an important role in AI-aided mSAR missions by gathering extensive amounts of high-quality aerial data necessary to train deep learning algorithms used for detection.

The ElevonX Sierra SkyEye VTOL (Fig. 2.4) combines some of the benefits of both,

Figure 2.2: DJI Matrice 210

quadcopters and fixed-wing aircraft. It has a significant endurance of up to 3 hours with electric propulsion, which is essential for extended missions. Similar to the Trinity F90+, it is a VTOL UAV and therefore comes with the same advantages and disadvantages compared to a quadcopter. The cruise speed and range are well-suited for both detailed search operations and extensive area coverage.

The adjustability of the ElevonX drone is particularly beneficial for maritime search and rescue. While we only used the the electric propulsion in our experiments, the availability of gas propulsion is a nice addition for operational deployment, as it offers the longest endurance for any of the tested drones. However, the vibrations a combustion engine produces may hamper image quality.

One drawback of the Sierra VTOL is its reduced user-friendliness, as it demands continual practice from operators due to its complex nature. Additionally, the system requires two operators to function effectively, which ties up limited resources.

The payload versatility and capacity of 3 kg mean that the drone can be equipped with various sensors, such as optical and thermal cameras, to capture a wide range of data during both day and night operations. In particular, we were able to install an NVIDIA Jetson Orin AGX 64 GB on the drone and connect it to our RGB camera (see Figure

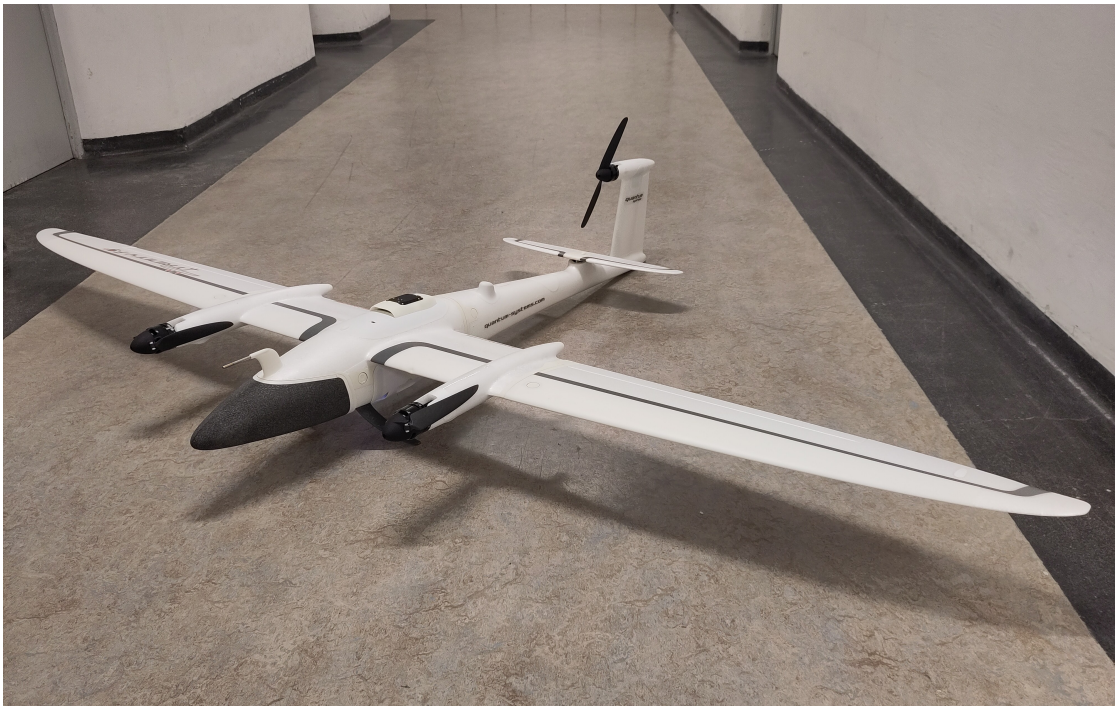


Figure 2.3: Quantum Systems Trininty F90+

2.5) which enabled us to perform onboard processing for Region of Interest proposals, as discussed in Section 2.3. The Sierra VTOL allows for up to 120 Watts of power within the drone.

In summary, the DJI M210 is ideal for targeted search operations and detailed inspections, the Trinity F90+ excels in extensive area mapping and data generation, and the ElevonX Sierra SkyEye VTOL offers a blend of endurance, adjustability, and payload capacity while being the most complex to operate. The choice of drone would depend on the specific phase of the mission and the mission itself. The critical considerations in this context are: is onboard processing desired? Is there a need to capture data extensively? Is there a focus on continuously monitoring small, specific areas, or is the objective to search larger spaces?

2.2 Software Solution for SAR Drones

Our proposed software solution is divided into two components: the software operating on the embedded GPU onboard the UAV, and the program running on the ground station. In the following, we provide a concise description of the functionalities. The software, containing both parts, is available on GitHub[1]. This design aims to address the challenges

[1] `https://github.com/cogsys-tuebingen/UAV-based-maritimeSAR`

Figure 2.4: ElevonX SkyEye Sierra VTOL.

typical in maritime mission scenarios, such as the instability of stable data links from the drone to the ground station. In missions where drones operate multiple kilometers off shore, stable data links like LTE coverage are generally unavailable. Consequently, to minimize the volume of transmitted data, only selected parts of the image are streamed in full quality, as discussed later in this section.

Onboard UAV Software

The software onboard the UAV assumes that the flight controller and path planning system are taken care of. For example, with the UAVs discussed in section 2.1 these functionalities are provided by the manufacturer. This software has, in essence, three main tasks:

1) Running the camera: The software fetches the video stream from the connected camera and then preprocesses this video data for both detection and streaming.

2) Region of Interest (RoI) proposal: Detect regions of interest for closer examination by the ground station software. The identified regions will be transmitted to the ground station in full detail. However, this process might be constrained by the available

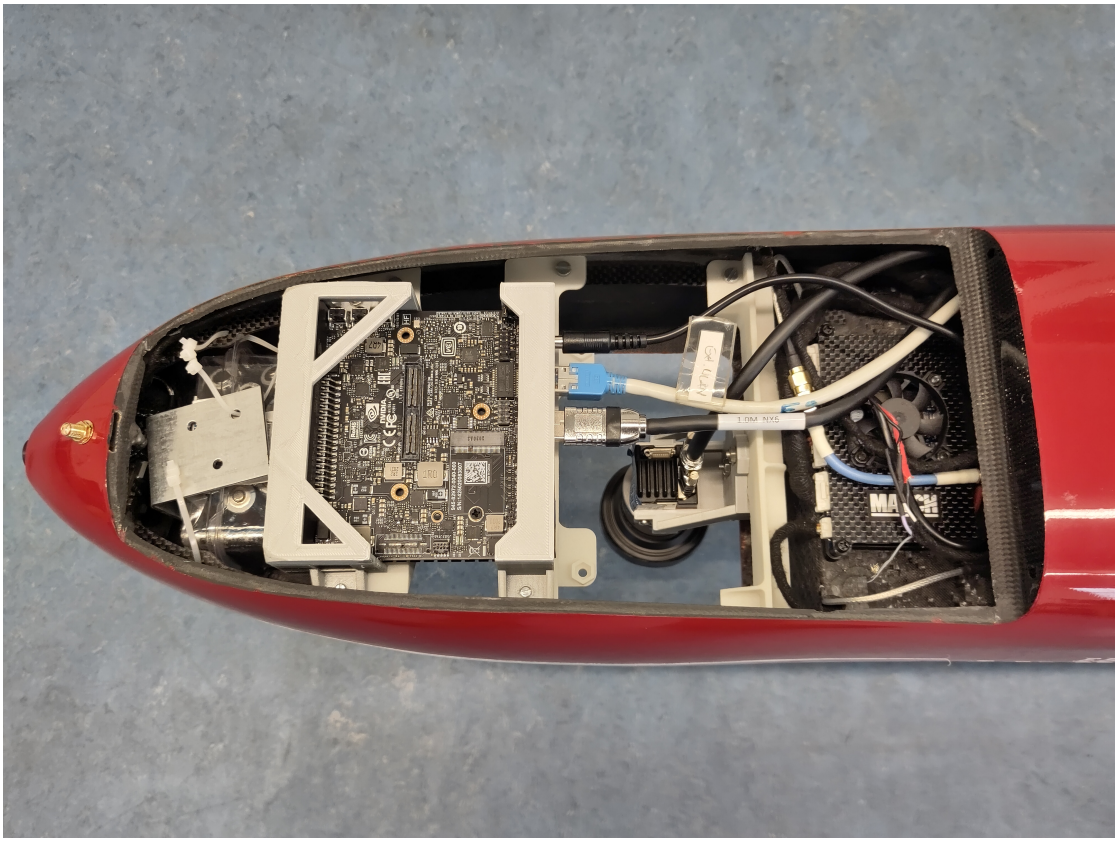


Figure 2.5: NVIDIA Jetson Orin AGX 64 GB built into the Sierra VTOL with 3D-printed mount.

bandwidth.

3) Data Transmission: Establishing a connection with the ground station software is crucial. The onboard software streams the down-scaled full image, the RoIs in full quality, and metadata about these RoIs. This metadata includes the time stamp to align the RoIs with the video stream. Note: The actual streaming mechanism is a topic for potential further research work but falls outside the scope of this dissertation. Some possible solutions involve streaming FPGAs implementing a dedicated codec for high-quality RoIs embedded in a low-quality video stream (Stabernack and Steinert (2021)).

In our basic setup, we implemented three distinct streams: an RTSP stream for the down-scaled[2], grey-scale image, the full-quality, full-color RoIs, and the accompanying meta-data, like position of the RoIs in the image. Each of these streams includes a timestamp, enabling their synchronization at the ground station.

[2]In our implementation, we reduced the scale by a factor of 8 in both the x and y directions, resulting in streaming every 64th pixel in total. Naturally, this scaling factor can be adjusted to fit the available bandwidth.

Ground Station Software

The receiving part of the software assumes the ground station to be a GPU server equipped with powerful graphics cards to run demanding object detection models. This way, the software can run these detectors on the received RoIs. Essentialy this software needs to perform four tasks:

1) Data Streaming: This software needs to establish and maintain a connection with the UAV software, receiving the various data streams sent by it.

2) Detailed Detection: It runs powerful object detectors on the RoIs proposed and streamed by the drone software. For this, the software in the ground station needs to manage the workload imposed by multiple RoIs and over multiple video frames across all available GPUs.

3) Operator Interface: The software presents the information in a GUI to an operator at the ground station, enabling them to take action, if required, such as identifying individuals in distress.

4) Custom RoI Requests: Additionally, the operator has the ability to request custom RoIs if they believe the drone software may have missed something. These custom RoIs are then transmitted back to the drone software and then are also transmitted in full quality to the ground station.

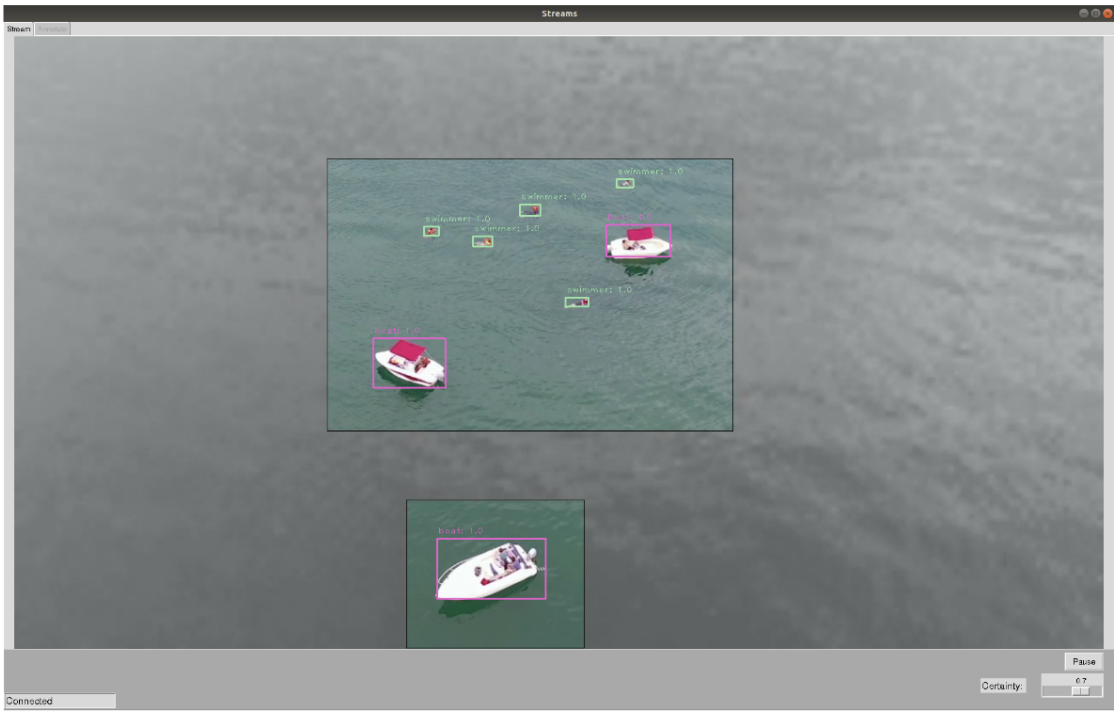


Figure 2.6: Example footage from the GUI of the ground station software.

2.3 Region of Interest Proposer Methods

The first method in this chapter's evaluation for Region of Interesent (RoI) proposals is hugely based on Grad-CAM++ (Chattopadhay *et al.* (2018)). Its goal it to enhance the understanding of how deep learning models, particularly CNNs, process and interpret visual data. The authors do so by generating visual explanations, so called *saliency maps*, for the predictions made by Convolutional Neural Networks. Figure 2.7 shows an example of a saliency map.

In essence, this technique creates a heat map on the image, where pixels with higher values indicate areas that the neural network relies on more for its decision-making process. This is done by calculating higher order derivatives of the network's classification output in relation to specific pixels. Intuitively, this approach makes sense; examining how altering a particular pixel influences the network's class output of a specific bounding box gives insight into the significance of that pixel in the network's decision-making process.

While originally developed for shedding light on the decision-making process of convolutional neural networks, we adapt this method in our study for class-agnostic object detection. We achieve this by processing the output heatmap and rounding each value to either 0 or 1, resulting in a binary map. Next, OpenCV's (Bradski (2000)) `findContours` (Suzuki *et al.* (1985)) function, traces the outlines of the shapes in the image that correspond to the regions marked by the value 1 in the binary map. Since the `findContours` function is furthermore able to fill the interior of this outlined shape, it can recover bounding boxes by simply fitting the smallest possible axis-aligned rectangular box around each connected component of the areas marked as 1 in this binary heat map. For this approach we merely employ a generic purpose backbone frequently used for computer vision tasks, ResNet-18 (He *et al.* (2016)).

The primary benefit of this method lies in its class-agnostic nature. Given that the model isn't specifically trained on maritime data, applying it to out-of-distribution scenarios poses no issue. This flexibility allows for broader applicability across various data, even if it differs significantly from the model's training data. In the following we call this approach *saliency detection*. Figure 2.7 shows a graphical example of this algorithm.

The second method we employ for RoI proposals is YOLOv7 (Wang *et al.* (2023)). Since this approach is a vanilla generic object detector, it needs training on data that is representative of the same distribution as the test data. This is an obvious drawback in comparison to the saliency detection method.

However, this method ranks among the most efficient and accurate object detectors currently available to the research community. This model consistently ranks as one of the top-performing models in relevant benchmarks for real-time object detection (Papers with Code (2024a)), such as the MS COCO dataset (Lin *et al.* (2014)). Another significant advantage of YOLOv7 is its user-friendliness. Its implementation is easy to use and operational immediately, requiring only a single line of command for retraining or fine tuning. This feature is particularly beneficial for our application, as the model will

be handed over to users who are not AI-experts. They still might need to fine tune the model on newly gathered data.

Since the task differs slightly from standard object detection, we experimented with two different configurations of YOLOv7. In the first (denoted by SBB later), we use it as a conventional object detector, trained on SeaDronesSee in the usual manner[3]. For the second configuration (denoted by LBB later), we modified the dataset as follows: every bounding box in the dataset is expanded to a minimum size of 500×500 pixels (while images are 3840×2160). This adjustment is based on the assumption that the detection of an individual or object might indicate the presence of additional items in its vicinity, potentially leading to their detection as well. This modification is a significant alteration to bounding boxes for many objects considering that a vast number of them are merely around 20×20 pixels in size.

This strategy, in theory, might lead to a decrease in detection performance in scenarios with limited streaming bandwidth, but proves advantageous when the available bandwidth is sufficiently high, see Figure 2.8.

2.4 Experiments

As object detection is studied extensively already (Carion *et al.* (2020); Girshick (2015); Wang *et al.* (2023)) and theory and experiments on the path planning of the drone are out of the scope of this work, we confine ourselves to exploring various proposal methods for regions of interest predicted by the embedded GPU on the drone. This requires algorithms have a high recall and be fast, to ensure that no objects present in the scene are overlooked.

Data Set

We conducted evaluations of our algorithms using the test set from the second iteration of the SeaDronesSee dataset (see Chapter 5). This test set consists of 4235 images, while the training and validation sets contain 8125 and 1431 images, respectively. It is composed of images featuring from 1 to 15 swimmers per image, as well as small vessels in open water. These images were captured using multiple different drones, including the DJI M210 and Trinity F90+ models discussed in this work. The dataset also was captured by various cameras and diverse weather and lighting conditions, as it was gathered over multiple days.

[3] `https://github.com/WongKinYiu/yolov7/blob/main/data/hyp.scratch.p5.yaml`

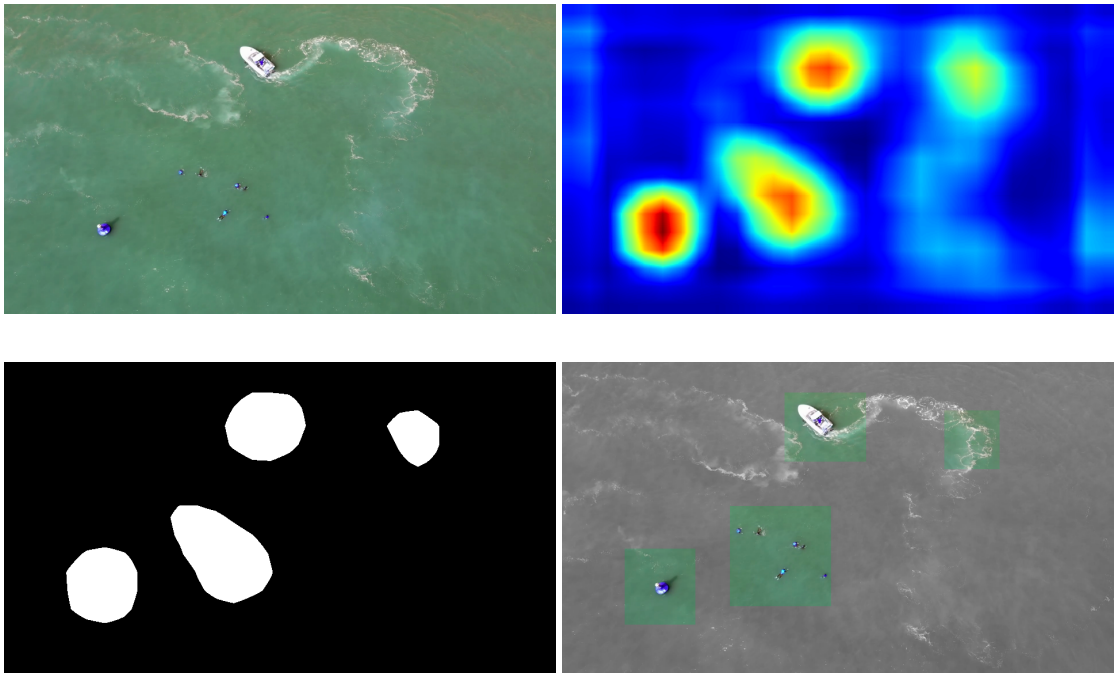


Figure 2.7: Example of a heat map produced by our saliency detection method. The image on the top left is from the test set of the SeaDronesSee data set, as described in Chapter 5. The image on the top right shows the corresponding heat map. The bottom left shows the resulting binary map after rounding each pixel to 0. and 1.. The bottom right shows the resulting RoIs on the images, the background is in gray scale to highlight the predictions. We can see, that all relevant instances in the image are detected. From left to right, the bounding boxes detect a buoy, swimmers, a boat, and waves – the only irrelevant detection.

Evaluation Metrics

To accommodate the unique requirements of the application at hand, we employ specialized evaluation metrics. This involves a slightly modified version of the conventional metrics precision, recall, and the F1 score. These metrics are widely used in the field of computer vision and machine learning to assess the accuracy of predictive models. The necessary modification lies in how a 'True Positive' is defined in this context.

Typically, in the process of matching predicted and ground truth bounding boxes, the *Intersection over Union* (IoU) is calculated for each pair of predictions and ground truth boxes $(p_j, g_k) \in P \times GT$, where P is the set of predicted boxes and GT the set of ground truth boxes. A match is established when a pair yields an IoU greater than 0.5 and also

represents the highest IoU among all possible pairings, formally:

$$\text{IoU}(p_j, g_k) \geq \frac{1}{2} \tag{2.1}$$

$$\text{IoU}(p_j, g_k) = \max_{m,n} \text{IoU}(p_m, g_n) \tag{2.2}$$

Here, the maximum in line 2.2 only is taken over all predictions p_m and ground truth boxes g_n that are not yet matched. This matching process is one-to-one, linking each ground truth box to at most one predicted box. Each prediction matched to a ground truth box is called a *True Positive*.

This approach is well-suited for precise object detection, where each prediction is expected to correspond to exactly one object within the image. However, for Region of Interest (RoI) prediction, where the goal is to enclose all objects present in the given scene, the matching method needs to be adjusted.

Specifically, this implies that the strict one-to-one correspondence between predictions and ground truth boxes are not necessary, nor is the precise size alignment of the predicted bounding boxes with the actual objects.

Therefore, instead of IoU we use *Intersection over Ground Truth* (IoGT), defined as

$$\text{IoGT}(p, g) := \frac{\text{area}(p \cap g)}{\text{area}(g)} \in [0, 1].$$

This term focuses only on the area of the actual object, rather than looking at the area covered by both the predicted and actual object together. This way, even if a predicted area is much larger than needed, it can still be considered a match. In the extreme case of a predicted RoI covering the whole image, every ground truth bounding box would be counted as a true positive. With the traditional Intersection over Union method, a very large prediction would result in a low value, possibly missing the match. More formally, similarly to above equations, our matching procedure is:

$$\text{IoGT}(p_j, g_k) \geq \frac{1}{2} \tag{2.3}$$

$$\text{IoGT}(p_j, g_k) = \max_{m,n} \text{IoGT}(p_m, g_n) \tag{2.4}$$

Here, the maximum in equation 2.4 is taken over all p_m and g_n, regardless of whether they were matched to some other box already or not. Each ground truth box that matches with a predicted box is now considered a true positive.

Using this altered metric, we strive to achieve a shift in focus from accuracy on individual objects to predicted areas covering all present objects in the scene, even if it's not a perfect match for each one. This is the goal of region of interest prediction. We believe, our metric achieves precisely this goal.

Results

To assess the performance of the discussed region proposal methods, we compute their adapted precision, recall, and F1-score with the discussed modifications. Additionally, we simulate a scenario with reduced bandwidth for the data link from the drone to the ground station. In our evaluation, this is achieved by limiting the predictions generated by the algorithm to the quantity that can feasibly be transmitted given the bandwidth constraints. We do so as follows: given a portion $r \in [0,1]$, which denotes the maximum portion that can be streamed in full quality, and predictions $P = \{p_1, \ldots, p_n\}$. Then, in the case of saliency detection, where we do not have information about the quality of the boxes, we chose a subset $\{p_{k_1}, \ldots, p_{k_m}\} \subset P$ such that area $\left(p_{k_1} \cup \ldots \cup p_{k_m}\right)$ is maximal and not greater than r. For YOLOv7, where each bounding box comes with a confidence score, we chose the boxes with the highest score until the area exceeds r. In either case, once the bandwidth limitation is reached, we select the next bounding box in this order. For saliency detection, we opt for the largest remaining box, assuming that it represents the highest quality detection. We fit the largest axis-aligned box (with equal center) into the selected box, while still accounting for the streaming restriction, and append this box to the subset. Figures 2.8, 2.9, and 2.10 show the results of the performance under these conditions. There, the maximum streamable percentage of the image is given by the x-axis while the result of the algorithm in the respective metric is plotted on the y-axis.

This approach allows us to understand how the methods perform under restricted data transmission conditions.

Figure 2.8 shows the recall of the compared algorithms, arguably the most important metric for RoI proposal methods in mSAR missions. We observe that vanilla YOLOv7 outperformed the other models in almost all scenarios with bandwidth limitations, making it the preferred choice under heavy bandwidth limitations. However, as the bandwidth restriction vanishes, meaning $r \to 1$, the YOLO LBB variant (trained on the modified data set with larger bounding boxes) begins to excel, surpassing the vanilla version. Similarly but even more so, the saliency detection method surpasses both YOLOv7 variants once the data link allows for around 60 % or more of the image to be streamed. This is not surprising, since the YOLO methods are trained to predict few false positives.

In addition, saliency detection has a unique benefit: it maintains consistent performance even when applied to data it wasn't trained on, unlike its counterparts, which were explicitly trained on SeaDronesSee. An interesting observation can be made regarding the YOLO model with enlarged bounding boxes: the noticeable jumps upwards in the graph occur when the bandwidth allows for one additional bounding box, leading to more additional detections quickly. These jumps are also visible in figures 2.9 and 2.10. In graph 2.9 they look like these additional bounding boxes harmed performance but figure 2.10 shows, that they are a net-gain for overall detection performance. The superiority of saliency detection for a high bandwidth can also be seen in the precision and F1-score plots, figures 2.9 and 2.10. Here, when roughly 30 % of the full image can be streamed, the figures show that saliency detection overtakes the other two algorithms

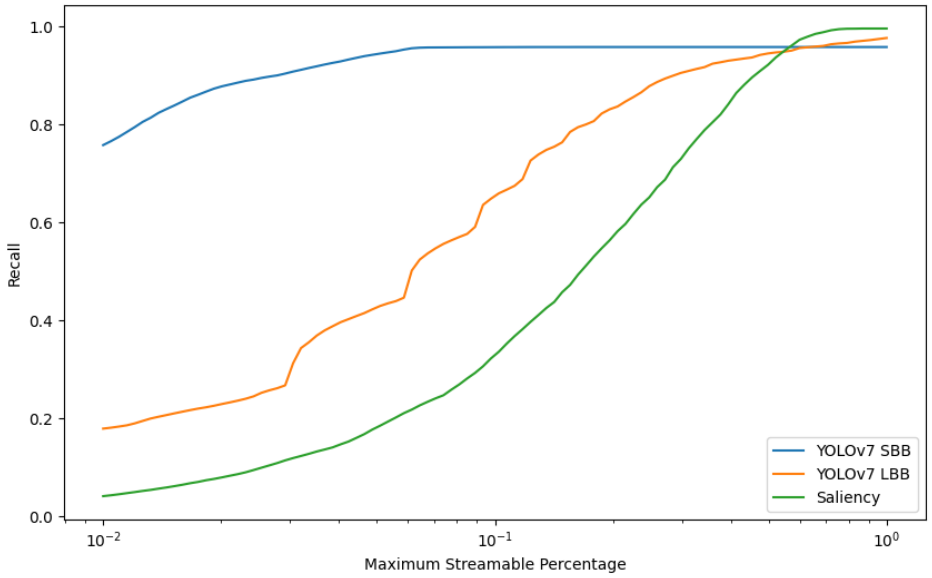


Figure 2.8: Logarithmic plot of the recall on the SeaDronesSee (see Chapter 5) test set for saliency detection and two different versions of YOLOv7 (Wang *et al.* (2023)). The portion of the maximum streamable resolution in relation to the full resolution is plotted on the x-axis. The different YOLO variants, YOLOv7 SBB and YOLOv7 LBB are introduced in section 2.3.

in performance. Although a high recall is the most important aspect for RoI proposal methods in mSAR missions, a high precision might as well be beneficial, as it reduces the burden on the object detector or the human operator present at the ground station.

Speed Benchmarks

In order to evaluate the possibility of onboard processing for the investigated algorithms, we conducted speed benchmarks on an NVIDIA Jetson Orin AGX 64 GB (Orin (2020)). We averaged running times for both algorithms on the test set of the SeaDronesSee benchmark data set. We used the TensorRT framework (Vanholder (2016)) to optimize the ResNet and the subsequent heat map generation in the saliency detection algorithm as discussed above. This part achieved an average processing speed of 51.0 frames per second (FPS) on the Orin. The post processing generating bounding boxes, employing OpenCV (Bradski (2000)), averaged 30.1 FPS. Running them in sequence hence results in roughly 18.9 FPS, falling short of running in real time. We conclude that, since the tasks are naturally separate, splitting the two tasks on two different processing units, one dedicated GPU for the neural forward pass and heat map generation and one dedi-

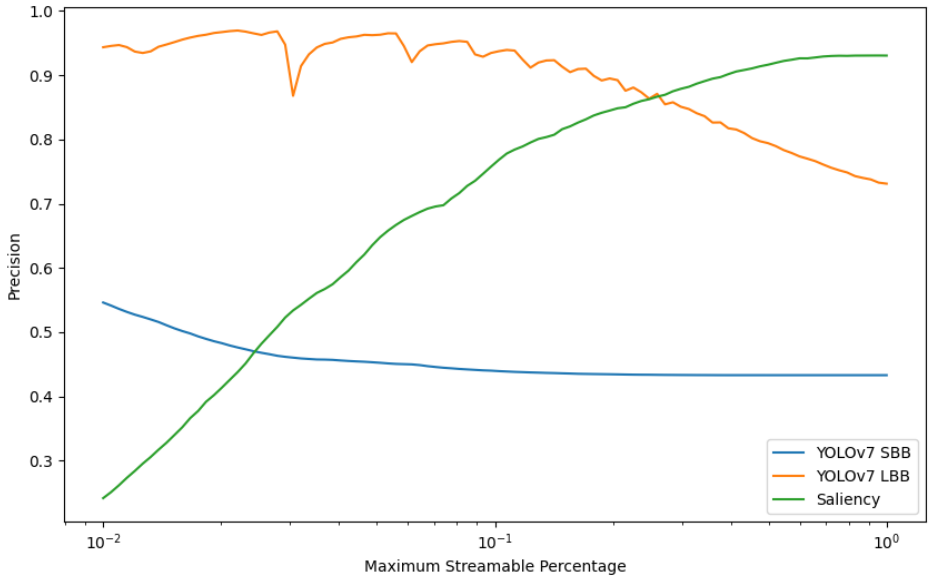


Figure 2.9: Logarithmic plot of the precision on the SeaDronesSee test set for saliency detection and the discussed versions of YOLOv7. The portion of the maximum streamable portion of the full resolution is plotted on the x-axis. The different YOLO variants, YOLOv7 SBB and YOLOv7 LBB are introduced in section 2.3.

cated CPU for the OpenCV computations, would achieve real-time by being as fast as the slower of the two operations, resulting in 30.1 FPS. On the other hand, YOLOv7 achieved an average of 45.5 FPS on the NVIDIA Jetson Orin AGX 64 GB while running natively in PyTorch (Paszke *et al.* (2019)). Although there is potential for further acceleration through TensorRT optimization, we deemed it unnecessary since it already achieved real-time performance. Since both researched YOLOv7 methods only differ in the training of the network, inference speed is equal for both.

2.5 Conclusion

In this chapter, we highlighted some of the features and caveats about using small fixed-wing drones and quadcopters in maritime search and rescue (mSAR) operations. Furthermore, we developed and implemented a specialized software framework specifically tailored for mSAR missions. This framework allows for models predicting regions of interest (RoIs) onboard the UAVs and also using more powerful object detectors at a ground station, thereby reducing the cognitive and visual load on operators. Additionally, we evaluated various RoI proposal methods specifically focusing on different bandwidth constraints, thereby highlighting their practicality in real-world mSAR scenarios involv-

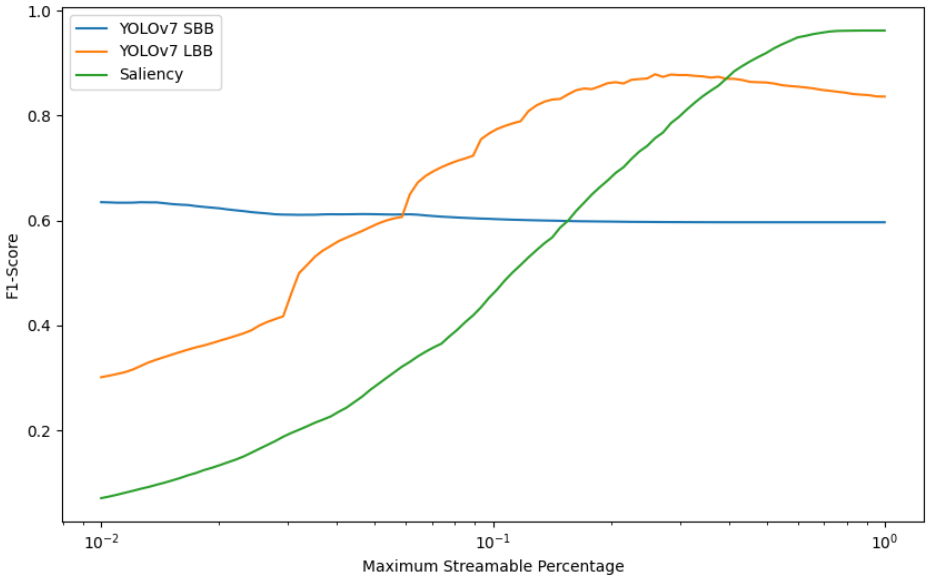


Figure 2.10: Logarithmic plot of the F1-score on the test set of the SeaDronesSee data set for saliency detection and two different versions of YOLOv7. The portion of the maximum streamable portion of the full resolution is plotted on the x-axis.

ing UAVs. In contrast to this technical discussion, the next chapters will be discussing some methods for improving object detection from UAVs.

Chapter 3

Leveraging Domain Labels in Object Detection on UAVs

3.1 Introduction

As discussed in Chapter 1, generic object detection has improved drastically over the last years (Zhao *et al.* (2019)), while object detection on images captured from UAVs still poses great challenges (Zhu *et al.* (2020a)). In this chapter, we will highlight the specific problem of variations across domains in UAV imagery, why this is particularly challenging for object detectors, and some approaches to alleviate this problem algorithmically.

For example, an object detector encounters images taken from varying altitudes. Therefore, the scales of objects vary enormously, often ranging from 10 pixels to over 300, see Chapter 5. At lower altitudes, objects are captured with more detail while at higher altitudes, more objects appear, but blurrier. Furthermore, modern UAVs are equipped with so-called gimbal cameras (Rajesh and Kavitha (2015)). These allow for capturing objects with various viewing angles (pitch axis). Moreover, a UAV's roll axis introduces yet more variation. As a result, objects are captured with diverse aspect ratios and orientations. In particular, top-down views often result in ambiguous object appearances, such as distinguishing between a car or a van.

Many more factors influence objects' appearances. These include but are not limited to: variations in weather and time, both affecting the illumination of objects; GPS location; camera sensor. Examples for the individual factors might be: images during rain vs. at sunny weathers; at day vs. at night; different backgrounds resulting from images taken in cities vs. rural areas; lens distortions from different cameras.

These variations become more critical when the interplay with different domains is considered. For example, in Fig. 3.1 the very same scenery is shown from altitudes 10m and 100m, respectively, and from viewing angles 10° (nearly horizontally facing) and 90° (top-down), respectively.

In contrast, many traditional data sets in other applications consist of less restricted-view data, such as COCO (Lin *et al.* (2014)) for everyday objects, KITTI (Geiger *et al.*

© 2024 IEEE. Reprinted, with permission, from Kiefer *et al.* (2021).

Figure 3.1: Example images of the data set POG, showing the same scenery taken from different perspectives (top: 10m, 10°, bottom: 100m, 90°).

(2013)) for autonomous driving and DOTA (Xia *et al.* (2018)) for remote sensing. Therefore, models trained on these data sets do not have to take the aforementioned domain variations into account.

Ultimately, the goal of object detection from UAVs is to detect objects in all of the considered domains. However, data sets are commonly unbalanced with respect to different domains (see Figure 3.2). Therefore, trained models are biased towards over-represented domains while failing to perform well in under-represented domains. As a result, even state-of-the-art models are underoptimized in the latter domains, as will become clear in Section 3.4.

In part, this is a consequence of using the commonly used metric average precision. This domain-agnostic metric favors models, which perform well in over-represented domains but may fail in others.

Motivated by these observations, we propose to leverage domain labels to alleviate this bias. While domain information is implicitly encoded in the captured images, it is also explicitly available from the UAVs' sensors: the altitude of the aircraft can be retrieved from the onboard GPS or barometer, the viewing angle from the gimbal pitch angle of the camera, and the time from an onboard clock. We propose to use these domain labels to train so-called expert models. These experts adapt to their respective domains to capture the domain-specific features. This multi-domain learning approach is in contrast to domain adaptation, which aims to eliminate these recognized types of domain. It is furthermore different from multi-task learning as we try to solve the same task across all domains. We show that these experts prove highly effective and efficient across various models, data sets and metrics.

In summary, our contributions are threefold:

- We analyze domain imbalance in three UAV object detection data sets and their effects on the overall model performance. We also propose a simple domain-sensitive metric to capture domain specific particularities.

- We propose a simple method, which leverages domain knowledge, to alleviate domain bias. We show that using this method we can significantly outperform domain-agnostic models without sacrificing speed. Further, we analyze the method on two established UAV object detection data sets.

- We capture and annotate a UAV object detection data set dubbed PeopleOnGrass (POG). We show that more precise domain labels can improve detection accuracy even further.

Deep learning-based object detection can roughly be divided into two categories: two-stage detectors, like Fast R-CNN or Faster R-CNN (Ren *et al.* (2016)), and the much faster, but less accurate one-stage detectors such as YOLO (Farhadi and Redmon (2018)) or EfficientDet (Tan *et al.* (2020)). While there is a large amount of research towards improving these object detectors, much of the research community focuses mainly on popular benchmarks, such as MS COCO (Lin *et al.* (2014)).

While research fields such as remote sensing used geo-spatial image data sets (e.g. satellite data), they are not that useful for object detection from UAVs, since they employ very low pixel per centimeter resolutions and vary very little in their altitude and angle information (Li *et al.* (2018)). Furthermore, a common practice in object detection from UAVs is still to use off-the-shelf detectors (Zhu *et al.* (2018)).

With the release of the UAVDT (Du *et al.* (2018)) and VisDrone (Zhu *et al.* (2018)) data sets, several works developed models specifically aimed at object detection from UAVs (Ševo and Avramović (2016); Sommer *et al.* (2017); Ding *et al.* (2018)). Many works focus on detecting small or clustered objects (Hong *et al.* (2019); Yang *et al.* (2019)).

With (Bashmal *et al.* (2018)), the concept of domains enters the field of object detection from UAVs, where a Siamese-GAN is introduced to learn invariant feature representations for labeled and unlabeled aerial images from two different domains. However, such a domain adaptation method's focus is to adapt the model from a fixed source domain to a fixed target domain. Fine-grained domains are utilized by (Wu *et al.* (2019)), where adversarial losses are employed to disentangle domain-specific nuisances. However, the training is slow and unstable, and domain labels are ignored at test time. Expert models are proposed in (Lee *et al.* (2019)) to account for objects with particular shapes (horizontally/vertically elongated, square-like). Since no domain labels are used in this work, they are formulated as a model ensemble too expensive to employ in multiple domains. A multi-domain learning approach for object detection is investigated in (Wang *et al.* (2019)), where the focus is on learning from multiple distinct data sets. Transfer learning (Zhuang *et al.* (2020)) is different in that it generally aims to learn invariant

Domain type	Domain name	Estimated ranges
Altitude	high (H)	80-100m
	medium (M)	30-80m
	low (L)	0-30m
Angle	bird-view (B)	70-90°
	acute angle (A)	0-70°
Time	day (D)	6am-10pm
	night (N)	10pm-6am

Table 3.1: Available domain labels in the data sets VisDrone and UAVDT and its ranges. Note that the ranges have been estimated by visual inspection since they have not been reported.

representations, whereas multi-domain learning preserves the domain-specific representations.

As opposed to the aforementioned works, we aim to leverage freely available environmental data from the drones' sensors. We try to leverage these so far overlooked domain labels at training and runtime to reduce the domain bias induced by highly imbalanced data sets.

3.2 Analyzing Domain Imbalances

In the following two subsections, we analyze domain imbalances and their consequences in two of the most popular UAV object detection data sets. First, we consider imbalances in the training set and then in the testing set.

3.2.1 Domain Imbalances in the Training Set

Imbalance problems in data-driven object detection have been known for a long time. However, most of the literature focuses on class, scale, spatial, and objective imbalances, like how much different tasks (e.g. classification or regression in the case of object detection) contribute to the loss function (Oksuz *et al.* (2020)). In contrast to many other applications areas, data in object detection from UAVs is versatile with respect to environmental domains.

So far, we loosely used the term 'domain' to depict a particular environmental state a UAV is in at the time of image capture. Some of these states give rise to some of the imbalances mentioned above: Altitude imbalances give rise to scale imbalances as object sizes directly correlate with the altitude an image is captured at. Also, foreground-background imbalances are affected by the altitude. Viewing angle imbalances give rise

to spatial and aspect ratio imbalances. However, there might be many other domain imbalances that may not directly relate to the aforementioned imbalances, such as lighting imbalances caused by the time or weather.

However, it is not clear what separates one domain from another. In fact, many environmental factors are continuous, such as the altitude or angle an image is captured at. Nevertheless, in current UAV object detection data sets, only coarse domain labels are reported. Two of the most established data sets, UAVDT and VisDrone, feature domain labels with coarse information about altitude, viewing angle and time as depicted in Table 3.1. Although these divisions seem arbitrary, they already help distinguish features in different domains, as will be seen in Section 3.4.

The large amount of varying domains causes data sets to be highly unbalanced with respect to these domains. Figure 3.2 shows the number of labeled objects in every domain for the UAVDT and VisDrone training sets. Note that a domain is a combination of one or more influencing variables. For example, the domain 'high' (H) + 'bird view' (B) + 'night' (N) in VisDrone contains 4,120 objects. Furthermore note, that we deliberately compared the number of objects and not the number of images because common object detection losses are back-propagated for every object instance – as opposed to every image.

In both data sets, many domain imbalances exist. For example, in both data sets, there are fewer labeled objects at night than at day. Both data sets show most objects from a horizontal viewing angle as opposed to from bird-view. These imbalances can be quite large. For example, in VisDrone, the domain H+B+N contains roughly only 1% ($\approx 4,120/343,205$) of objects, whereas the domain L+A+D contains roughly 33% ($\approx 114,504/343,205$). Even more extremely, in UAVDT, there are no objects in H+B.

These domain imbalances result in models being biased towards the over-represented domains. In turn, this may hamper models to predict objects in every domain accurately. In Section 3.3, we aim to propose a simple model family to diminish these biases.

3.2.2 Domain Imbalances in the Testing Set

While domain imbalances in the training set cause a trained model to be biased towards the over-represented domains, domain imbalances in the testing set cause a trained model to be rewarded for that behavior. If we want to faithfully measure the performance of an object detector across domains equally, we ought to include this in the corresponding metric. However, the conventional metric 'mean average precision' (mAP) does not capture the concept of a domain. Indeed, it is designed to be a general-purpose metric that weighs precision and recall. It is the area under the precision-recall curve averaged

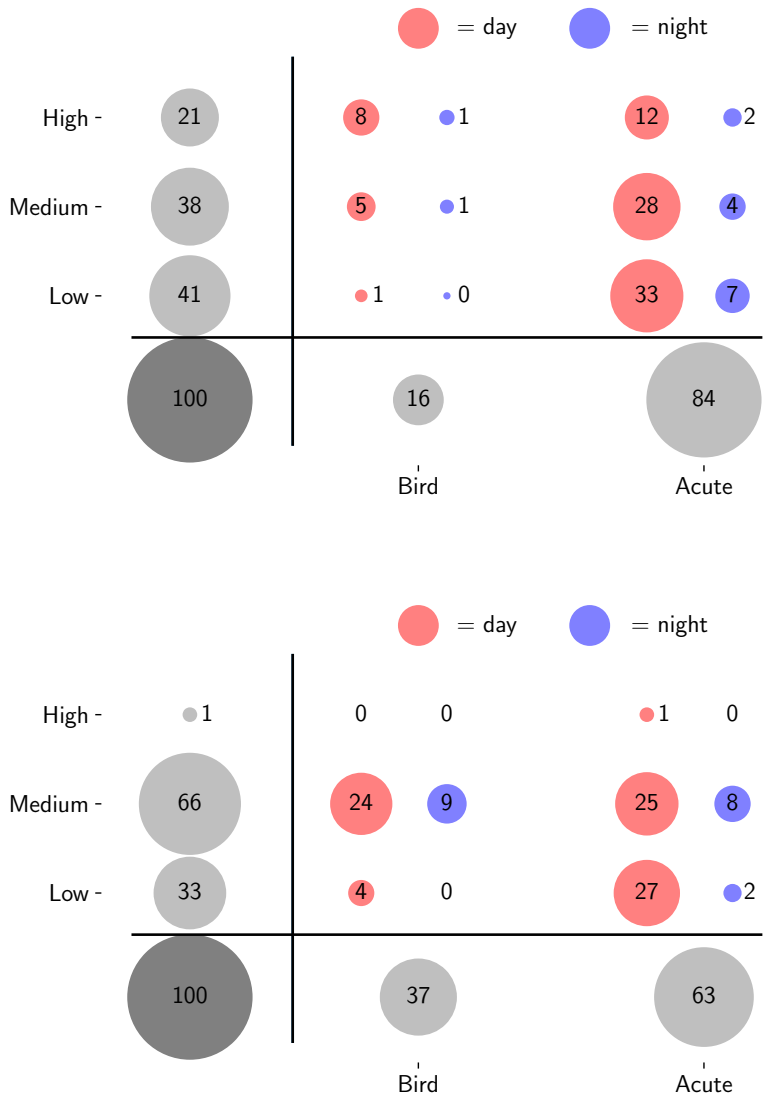


Figure 3.2: Distribution of objects across domains in the VisDrone (top) and UAVDT (bottom) training set. The lower left circle represents the size of the whole data set (100%), the other circles the relative size to it (rounded to the closest integer). The domains are high, medium, low, bird view, acute viewing angle, day and night, and combinations thereof.

over all classes $c \in \{1,\dots,C\}$ defined as follows:

$$\text{mAP} := \frac{1}{C}\sum_{c=1}^{C}\text{AP(c)} := \frac{1}{C}\sum_{c=1}^{C}\int_0^1 p_c(r)\mathrm{d}r, \tag{3.1}$$

where $p_c(r)$ denotes the precision for class c for a recall value r. True positives are determined by measuring the intersection-over-union (IoU) of the predicted bounding box and the ground truth. The threshold varies across data sets. Without any subscript, the value denotes the average value over thresholds from 0.5 to 0.95 in steps of 0.05 (Lin *et al.* (2014)). Because there are only finitely many predictions, the integral simplifies to a sum over the ordered object predictions.

To illustrate the severeness of mAP being domain agnostic, consider the following toy example: Suppose we have two distinct domains d_1 and d_2 in our UAV object detection data set. Let mAP_{d_1} and mAP_{d_2} be the mAP scores of a model trained on all data but evaluated only on d_1 and d_2, respectively. Denote by $s \in [0,1]$ the size of d_1 relative to the size of the whole data set $d_1 \cup d_2$. In Figure 3.3, we plot the mAP on $d_1 \cup d_2$ as a function of s for certain fixed values of mAP_{d_1} and mAP_{d_2}. Note that these curves depend on the distribution of true/false positives, true/false negatives and scores of the predictions and are therefore not unique.

From that hypothetical example, it is evident that small domains contribute very little to the overall mAP score. For example, consider the blue curve. In this case, $\text{mAP}_{d_1} = 0.1$ and $\text{mAP}_{d_2} = 1$. If the size of d_1 is less than 1/4 of the whole data set size, the overall mAP still is above 80%. This leads to overestimating models that just perform well on over-represented domains and underestimating models that perform well on under-represented domains.

Ideally, a UAV object detection data set is roughly balanced with respect to domains. However, as we saw in the subsection before, this condition often is violated. Therefore, the only way to obtain models that are robust across domains is to incorporate this domain performance into the metric. We propose to use the simple domain-averaged metric

$$\text{mAP}^{\text{avg}} := \frac{1}{D} \sum_{j=1}^{D} \text{mAP}_{d_j}, \tag{3.2}$$

where mAP_d denotes the mAP on domain $d \in \{d_1, \ldots, d_D\}$. To obtain well-calibrated models, we evaluate on both, mAP and mAP^{avg}. Note that it is a user question of how to weigh each domain. Non-uniform weightings of domains are possible as well. However, we argue that a priori all domains should be weighted equally to allow for cross-domain robust models. In the example from before, the dashed purple curve depicts mAP^{avg}, which is independent of the the sizes of each domain.

3.3 Multi-Domain Learning Approach

For a fixed model architecture, learning from multiple domains is inherently a trade-off. Large domains cause the model to be biased towards these domains. Our goal is to diminish this bias by leveraging freely available domain labels in a multi-domain learning setting.

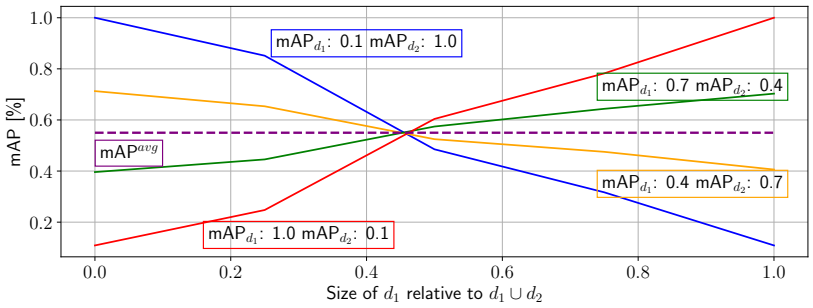


Figure 3.3: Hypothetical mAP values for a two-domain UAV object detection data set. mAP^{avg} is the average mAP over both domains as defined in equation 3.2.

In multi-domain learning, image samples $\{x_j\}$ with corresponding bounding box annotations $\{y_j\}$ are accompanied by a discrete domain indicator $d_x \in \{d_1, \ldots, d_D\}$ (which also is available at test time), such that a training sample is (x_j, d_{x_j}, y_j) and a test sample is (x_j, d_{x_j}). In particular, that means, we can leverage this domain information d_x at test time, which is the key to our expert models.

Motivated by (Caruana (1997)), we propose a multi-head architecture. Given a general object detector model, we share earlier layers across all domains and leave later layers domain-specific. This approach follows the empirical observations that earlier layers extract lower-level features, which are present across all domains, while later layers extract higher-level features, which may differ substantially across domains (such as the people in Fig. 3.1). Empirically, this is backed up by (Wang *et al.* (2019)), which shows that activations in later layers differ vastly.

This approach effectively allows the heads corresponding to smaller domains to learn domain-specific features without suffering from the domain bias induced by the domain imbalances that are favoring larger domains. Note that earlier layers may still be biased towards larger domains. However, as in earlier layers more general-purpose features are extracted (Yosinski *et al.* (2014)), this bias is less severe than in later layers.

From preliminary experiments, we found that it is best to split models not based on individual layers, but on groups of layers, which are known as stages or blocks (Wang *et al.* (2019)). These stages are model-dependent. For example, a Faster-RCNN with a ResNet-101 backbone consists of 5 stages prior to the region-of-interest pooling layer. That means, we share all stages across all domains until a certain stage is reached. From here on, we split the model into so-called experts. For simplicity, these experts are copies of the original model. Therefore, this approach does not need a reorganization of the model architecture and can be applied to many object detectors as will be seen from Section 3.4.

For illustrative purposes, see Figure 3.4. Here, a Faster-RCNN with ResNet-101 backbone is taken as an example. The first three stages are shared across all domains. Based on the domain label - in this case day or night - the corresponding expert branch is cho-

sen. We denote such a model as Time@3 because the available domains are based on the attribute 'time' and the model is shared until the third stage.

A priori, it is not clear, how many stages should be shared. We explore empirically which stages are to be shared in Section 3.4.

While the number of parameters scales linearly with the number of domains, the *inference speed stays constant* as only a single expert is evaluated at a time. Therefore, the experts effectively increase a model's capacity without hampering the inference speed. Furthermore, the experts' sizes are still small enough such that they all fit even in embedded GPUs' memory, as will be seen in section 3.4.

3.3.1 Simplified Training Realization

So far, the proposed approach may seem as if an adaptation to the model architecture was necessary. However, in the following we want to demonstrate that the expert approach can be implemented in every architecture. Furthermore, it introduces only very little training overhead.

Given an object detector and training pipeline, we train it until an early stopping criterion is met. That means, training it further would increase the validation error. Then, similarly to what is done in transfer learning (Zhuang *et al.* (2020)), we freeze the shared stages in order to transfer knowledge between domains and such that weights will not be biased towards the over-represented domains (Oksuz *et al.* (2020)). This is particularly beneficial for data sets with great domain imbalances, such as UAVDT and VisDrone. We only train the domain-specific stages further on each respective domain. We split a subset from the training set for that particular domain and use it as the validation set. We train until the validation error increases again. Finally, we take the weights corresponding to the lowest validation loss as our final weights for that expert. Even though the number of trainable parameters shrinks, we want to emphasize that having a validation set is especially critical in this case to avoid overfitting on the small domains.

Post-training the domain-specific layers on their corresponding domains introduces little overhead to the overall pipeline. This is because only a small number of layers needs to be trained which decreases the time for the backward pass because only parts of the weights need to be back-propagated and the freed GPU memory space can be used to increase the batch size. Furthermore, training for different domains can be done in parallel. We report actual training times for various experiments in Section 3.4.

3.3.2 Introducing a Multi-Modal Data Set

Lastly, we would like to note that there are no publicly available data sets for object detection from UAVs that include precise domain labels regarding altitude and viewing angle. E.g. (Bozcan and Kayacan (2020)) includes limited altitude values between $0 - 30\ m$. We argue that this is a major impediment in the development of domain-aware models since these two factors majorly contribute to appearance changes.

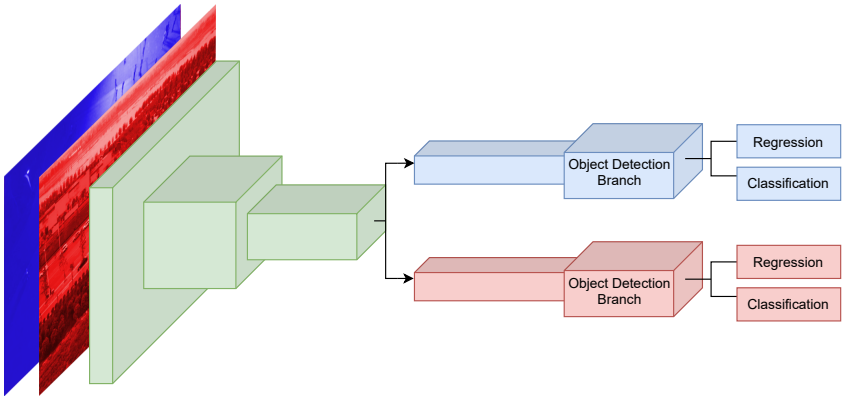


Figure 3.4: Illustration of a Time@3 model with day and night experts. The time is split into two domains, day (red) versus night (blue), where green outputs represent the shared stages (first, second, third). Every image is passed through the shared green stages. Then it is checked whether it is a day or night image and consequently passed through the red or blue stages, respectively.

For that reason, we record the experimental data set PeopleOnGrass (POG) containing 2,9k images (3840×2160 pixels resolution), showing people from various angles and altitudes varying from 0° (horizontally facing) to 90° (top-down) and 4 m to 103 m, respectively, each labeled with the precise altitude and angle it was captured at. See Figure 3.5 for a distribution of objects. Further metadata, such as GPS location, UAV speed and rotation, timestamps and others are also included. We use a DJI Matrice 210 equipped with a Zenmuse XT2. The meta data is obtained through DJI's onboard software developing kit. Accompanied with every frame there is a meta stamp, that is logged at 10 Hertz. To align the video data (30 FPS) and the time stamps, a nearest neighbor method was performed. The following data is logged and provided for every image/frame read from the onboard clock, barometer, IMU and GPS sensor, respectively:

- current date and time of capture
- latitude, longitude and altitude of the UAV
- camera pitch, roll and yaw angle (viewing angle)
- speed along the x-, y and z-axes

We want to emphasize that the meta values lie within the error thresholds introduced by the different sensors but an analysis is beyond the scope of this dissertation (see Sitemark (2020) for an overview).

See Figure 3.1 for example images. We manually and carefully annotated 13,713 people. We note that this is a simple real-world data set, suffering from fewer confounders than large data sets, which is ideal for testing out the efficacy of multi-modal methods.

This data set is available to the community[1] and hopefully will benefit the development of multi-modal models.

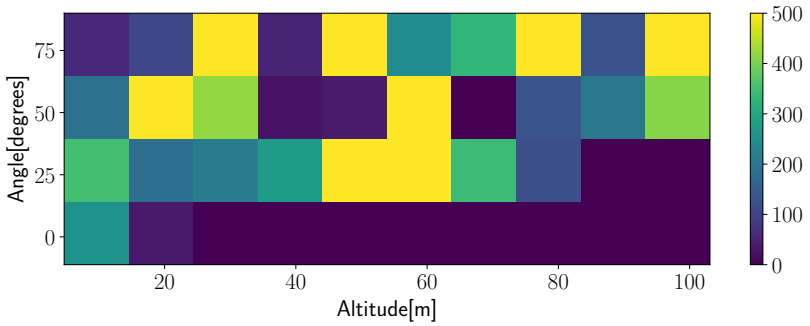


Figure 3.5: Distribution of objects in PeopleOnGrass (POG) across different levels of altitude and camera pitch angles.

3.4 Experimental Results and Ablations

In the first two sets of experiments, we show how leveraging domain labels on UAVDT and VisDrone improves multiple model architectures' performances. Furthermore, we investigate the effect of different splitting strategies and ablations. Lastly, we show that finer domain splitting is possible in the case of the data set POG.

3.4.1 VisDrone

We evaluate our models using the official evaluation protocols, i.e. AP_{70} for UAVDT and mAP and mAP_{50} for VisDrone, respectively. Furthermore, we report results on individual domains and the domain-averaged metric from Section 3.2.2, i.e. AP_{70}^{avg} and mAP_{50}^{avg} over all respective domains to measure the universal cross-domain performance. The subscript 50 and 70 denote the intersection-over-union (IoU) a prediction needs to have with a ground truth bounding box in order to be counted as a true positive. Note that we leave out the 'm' in 'mAP' for UAVDT since it contains only one class.

Furthermore, we report the additional training times Δt in percent (rounded to integers) to train a model longer than its baseline, i.e. $\Delta t = 10\%$ would mean that it takes additional 10% to train a model further than its baseline.

The object detection track from VisDrone consists of around 10k images with 10 categories. All frames are annotated with domain labels regarding altitude (low (L), medium (M), high (H)), viewing angle (front, side, bird (B)) and light condition (day (D), night

[1] https://cloud.cs.uni-tuebingen.de/index.php/s/yFztfJePREqj4om

	L	M	H	mAP_{50}	mAP	mAP_{50}^{avg}	Δt
DE-FPN (Zhu *et al.* (2018))	49.1	49.7	36.0	48.6	26.1	44.9	–
Altitude@0	49.4	49.6	35.5	48.3	25.9	44.8	12%
Altitude@1	49.5	49.7	35.7	48.5	25.9	45.0	11%
Altitude@2	49.5	49.9	36.1	48.7	26.1	45.2	11%
Altitude@3	50.2	**50.2**	36.8	49.2	26.6	45.7	10%
Altitude@4	**50.7**	**50.2**	**37.5**	**49.9**	**27.4**	**46.1**	8%
Altitude@5	50.5	50.0	**37.5**	49.7	27.0	46.0	7%
	B	A					
DE-FPN (Zhu *et al.* (2018))	38.0	49.0		48.6	26.1	43.5	–
Angle@4	**39.6**	**49.8**		**49.4**	**27.0**	**44.7**	6%
	D	N					
DE-FPN (Zhu *et al.* (2018))	48.5	52.0		48.6	26.1	50.2	–
Time@4	**49.0**	**52.6**		**49.0**	**26.6**	**50.8**	7%

Table 3.2: Several domain expert results for various freezing strategies on VisDrone. Altitude@x means that all stages until the *x*th. stage are shared. As described in Section 3.4.1, Δt is the additional training time as a percentage of the time needed to train the respective baseline model. The domains are abbreviated as follows: A = acute viewing angle, B = bird's-eye view, D = day, N = night, L = low capture-altitude, M = medium capture-altitude, H = high capture-altitude.

(N)) (Wu *et al.* (2019)). Note that we fuse the two domains "front" and "side" into a single domain "acute angle (A)", as, at test time, we can only distinguish between bird view and not bird view based on the camera angle. We reimplement the best performing single-model (no ensemble) from the workshop report, DE-FPN (Zhu *et al.* (2018)), i.e. a Faster R-CNN with a ResNeXt-101 64-4d (Xie *et al.* (2017)) backbone (removing P6), which was trained using color jitter and random image cropping achieving 48.7% mAP_{50} on the test set. To compare with Wu *et al.* (2019), we evaluate our models on the unseen validation set, on which our implementation yields 48.6% mAP_{50}.

From Table 3.2, we can make four observations: First, the altitude-experts improve over the baseline DE-FPN on the whole validation set and on all domains individually if the first three or more stages are shared. The performance drop of Altitude@0 and Altitude@1 is likely caused by overfitting on the small domain H, on which the performance drop is -0.5 mAP_{50}. Note that Altitude@0 essentially has a separate model for each domain. Second, there seems to be an upward trend in performance, peaking at the fourth stage and dropping at the fifth stage. Third, improvements are seen for all experts: +1.3, +0.8 and +0.4 mAP_{50} for the Altitude-, Angle- and Time-experts, respectively. Furthermore, the performance improvements are also seen in the domain-sensitive metric mAP_{50}^{avg}, yielding +1.2, +1.2 and +0.6 points for the respective experts. Fourth,

	$\downarrow + \rightarrow$	L	M	H	mAP_{50}	mAP	mAP_{50}^{avg}	Δt
DE-FPN (Zhu *et al.* (2018))	B	84.6	42.5	35.6	48.6	26.1	50.5	–
	A	49.1	50.0	41.2				
Altitude-angle@4	B	**87.4**	**44.8**	**39.6**	**49.0**	**26.3**	**52.3**	10%
	A	**49.7**	**50.1**	**42.2**				
DE-FPN (Zhu *et al.* (2018))	B+D	84.6	42.5	35.6	48.6	26.1	50.9	–
	A+D	49.0	50.2	41.2				
	A+N	52.8	51.6	–				
Altitude-angle-time@4	B+D	**87.5**	**44.8**	**39.6**	**49.6**	**26.8**	**52.9**	11%
	A+D	**50.1**	**50.6**	**42.2**				
	A+N	**54.4**	**56.5**	–				

Table 3.3: Results in mAP_{50} on specific domains for multi-dimension experts on VisDrone. The domains are abbreviated as in Table 3.2. For example, the Altitude-angle-time@4-expert achieves 54.4 mAP_{50} on the domain L+A+N (low altitude, acute viewing angle, and at night).

the additional training time Δt is low, with $8\%, 6\%$, and 7% for the most accurate domain experts. As it yielded the best results, we always freeze until the 4th stage for VisDrone from here on.

Table 3.3 shows that sharing along two and three domain dimensions is advantageous. The Altitude-angle@4-experts and the Altitude-angle-time@4-experts improve DE-FPN on all domains individually and overall. In particular, we obtain a +1.8 and +2 mAP_{50}^{avg} increase, respectively. The standard metrics mAP and mAP_{50} show an improvement as well, albeit a lower one, which is attributed to the failure of these metrics to capture domain imbalances in the validation set (see Figure 3.2).

This contrast is shown by the most significant improvements occurring in underrepresented domains, suggesting a reduction in domain bias. For example, the Altitude-angle-time@4-experts improve the performance on the domains M+A+N and H+B+D, which only contain roughly 4% and 8% of all objects (see Figure 3.2), from 51.6 mAP_{50} to 56.5 mAP_{50} and 35.6 mAP_{50} to 39.6 mAP_{50}, respectively.

	mAP_{50}	mAP	mAP_{50}^{avg}	Δt
DE-FPN (Zhu *et al.* (2018))	48.6	26.1	49.7	–
Altitude-time@4	**49.1**	**26.3**	**51.5**	11%
DE-FPN (Zhu *et al.* (2018))	48.6	26.1	50.1	–
Angle-time@4	**49.2**	**26.4**	**51.9**	13%

Table 3.4: Altitude-time@4 and Angle-time@4 experts on the VisDrone validation set.

	B	A	mAP_{50}	mAP_{50}^{avg}
EfficientDet-D0	21.5	24.9	26.3	23.2
Angle@backbone	**22.1**	**26.2**	**27.6**	**24.2**

Table 3.5: EfficientDet-D0 Angle experts on VisDrone validation set.

	L	M	H	AP_{70}	AP_{70}^{avg}	Δt
ResNet-101-FPN (Wu *et al.* (2019))	61.9	58.1	**24.1**	**49.4**	48.0	–
Altitude@2	**62.5**	**60.5**	**24.1**	**49.4**	**49.0**	10%
	B	A				
ResNet-101-FPN (Wu *et al.* (2019))	28.9	59.1		49.4	44.0	–
Angle@2	**33.6**	**60.4**		**50.4**	**47.0**	9%
	D	N				
ResNet-101-FPN (Wu *et al.* (2019))	51.4	50.6		49.4	51.0	–
Time@2	**53.4**	**54.1**		**50.1**	**53.8**	10%

Table 3.6: Domain experts on the UAVDT test set.

Similar observations can be made from Table 3.4, where the Altitude-time@4- and Angle-time@4-experts both improve by +1.8 mAP_{50}^{avg}.

To further test our approach in real-time scenarios, we choose the current best model family on the COCO test-dev according to Papers with Code (2021), i.e. EfficientDet (Tan *et al.* (2020)), and take the smallest model D0 as our baseline model. We employ it on the NVIDIA Jetson AGX Xavier suitable for on-board processing (Ditty *et al.* (2018)). For that, we convert the trained model to half-precision using JetPack and TensorRT (Vanholder (2016)) and set the performance mode to MAX-N. The inference speed is reported in frames per second (FPS) averaged over the validation set. Similar to (Ringwald et. al. (2019)), the FPS values do not include the non-maximum suppression stage as TensorRT does not supported it yet. Keeping the image ratio, the employed longer image side is 1408 pixels for training and testing.

We freeze the whole backbone and only leave the box-prediction net (Tan *et al.* (2020)) domain-specific. As shown in Table 3.5, sharing the backbone yields an improvement of 1.3 point mAP_{50} and 1 point mAP_{50}^{avg} for the angle experts. Both models run at 21.8 FPS, suitable for live on-board processing. With all pre- and post-processing steps, we obtain a frame rate of 18.1 FPS.

3.4.2 UAVDT

The UAVDT benchmark data set contains around 41k annotated frames with cars, busses and trucks. Similar to Wu *et al.* (2019), we fuse all vehicle classes into a single vehicle

	B	A	AP_{70}	AP_{70}^{avg}
ResNet-101 (Wu *et al.* (2019))	27.1	54.4	45.6	40.1
NDFT (Wu *et al.* (2019))	28.8	56.0	47.9	43.4
Angle@2	**31.6**	**58.6**	**48.6**	**45.1**

Table 3.7: Results for ResNet-101 backbone on UAVDT.

	AP_{70}	FPS	AP_{70}^{avg}
EfficientDet-D0	17.1	**21.8**	16.7
UAV-Net (Ringwald et. al. (2019))	26.2	18.3	–
Altitude@backbone	**38.1**	**21.8**	**37.0**

Table 3.8: Altitude experts results on UAVDT test set.

class. All frames are domain-annotated like VisDrone. To compare our experts, we trained a Faster R-CNN with ResNet-101-FPN similar to Wu *et al.* (2019), which report 49.1 AP_{70} on the testing set. We obtain 49.4 AP_{70} on the testing set and we compare with that value.

As Table VI shows, the Angle@2- and Time@2-experts improve performance over the baseline on both metrics. In particular, the Angle@2-expert improves the baseline by +3 points AP_{70}^{avg}. Furthermore note, that there is not an accuracy increase in domain H, since there are almost no training images available ($\approx 1\%$).

We also demonstrate that the performance gain using expert models does not vanish as we switch to another backbone, e.g. ResNet-101. As shown in Table VII, the angle experts yield an increase in +3 AP_{70} and +5 AP_{70}^{avg} and even outperform NDFT (Wu *et al.* (2019)), an approach using adversarial losses on domain labels.

Finally, we also test a real-time detector on UAVDT. Similar as for VisDrone, Table VIII shows how the altitude experts with shared backbone can regain precision that has been sacrificed to the high speed of the D0 model. The large improvement of +21.0 AP_{70} is likely caused by the domain bias induced by the heavy altitude imbalance of UAVDT (see Figure 3.2), which the experts are successful to mitigate.

In particular, we set a new state-of-the-art performance for real-time detectors on embedded hardware by improving upon Ringwald et. al. (2019) by +11.9 AP_{70}, while being 3.5 FPS faster. Note that they tested on different embedded hardware.

3.4.3 POG: Baseline and Expert Results

Finally, we test the expert approach on our own captured data set POG. For future reference, we establish an EfficientDet-D0 baseline, which can run in real-time on embedded hardware such as the Xavier board. Finally, we employ altitude experts with shared

	AP_{50}	AP	AP_{50}^{avg}
EfficientDet-D0	82.0	36.4	82.9
3xAltitude@backbone	86.2	40.3	86.0
6xAltitude@backbone	**87.9**	**40.8**	**88.1**

Table 3.9: (Finer) Altitude experts results on POG test set.

backbone to showcase the effectiveness of a multi-domain learning approach on finer domains.

We split the altitude range (4m – 103m) into three and six *equidistant* domains, respectively. That is, our domains are

1. $d_1 = (4, 37)$, $d_2 = [37, 70)$, $d_3 = [70, 103)$
2. $d_1 = (4, 20.5)$, $d_2 = [20.5, 37)$, $d_3 = [37, 53.5)$,
   $d_4 = [53.5, 70)$, $d_5 = [70, 86.5)$, $d_6 = [86.5, 103)$,

respectively. We denote the corresponding experts as 3xAltitude (1.) and 6xAltitude (2.), respectively. As before, we freeze the backbone and report results for the fast EfficientDet-D0. Table IX shows that the baseline achieves 82.0 AP_{50}, which the experts improve by +4.2 and +5.9 AP_{50}, respectively, showing that experts further benefit from finer domain splits (6xAltitude +1.7 AP_{50} compared to 3xAltitude).

3.5 Conclusion and Limitations

In this chapter, we successfully applied a multi-domain learning method to object detection from UAVs. We proposed and analyzed expert models, leveraging domain data at test time. Although these expert models are conceptually simple, they achieve domain awareness and consistently improve several heavily optimized state-of-the-art models on multiple data sets and metrics. In particular, our EfficientDet-D0 altitude expert yields 38.1% AP_{70} on UAVDT, making it the new state-of-the-art real-time detector on embedded hardware.

However, we believe that domain labels in UAV object detection can be exploited even more. In particular, the assumption that domains are regarded as equally discrete may be overly strict. An open question remains the interplay in between different domains on a deeper level. For this matter, incorporating softer boundaries between domains could be a promising direction. Furthermore, different sampling strategies, such as oversampling small domains, could be investigated. An additional insight from this chapter is, that the bird's eye view domain is one of the most challenging. This observation serves as a motivation for the next chapter, which specifically aims to tackle improving object detection on this particular domain.

Chapter 4

Gaining Scale Invariance in UAV Object Detection by Adaptive Resizing

4.1 Introduction

From last chapter's Table 3.2 we observe, that the bird's eye view domain is especially challenging in UAV object detection data sets[1]. One of the main reasons for this discrepancy is the versatile application areas of UAVs with mounted cameras which lead to vast differences in the altitude above the ground of the UAV at the time of capture (capture-altitude). For example, in traffic surveillance applications, the altitudes can vary from 5 to 100 meters (Zhu *et al.* (2018)), while in search and rescue tasks, the span may be as large as 5 to 260 meters, see Chapter 5. This variance in altitudes results in a variance in objects' sizes. While humans are believed to have a scale-invariant perception and internal representation of objects (Han *et al.* (2020)), current object detectors do not. In fact, scale variation is a major cause for poor detection (Singh and Davis (2018)). While there is a corpus of works addressing this issue for generic object detection (Singh and Davis (2018); Huang *et al.* (2019); Kokkinos and Yuille (2008); Liu *et al.* (2019)), it remains a complicated problem to solve.

On the other hand, in UAV bird's eye view object detection, objects' sizes mainly depend on the UAV's altitude. In turn, the altitude information is freely available via the UAV's onboard barometer and GPS sensor. Current object detectors ignore this information entirely. We argue that it is utterly helpful to include this valuable information as it tells us about the objects' sizes and how closely we have to look for objects. Analogous to humans' intrinsic understanding of their environment (Epstein and Baker (2019)), we can incorporate that environmental information in the object detection pipeline to achieve a scale-invariant understanding of the scene.

Furthermore, ignoring the scale information of objects leads to models learning differ-

[1] Table 3.2 also shows, that the domain of 'high' images is a particularly difficult one for object detectors. However, the proportion of bird's eye view images within this domain is relatively high, as shown in Figure 3.2. Therefore, improving the performance on the BEV portion will improve the results on images with high capture-altitude as well.

© 2024 IEEE. Reprinted, with permission, from Messmer *et al.* (2022).

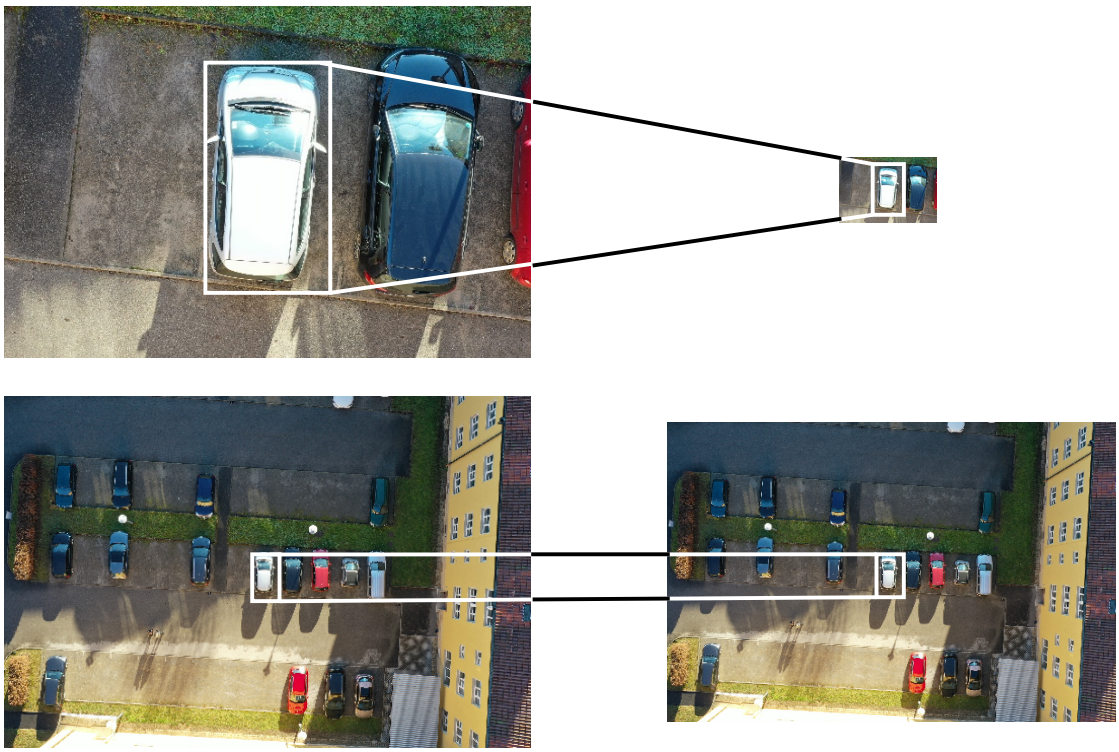


Figure 4.1: Example of the resizing process. On the left, we have two images from a possible UAV data set; the top one is captured at $10\,m$ flight altitude, the bottom one at $60\,m$. On the right, we again have both images resized according to their respective height. The bottom one stays roughly the same while the top one is resized by a large factor. Note how the bounding box of the white car at the center of each respective picture is equal in size after resizing.

ent representations of the very same objects if they are perceived at sufficiently different altitudes (and thus scales). In turn, this results in potential redundancy among the learned features. However, as onboard computation capabilities of UAVs are usually smaller than those of high-end consumer graphics cards, highly condensed models (with lower capacity) are needed.

Lastly, for higher altitudes, it is inevitable to provide large image resolutions to detect smaller objects (Varga and Zell (2021)). However, these large resolutions may be redundant in lower altitudes. Thus, an altitude-aware method benefits the inference time even further.

In this chapter, we tackle these problems by introducing a method we call Adaptive Resizer. At its core, this is a preprocessing technique designed to ensure that two arbitrary instances of the same class are of the same size throughout the entire data set. We do this by adaptively resizing each image depending on the altitude it has been captured in a principled way before passing it to an object detector.

This achieves two things: first, the object detector itself does not need to be scale-invariant. Second, the inference is much quicker because images taken at low altitudes are downscaled by a significant factor because they feature the largest objects.

Our approach works for the special case of bird's eye view (BEV) images, i.e. images facing directly downwards, which form the most challenging subset (Wu *et al.* (2019)). However, we also show the usefulness in general UAV object detection. To summarize, the key contributions of this chapter are as follows:

- We propose a novel height-adaptive image preprocessing method, which improves UAV bird's eye view object detection performances in both accuracy and inference speed and is applicable to all state-of-the-art object detectors.
- We construct a fast object detector for embedded applications that builds upon this method.

Object detectors can broadly be divided into two categories; one-stage and two-stage detectors. Two-stage detectors (Ren *et al.* (2016); Girshick (2015)) are generally more accurate and therefore occupy the first places on established leader boards (Du *et al.* (2019)). However, their inference speed is generally a lot lower than that of one-stage detectors (Redmon et. al. (2016); Zhou *et al.* (2019); Tian et. al. (2019); Lin *et al.* (2017)), which makes the latter more suitable for onboard object detection scenarios. Most recently, there are also transformer-based object detectors performing very well in generic object detection (Liu *et al.* (2021); Zhu *et al.* (2020b); Carion *et al.* (2020)). They have, however, not proven to be useful for UAV or BEV object detection so far.
The closest method to ours is Kim et. al. (2020). There, images are also resized in accordance with the height. However, the authors resize every image to the same resolution (an average over the data set) while we calculate an individual size for each image. Furthermore, they merely test their method on class agnostic detection tasks.
While the authors in Singh and Davis (2018) analyze the problem of scale invariance in CNN's in great depth, their solution employs an image pyramid, which is not well-suited for real-time detection. Another approach is presented in Yang *et al.* (2019), where the authors try to detect clusters of potential targets and then predict the scale offset before regressing the objects in each cluster more accurately. A drawback is the need for ground truth labels of clusters. Furthermore, the sequential use of multiple different networks is computationally expensive, while our approach estimates scales for the whole image deterministically.
Most papers tackling real-time object detection in general (Redmon et. al. (2016); Farhadi and Redmon (2018)) or on mobile platforms (Ringwald et. al. (2019)) design a whole network architecture. Meanwhile, this chapter introduces a method applicable to most modern object detectors, improving their speed and detection performance.
The authors of Wu *et al.* (2019) propose to apply adversarial learning techniques to the meta-data of UAV imagery. While they achieve good results, they only use the meta-data

during training and not during validation. Also using it at test time can improve performance even further, as we show.
One recent work exhaustively examines how feature pyramid networks work and how object detectors (don't) benefit from them (Chen *et al.* (2021)). However, compared to their approach we can choose a rather simple method to cut the feature pyramid network and therefore save on computational cost. That is, because the approach in (Chen *et al.* (2021)) aims at generic object detection, while we go for the special case of BEV object detection.

4.2 Method

The *Adaptive Resizer* is a preprocessing strategy designed to address bird's eye view (BEV) object detection, i.e. object detection from UAVs, where the angle of view is pointing downwards in a right angle. The Adaptive Resizer rescales every image in a principled manner to diminish the scale variance problem in BEV object detection. In this section we describe how this works on a technical level.

One problem in BEV object detection is that object instances of the same class appear in vastly different sizes; see, for example, the left two images in Figure 4.1. This scale variance is primarily attributed to the altitude an image is captured at (*capture-altitude*). A vanilla object detector is not aware of the fact that it observes instances of the same class (or even the same object like in Figure 4.1) but at different scales (Lin et. al. (2017)). Therefore, it learns different representations for different scales of the same object. That means some of the capacity of the detector is tied up in learning these different representations. One could either make use of this capacity in a different way or use a smaller object detector to increase inference speed. Furthermore, an object detector that can make use of differently scaled training samples of the same objects makes more efficient use of the training samples.
However, the advantage of UAV object detection is the availability of freely available meta-data generated by the UAV during flight. That includes data like the camera's angle, capture-altitude, or time-stamp. The necessary meta-data for the Adaptive Resizer is the capture-altitude. Unique to BEV object detection is that all instances of the same class are roughly equal in size on any single image, because all objects are about the same distance from the camera.

Building on that, the Adaptive Resizer achieves its goal (scaling each object of the same class to the standard size) by resizing each image according to its height. For this, the relevant determinant is the

Ground Sample Distance (GSD)

To define the GSD of an image, let p be its centre pixel. The definition of the GSD is the side length of the area on the ground that p depicts. For the calculation of the GSD we

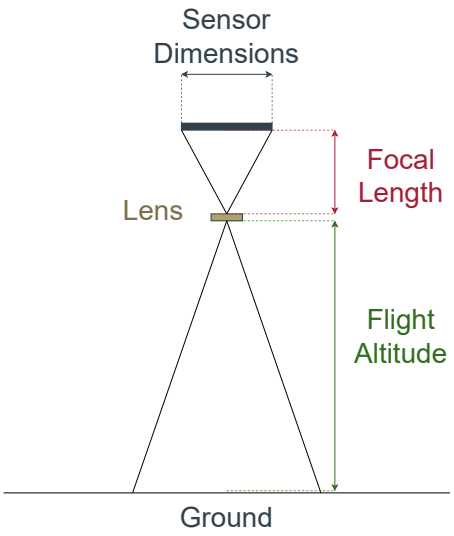


Figure 4.2: Pictogram of a camera setup mounted on a UAV.

assume a fixed camera setup on the UAV. We can readily deduce the following formula from fundamental properties of the camera geometry (see Figure 4.2).

$$\mathrm{GSD} = \frac{S}{L} \cdot \frac{A}{I}. \tag{4.1}$$

S refers to the optical sensor's side length, while A denotes the capture-altitude. L refers to the camera's focal length, and I denotes the captured image's side length. With a fixed camera setup, the only varying factors in Equation (4.1) are the distance above ground (A) and the image size (I). Therefore, if we ensure that the ratio $\frac{A}{I}$ is constant over the data set, the GSD is also constant across the entire data set. Ensuring that the GSD is constant over the data set is just a reformulation of the Adaptive Resizer's objective to alleviate the scale variance problem within each class.

To implement adaptive resizing and make use of Equation (4.1), we need to fix a reference class from the data set to determine the desired GSD, e.g. 'car'. Also, we fix a reference area, which is the goal size for all objects of the reference class after resizing. Then, there are two ways; ideally, we know how large the standard representative of this reference class is. For example, if we fixed 'car' and know that the average car in the data set is $4\,m \times 2\,m$ while our reference area is 32 px $\times$ 32 px, we get the desired GSD in two easy steps:
First, we compute the reference area with the same aspect ratio as the average car. Here, this is roughly 45.25 px $\times$ 22.63 px. Then the desired GSD is $\frac{4}{45.25}\,m/\mathrm{px}$. If we plug that into Equation (4.1), we get the image size to resize to by solving for I.
If we do not know the size of the average car in our data set, we can still apply the Adaptive Resizer. In this case, we compute the average area of the bounding boxes of

our reference class for a given image from the data set. Then we resize the image for this average to match our reference bounding box size. So if $\tilde{I}$ is the size of the image, M is the mean over the bounding box areas, and R is the reference area, the image size to resize to is computed by

$$I = \frac{R}{M} \cdot \tilde{I}. \tag{4.2}$$

The second method, taking the image-wise means of the bounding boxes, works consistently. However, the first method is more desirable as it filters annotation mistakes. Also, the second method does not work for images without instances of the reference class.
For an illustration of the whole process, see Figure 4.1. This method works together with any modern deep learning-based object detector since our approach is a preprocessing step.

Also note that we disregard effects of lens distortion and perspective projection as, we argue, these are minor compared to the general relation of altitude to object size.

Height Transfer

An additional feature of models employing Adaptive Resizing is their ability to generalize well to images captured at altitudes that were not represented in the training data. By the above discussion, without Adaptive Resizing, a model learns different representations for object instances with varying scales. Therefore the model learns separate representations for objects belonging to the same class but appearing on images from different altitudes. Consequently, the objects on which the model without Adaptive Resizer did not train can not be recognized during testing.

In contrast, a network endowed with the Adaptive Resizer learns representations for every class at one specific scale. Therefore, the capture-altitude affects detection performance very little as long as every image is resized like our presented method is doing. Hence the Adaptive Resizer allows for transferring knowledge in between altitudes, for example a network may learn from images taken between $0\,m$ and $50\,m$ above ground and then perform well on images captured in between $50\,m$ and $100\,m$, if they are scaled accordingly.

This is a way to overcome data set imbalances as discussed in Chapter 3 and in particular Figure 3.2. The above discussion also shows, that the Adaptive Resizer is a highly specialized technique for transfer learning (Weiss *et al.* (2016)), making it possible to use object detection models on data, which is out of the distribution of the training data. The claims made in this subsection are proved empirically in section 4.4.5.

4.2.1 Building a Detector for Embedded Deployment

In this section we will leverage the new features the Adaptive Resizer brings to an object detector to build a fast detector for BEV imagery meant for embedded use. We start with

an EfficientDet–*D*0 (Tan *et al.* (2020)) in order to have a fast state-of-the-art detector and then omit the parts that we argue are not necessary in combination with adaptive resizing. EfficientDet is a family of models which are building on EfficientNet-backbones (Tan and Le (2019)) and are therefore scalable in parameters, ranging up to EfficientDet–*D*7. Here, a higher number stands for the model being larger and more accurate, while a lower number means it is faster. We choose this detector because it is the smallest representative of its family, which in turn is the current AP50-state-of-the-art on COCO (Papers with Code (2021)).

EfficientDet–*D*0 employs a Feature Pyramid Network (FPN) (Lin et. al. (2017)), as is standard for modern object detectors (which are not transformer-based (Zhu *et al.* (2020b); Carion *et al.* (2020))). The FPN aims at making the detector perform well on multiple different levels of scale, because Convolutional Neural Networks (CNNs) are not inherently scale-invariant (Singh and Davis (2018)). An FPN extracts feature representations from the backbone network at different levels of depth, see Figure 4.5. Deeper ones are responsible for detecting larger objects because of their bigger field view (FOV), while earlier ones are being used to detect smaller objects. This is usually realized by distributing a vast number of prior boxes, called anchor-boxes, each corresponding to one feature map from the FPN. An anchor-box corresponding to a feature level of the FPN means, that the head from this feature level is used to classify and regress this anchor-box. In the case of EfficientDet, the FPN employs five different feature levels. These levels are responsible for detecting objects at exponentially increasing sizes; EfficientDet uses $(32, 64, 128, 256, 512)$. Therefore EfficientDet's anchor-boxes are of these sizes.

While this is an appropriate choice for data sets featuring everyday objects like COCO (Lin *et al.* (2014)) or Pascal VOC (Everingham et al. (2015)), in BEV object detection, four out of these five feature levels are almost unused for each given image, see Table 4.6. This is due to the object sizes the respective feature maps are looking for and because in one given image from the BEV portion of a UAV data set all objects of a given class are (roughly) equal in size.

However, the network itself does not need to be scale-invariant, if all the objects in the data set are of the same size. For BEV images, all instances of any given class on one single image are a priori roughly equal in size, because all of them are about the same distance from the camera. Consequently, the only remaining problem is the objects' difference in scale between different images, precisely what the Adaptive Resizer aims at.

Consequently, we eliminate the feature pyramid network (FPN) from our model and only use the earliest feature map of those extracted from the backbone network. For an EfficientDet–*D*0 this reduces the number of parameters from around 4m to roughly 0.5m. This also leads to a large boost in inference speed, see section 4.5.

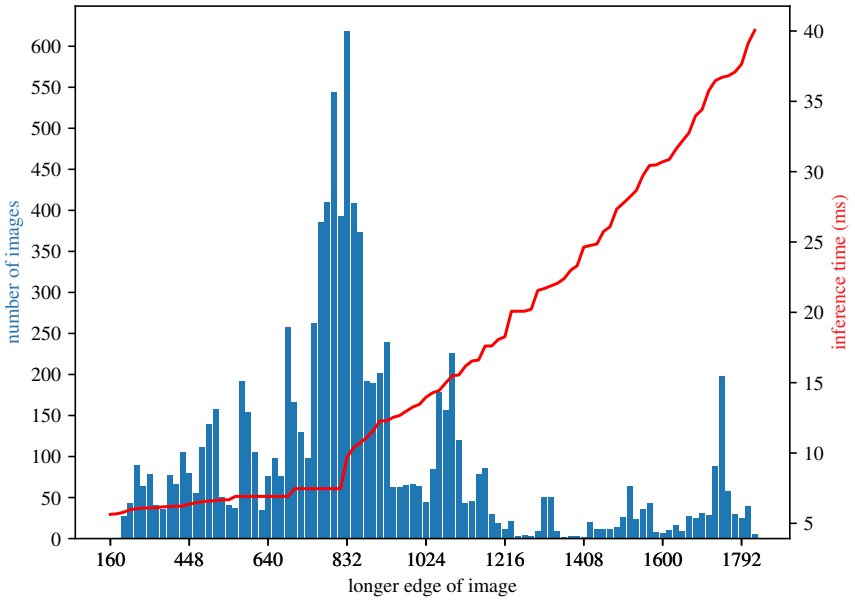


Figure 4.3: Distribution of image sizes after applying Adaptive Resizer on the UAVDT data set and the resulting inference time. The x-axis denotes the longer edge of the image, aspect ratios are kept during this process. The y-axis denotes the quantity in blue and the inference time in red.

4.3 Proof of Concept on Synthetic Data

In order to prove the assertions regarding the benefits of adaptive resizing we made thus far, we conduct experiments on a synthetic data set. The data set is supposed to contain very simple data. To achieve this, each image within the data set is entirely black with the exception of n objects scattered across the image in a random manner. Here, n represents a random number drawn uniformly from $\{2,3,4,5\}$. The objects themselves are images drawn uniformly at random from the MNIST data set (Deng (2012)). To explore the effectiveness of adaptive resizing, the objects scattered across the image are not placed there at their original size but instead scaled to various sizes. Within each image, the digits are uniform in size, thereby simulating the effect of a bird's eye view camera mounted on the drone. Over the whole data set, there are 20 different sizes of objects placed on the images, ranging from 28×28 px – the original image size for MNIST – to 112×112 px, an up scaling by a factor of four in each dimension. This approach is employed to emulate different flying altitudes, with larger digits on the images simulating lower flying altitudes and smaller digits representing higher altitudes. Figure 4.4 shows example images from this data set. In the following, we will call this data set *MNIST*

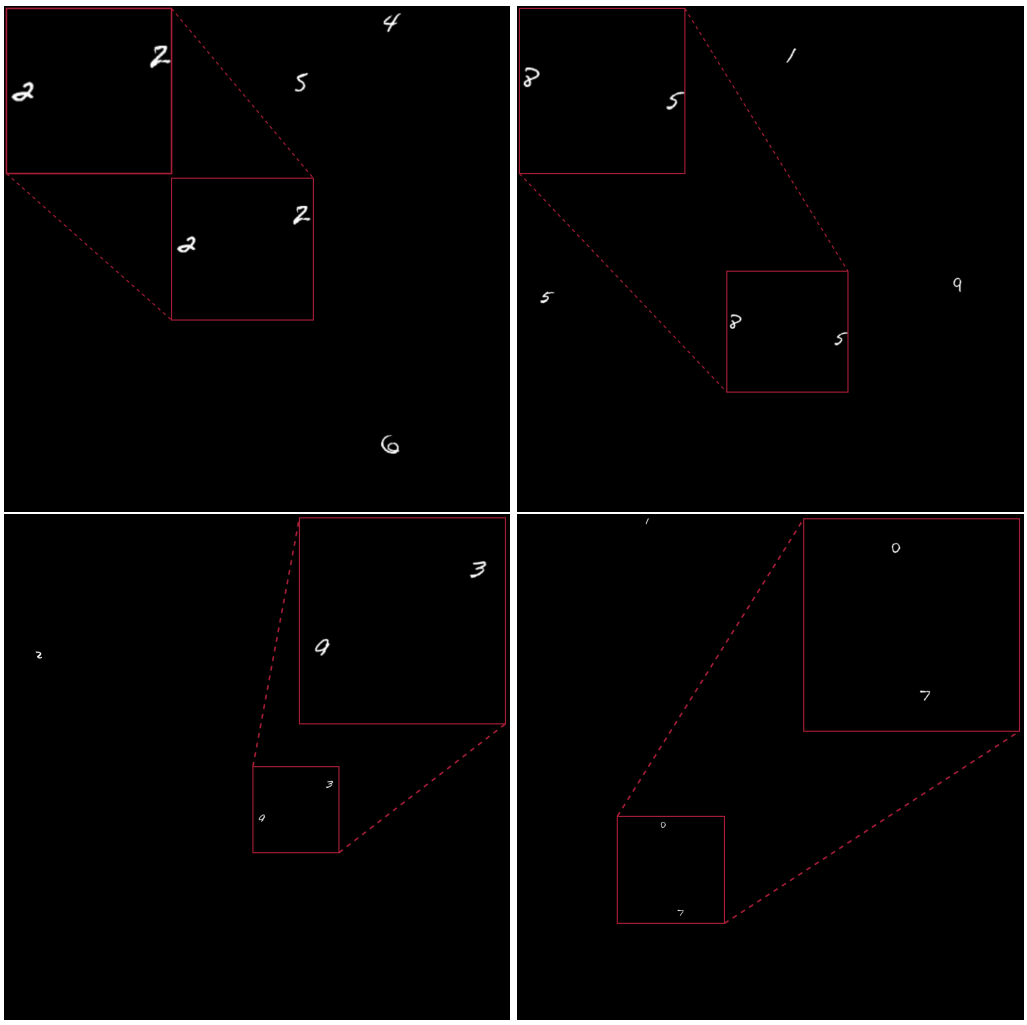


Figure 4.4: Example images from MNIST-Det. We can observe the different simulated capture-altitudes by the size of the digits in the image, where, for example, the top left image would be an image taken at a low altitude while the bottom right picture would be captured at a very high altitude. The marked areas are enlarged to get better impression of the visual data.

Detection (MNIST-Det). We observe, that the pictures are freed of the features and details present in real-world images that amplify the complexity of computer vision tasks such as object detection. Through this process, we generate an object detection data set encompassing ten classes – the digits '0' through '9' as it is in MNIST. We created this data set for the purpose of investigating adaptive resizing without complicating factors occurring in real-world images.

Table 4.1 shows the results for an EfficientDet–*D*0 on the whole MNIST-Det data set, with and without Adaptive Resizing. We observe, that the variant with Adaptive Resizing

essentially solves the task on this data set, while the vanilla detector struggles, trailing by roughly 27 points AP_{50}. We hypothesize that this is due to the high number of different scales present in the data set (20, as discussed above), which pose no problem to the Adaptive Resizer while overwhelming the limited scale-capacity of the standard object detection model.

Table 4.2 shows the performance of both models on the height transfer task on MNIST-Det. We can see, that the model without Adaptive Resizing virtually learns nothing. On the other hand, the EfficientDet models aided by the Adaptive Resizer achieve almost 90 points AP_{50} although being only trained on one of the 20 different scale appearances present in the data set. The decrease in performance compared to training on the whole data set might be due to two reasons; First, because we only train on $1/20$ of the data set. With a smaller training set the model has less data to extract information from, hence the expected decrease in performance. And second, because visual artifacts caused by the scaling of the images might hinder the detection further. Hinting at that is the higher number for $AP_{50}@112 \times 112$; we would expect less artifacts when shrinking large objects to a smaller size than the other way around.

	AP_{50}
EfficientDet–*D*0	70.1
*D*0+Adaptive	**97.6**

Table 4.1: Results in the AP_{50} metric on the validation set of the MNIST-Det data set of our baseline network EfficientDet–*D*0 with and without adaptive resizing.

	$AP_{50}@28 \times 28$	$AP_{50}@112 \times 112$
EfficientDet–*D*0	0.8	6.1
*D*0+Adaptive	**84.4**	**88.8**

Table 4.2: Validation AP_{50} results on the MNIST-Det data set when training on one size of objects only and then testing on the full validation set. The second column, $@28 \times 28$, denotes the results when training only on images with objects whose dimensions are 28×28 and analogously for $@112 \times 112$.

4.4 Experiments on Real Data

We employ Faster R-CNN (Ren *et al.* (2016)), CenterNet (Zhou *et al.* (2019)), and EfficientDet–*D*0 (Tan *et al.* (2020)) to test our approach. We chose these three to have experiments with representatives of multiple major classes of object detectors. The first is a well known two-stage detector which is highly adjustable, for example with different ResNet- (He *et al.* (2016)) or ResNeXt (Xie *et al.* (2017))-backbones. The latter two are well-known one-stage detectors. EfficientDet is an anchor-based object detector while CenterNet is an anchor-free object detector (Zhang *et al.* (2020a)).

In the following, we will always report AP_{50} values, as is usual for UAV data sets, except where explicitly stated otherwise.

4.4.1 Results on bird's eye view Portions

We conduct our experiments on two well-known UAV data sets, VisDrone (Zhu et. al. (2018)) and UAVDT (Yu et al. (2020)), and on the aforementioned People On Grass (POG) dataset, which is publicly available[2]. The two former consist of around 7k and 40k images, respectively, and were both captured in major Asian cities. The latter contains roughly 2.8k images, mostly showing people on a grass background. We captured POG to test our approach on because it features accurate height information per image, a very rare quality among UAV data sets. As mentioned earlier, we conduct our experiments on each data set's BEV portion. These subsets contain roughly 1.4k, 9.4k, and 1k images, respectively. Following the original authors of UAVDT, we combine all classes of their bounding box annotations into the single class 'car' for our experiments due to heavy class imbalances. Because the existing altitude annotations are too coarse for our purposes, we generate finer height data artificially for UAVDT and VisDrone. We do so using the second method from Section 4.2. More precisely, we generate the image sizes by Equation (4.2). For POG, we extract the log files from the UAV. Therefore, the data set contains meta annotations for each image, particularly altitude information, that is accurate to within one meter (Sitemark (2020)). The data set contains images in between $10\,m$ and $110\,m$.
The results in this section are generally lower than usually achieved on these benchmarks, compare to Chapter 3. This is due to the BEV portions being significantly smaller than the data sets (see e.g. Figure 3.2), which expectedly harms the detectors' performance. Additionally, the BEV portion is the most complex domain in UAV imagery, as discussed in Chapter 3. Among all possible viewing angles, the least recognizable visual features are present in the top-down view, see for example Figure 1.1.

For the experiments on one-stage detectors, we employ EfficientDet–$D0$ and CenterNet as described in their respective original papers (Tan *et al.* (2020); Zhou *et al.* (2019)). In the case of EfficientDet we fine-tuned hyper parameters like image size and anchor parameters (scales and ratios) to each data set. For CenterNet we did the same, except that it is anchor-free and therefore does not have anchor parameters. To test our approach, we also do experiments with both networks employing the Adaptive Resizer. We report the results of these experiments in Table 4.3 and 4.5. For both models we observe that employing Adaptive Resizer improves inference speed by a factor of two to three, see also Section 4.4.4. In the case of EfficientDet (Table 4.3) we can see that employing adaptive resizing achieves roughly an improvement of 3 points AP_{50} for VisDrone and POG. For UAVDT it even boosts performance by around 25 points AP_{50}. See below for a discussion of this large gap in performance increase. For CenterNet (Table 4.5) the

[2] https://cloud.cs.uni-tuebingen.de/index.php/s/yFztfJePREqj4om

models employing Adaptive Resizer consistently outperform their baseline counterparts. On UAVDT in the most extreme case even by 28 points AP_{50}. On VisDrone, however, the Adaptive Resizer only performs competitively with the baseline.

We also include results for the Adaptive Resizer on two-stage detectors. While these are not relevant for onboard processing, they are still the most capable object detection models. For UAVDT we employ the baseline from Wu *et al.* (2019) to compare with their approach, as they are also using meta-information like capture-altitude. It is a Faster R-CNN network with Resnet-101-FPN backbone (He *et al.* (2016)). For VisDrone, we reimplemented DE–FPN, which is the best-performing single model of the VisDrone Detection Challenge (Zhu *et al.* (2018)). We achieved 49.0 AP_{50} on the full validation set compared to their 49.1 AP_{50} on the full test set. To compare it to our model, we train and test it on the BEV portion, then employ this as the baseline (in both cases). From the results in Table 4.4 we observe that employing the Adaptive Resizer improves detection results for both data sets. While we improve by 5 AP_{50} points on VisDrone, we even achieve an improvement of over 13 AP_{70} on UAVDT compared to our baseline. We use the AP_{70} metric to compare our approach to Wu *et al.* (2019) and observe that our model outperforms theirs by around 4 points.

Summarizing all experiments, we observe that the Adaptive Resizer increases detection performance in general. However, the gain in performance is most prominent on UAVDT. We argue that this is due to the bad distribution of capture-altitudes in this data set. We observed that capture-altitudes are on average a lot lower in the training set of UAVDT than in its test set (which is not the case for VisDrone and POG). These are conditions the Adaptive Resizer can cope with very well, while generic object detectors' detection performance suffers greatly, see Section 4.4.5. Additionally, employing adaptive resizing speeds up inference by a factor of two to three on average.

	VisDrone	UAVDT	POG	FPS
*D*0 @ 2048	13.1	34.1	80.3	12
*D*0 @ 1792	17.7	30.0	74.3	15
*D*0 + Adaptive	**20.6**	**58.8**	**83.0**	**32**

Table 4.3: AP_{50} results on the bev portions of the data sets. EfficientDet–*D*0@*x* is a baseline model trained and evaluated such that the longer edge of each image is equal to *x*. All FPS values are benchmarked on UAVDT and an NVIDIA GeForce RTX 2080 Ti GPU.

4.4.2 Effects of Cutting the Feature Pyramid Network

In this section we compare the detector from Section 4.2.1, meant for fast inference and deployment to an embedded GPU, to a full-fledged EfficientDet–*D*0 model with Adaptive Resizer. They differ in the fact that the model from Section 4.2.1 has no feature

	UAVDT	VisDrone
Faster R–CNN	23.0	41.0
NDFT Wu *et al.* (2019)	32.9	–
Adaptive	**36.8**	**46.0**

Table 4.4: AP_{70} results on the BEV portions of UAVDT and AP_{50} on VisDrone. We use the AP_{70} metric to compare our approach with Wu *et al.* (2019).

	UAVDT	VisDrone	FPS
CN–RN18 Baseline	33.7	**23.7**	20
CN–RN18 Adaptive	**56.8**	22.1	**55**
CN–RN50 Baseline	35.4	**28.5**	8
CN–RN50 Adaptive	**63.4**	26.3	**23**
CN–RN101 Baseline	37.6	26.3	5
CN–RN101 Adaptive	**60.8**	**26.5**	**6**

Table 4.5: AP_{50} results and frames per second (FPS) of different CenterNet-models (Zhou *et al.* (2019)). They differ in their respective backbone, for example 'CN–RN101' is a CenterNet with a ResNet101 (He *et al.* (2016)) backbone.

pyramid network and therefore only uses one feature map. Table 4.6 provides empirical evidence for that measure. There, we can see the mean percentage of objects per image, that are detected by each feature map. Being detected by a certain feature map means, that the anchor which is selected to classify and regress the object in question (see description in Section 4.2.1 and Figure 4.5) is corresponding to this specific feature map. We discriminate between the detection percentage by feature map before and after applying non-maximum suppression (NMS). The values before NMS give an undistorted view of which feature maps would in principle be able to detect an object. The numbers after NMS, however, are more relevant to the application in practice, because only here does the detector filter predictions with poor scores; these are usually the ones which also regress the object worse than others. In Table 4.6 we can see that, after applying non-maximum suppression, on average less than two percent of all objects per image are not detected by the first feature map.

4.4.3 Results on the complete UAVDT data set

To also introduce a model that works on a full UAV data set, we use a multi-domain approach in the style of Chapter 3. More explicitly, we use the meta-data supplied by the UAV to distinguish between bird's eye view images and non-bird's eye view. During inference, we use the Adaptive Resizer model on the BEV images and a baseline model

	1	2	3	4 & 5
pre NMS	92.03 %	7.95 %	0.03 %	0.00 %
post NMS	98.01 %	1.97 %	0.02 %	0.00 %

Table 4.6: Average number of objects that are detected by each feature map before and after applying non-maximum suppresion (NMS). The average is taken over UAVDT. The investigated model is an EfficientDet–*D*0 with feature pyramid network and Adaptive Resizer.

	UAVDT	VisDrone	POG	FPS
*D*0–noFPN	49.3	**23.6**	79.2	**56**
*D*0–FPN	**58.8**	20.6	**83.0**	32

Table 4.7: AP_{50} results on UAVDT, VisDrone, and POG. The compared models are EfficientDet–*D*0 with Adaptive Resizer. *D*0–noFPN is a model without FPN like described in Section 4.2.1, *D*0–FPN is the standard model with Adaptive Resizer, including feature pyramid network.

on all other images. Both are loaded before inference and available in GPU memory, so there is little overhead added and no drop in inference time for each of the models. To achieve the results reported in Table 4.8 on UAVDT we use the models from Table 4.4 for the two-stage detector experiments. For the experiments with EfficientDet–*D*0 we use the model without FPN from Section 4.2.1 and the baseline from Table 4.3.

We observe that both models improve by circa 3 AP points. Note that we are en par with Perreault et. al. (2020), also achieving 52.8 AP_{70}. They give, to the best of our knowledge, the state-of-the-art detector on UAVDT. However, they employ a vastly more complicated method which needs short video sequences to perform well.

	Faster R–CNN	*D*0
Baseline	49.4	34.6
SpotNet (Perreault et. al. (2020))	**52.8**	–
Adaptive	**52.8**	**37.7**

Table 4.8: Results on the full UAVDT data set. We use the AP_{70} metric to compare our approach with the reported numbers in Perreault et. al. (2020).

4.4.4 Time benchmarks

Tables 4.3 and 4.5 show that the Adaptive Resizer makes a model two to three times faster than its respective baseline. The reported number is the average of the inference times over the UAVDT BEV data set. We take the mean because the inference time for

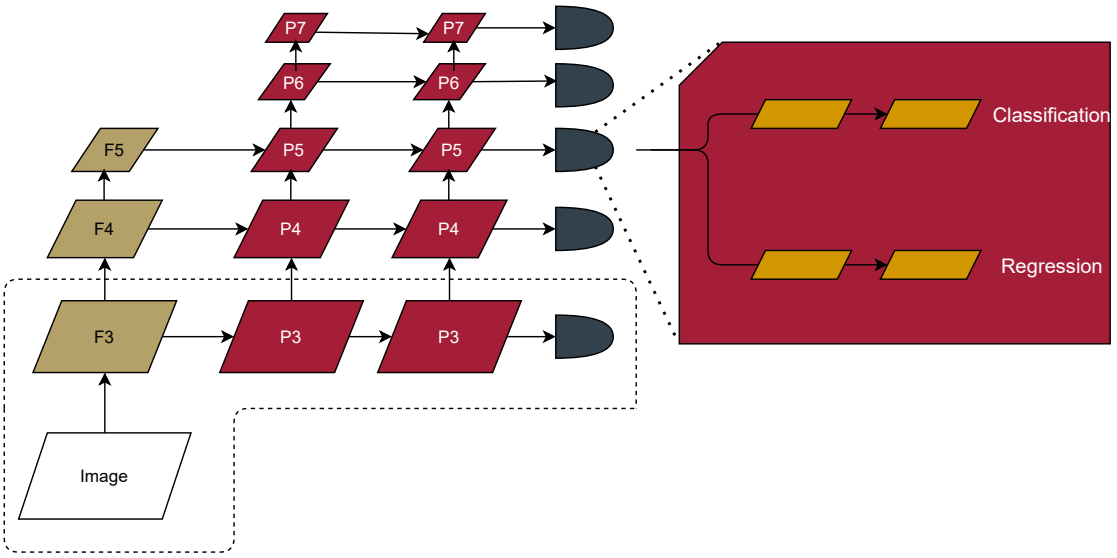


Figure 4.5: Schematic drawing of an EfficientDet–$D0$. Starting from the image, the backbone-network extracts feature maps $F3, F4, F5$ (gold). Then these are input to the feature pyramid network $P3 - P7$ (red) and afterwards handed to the heads (anthracite). These perform classification and regression. The object detector without FPN from Section 4.2.1 is encircled with the dashed line.

an Adaptive Resizer model is not constant; like for every object detector the inference time is dependant on the image size, which in turn is dependant on the capture-altitude. We chose UAVDT to average over because it is the largest of the data sets we tested on. Therefore, it is the least prone to statistical outliers during the benchmark test. Table 4.3 and 4.7 show, where the speed improvement of the EfficientDet–$D0$+Adaptive Resizer without FPN comes from. Cutting the FPN from the model brings an improvement of 24 FPS, which is larger than expected. We argue that this is due to the convolutional layers in the FPN, especially in later layers, having higher channel-dimensions than earlier layers (Tan *et al.* (2020)). Because convolutional networks are essentially fully-connected in the channel-dimension, cutting these brings the largest speed improvement.

Figure 4.3 explains the speed improvement when using Adaptive Resizer without any other alterations. All images captured at low altitudes are resized to comparably small image sizes, speeding the network up a lot, while the baseline runs at constant speed. One could argue that the speed comparison is not fair because the baseline is employing a larger image size. However, this is necessary; otherwise, the baseline's AP deteriorates (as we saw in experiments) because of the small objects in UAV data sets (Singh and Davis (2018); Varga and Zell (2021); Unel et. al. (2019)).

We also benchmarked our EfficientDet–$D0$ with Adaptive Resizer and without FPN on a Jetson AGX Xavier development board optimized with `TensorRT` and half-precision FP16. There, our model achieved roughly 16 FPS averaged over UAVDT. Meanwhile,

the baseline achieved 7 and 5 FPS when resizing the image's longer respective side to 1792 and 2048 px, respectively, as in Table 4.3. Therefore, on embedded hardware, the Adaptive Resizer improves inference speed by a factor of two to three.

4.4.5 Height Transfer

To prove the claims made in section 4.2 about the transfer learning capabilities of our Adaptive Resizing method on real-world imagery, we consider four different data set splits for our experiments on height transfer. The construction of these splits is as follows: starting from the above described BEV subsets, we order the images in the data set by their respective capture-altitude. We then use the 25 % images with the highest capture-altitude from the training set of the BEV portion as the training set for this task. For the validation set, we use all of the validation images from the BEV subset. Together we call this ABOVE75. Repeating this procedure for the bottom 25 %, bottom and top 50 % of the training images yields BELOW25, BELOW50, and ABOVE50, respectively. Note that the validation set for each of these splits is the entire validation set of the BEV portion, including all capture-altitudes.

Constructing the data set split this way makes this experiment fit to verify the above claims; if a model performs well on one of the above data set splits, it means that it can generalize from the images it trained on to images with capture-altitudes it never saw before.

Table 4.9 shows that the Adaptive Resizer models consistently outperform their respective baseline counterparts in these experiments. The reported numbers on VisDrone are generally relatively low, as expected, due to the size of the training sets, e.g. BELOW25 and BELOW75 contain ≈ 300 training images. Still, in the best case, Adaptive Resizer is three times as good as its baseline (4.9 vs 14.2 AP_{50}).

To explain the large improvement in the case of UAVDT, we assume that the baseline's improvement compared to Table 4.3 comes from UAVDT's gap in between training and validation images we already discussed. We perceive many more images captured at very high altitudes in its validation set, which do not appear in the training set. The Adaptive Resizer can handle this gap, being essentially en par with its performance on the whole BEV split of UAVDT, e.g. 47.9 versus 49.3 AP_{50} (see Table 4.7).

4.5 Conclusion

In this chapter, we proposed a novel preprocessing step. It adjusts the image size according to the height in which the image was captured, solving the scale variance problem in BEV imagery. This method significantly improves detection performance over multiple data sets and object detectors while also improving inference speed, making it applicable to near real-time object detection on mobile platforms. We also showed that this method enables object detectors to generalize well to images captured in heights they have never

	VisDrone		UAVDT	
	$D0$	$D0+$Adapt.	$D0$	$D0+$Adapt.
BELOW25	5.0	**7.2**	9.7	**47.9**
BELOW50	7.0	**12.0**	26.1	**45.5**
ABOVE50	4.9	**14.2**	32.1	**45.4**
ABOVE75	8.0	**11.2**	18.7	**44.5**

Table 4.9: Empirical results for height transfer on VisDrone and UAVDT. Each cell reports the AP_{50} result of either the baseline or adaptive resizer version of an EfficientDet–$D0$.

seen before. Furthermore, we used an expert-model approach as was introduced in Chapter 3 to capitalize on our method on generic UAV imagery.
Since the last two chapters highlighted the importance of meta-data in UAV object detection and to alleviate the lack of maritime data sets in the object detection community, the next chapter will introduce a large-scale maritime object detection data set.

Chapter 5

A Maritime Benchmark for Detecting Humans in Open Water

5.1 Introduction

Currently, the most effective algorithms for addressing the challenge of UAV object detection, which is crucial for achieving effective mSAR capabilities, are implemented using data-driven methods such as deep neural networks. Chapters 3 and 4 provide numerous examples of these methods in action. These methods depend on large-scale data sets portraying real-case scenarios to obtain realistic imagery statistics. However, there is a great lack of large-scale data sets in maritime environments. Most data sets captured from UAVs are land-based, often focusing on traffic environments, such as VisDrone (Zhu et. al. (2018)) and UAVDT (Du *et al.* (2018)). Many of the few data sets that are captured in maritime environments fall in the category of remote sensing, often leveraging satellite-based synthetic aperture radar (Crisp (2004)). All of these are only valuable for ship detection (Corbane *et al.* (2010)) as they don't provide the resolution needed for SAR missions. Furthermore, satellite-based imagery is susceptible to clouds and only provides top-down views. Finally, many current approaches in the maritime setting rely on classical machine learning methods, incapable of dealing with the large number of influencing variables and calling for more elaborate models (Prasad *et al.* (2019)).

This chapter aims to close the gap between large-scale land-based data sets captured from UAVs to maritime-based data sets. We introduce a large-scale data set of people in open water, called SeaDronesSee. We captured videos and images of swimming probands in open water with various UAVs and cameras. As it is especially critical in SAR missions to detect and track objects from a large distance, we captured the RGB footage with 3840×2160 px to 5456×3632 px resolution. We carefully annotated ground-truth bounding box labels for objects of interest including swimmer, floater (swimmer with life jacket), life jacket, swimmer† (person on boat not wearing a life jacket), floater† (person on boat wearing a life jacket), and boat.

Moreover, we note that current data sets captured from UAVs only provide very coarse or no meta information at all. We argue that this is a major impediment in the develop-

© 2024 IEEE. Reprinted, with permission, from Varga *et al.* (2022).

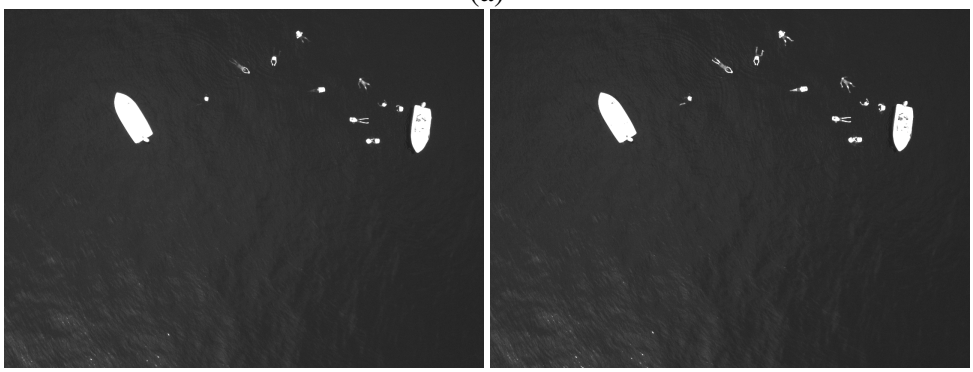
ment of multi-modal systems, which take these additional information into account to improve accuracy or speed. Recently, methods that rely on these meta data were proposed. However, they note the lack of large-scaled publicly available data set in that regime (see e.g. Wu *et al.* (2019) or Chapters 3 and 4 in this dissertation). Therefore, we provide precise meta information for every frame and image including altitude, camera angle, speed, time, and others.

In maritime settings, the use of multi-spectral cameras with Near Infrared channels to detect humans can be advantageous (Gallego *et al.* (2019)). For that reason, we also captured multi-spectral images using a MicaSense RedEdge. This enables the development of detectors taking into account the non-visible light spectra near infrared (842 nm) and Red Edge (717 nm).

Finally, we provide detailed statistics of the data set and conduct extensive experiments using state-of-the-art models and hereby establish baseline models. These serve as a starting point for our SeaDronesSee benchmark. We release the training and validation sets with complete bounding box ground truth but only the test set's videos/images. The ground truth of the test set is used by the benchmark server to calculate the generalization power of the models. We set up an evaluation web page, where researchers can upload their predictions and opt to publish their results on a central leader board, such that transparent comparisons are possible. The benchmark focuses on three tasks: (i) object detection, (ii) single-object tracking and (iii) multi-object tracking, which will be explained in more detail in the subsequent sections. This chapter's main contributions are as follows:

- SeaDronesSee is the first large annotated UAV-based data set of swimmers in open water. It can be used to further develop detectors and trackers for SAR missions.
- We provide full environmental meta information for every frame making SeaDronesSee the first UAV-based data set of that nature.
- We provide an evaluation server to prevent researchers from overfitting and allow for fair comparisons.
- We perform extensive experiments on state-of-the-art object detectors and trackers on our data set.

(a)



(b)

Figure 5.1: (a) Typical image examples with varying altitudes and angles of view: 250 m, 90°; 50 m, 30°; 10 m, 0° and 20 m, 90° (from top left to bottom right). (b) Examples of the Red Edge (717 nm, left) and Near Infrared (842 nm, right) light spectra of an image captured by the MicaSense RedEdge-MX. Note the glowing appearance of the swimmers.

Object detection	Env.	Platform	Image widths	Altitude	Range	Angle	Range	Other meta
DOTA (Xia *et al.* (2018))	cities	satellite	$800-20,000$	–	–	×	90°	×
UAVDT (Du *et al.* (2018))	traffic	UAV	1,024	×	5 – 200 m*	×	0 – 90°*	×
VisDrone (Zhu et. al. (2018))	traffic	UAV	$960-2,000$	×	5 – 200 m*	×	0 – 90°*	×
Airbus Ship (Airbus (2018))	maritime	satellite	768	–	–	×	90°	×
AU-AIR (Bozcan and Kayacan (2020))	traffic	UAV	1,920	✓	5 – 30 m	×	45 – 90°	✓
SeaDronesSee	maritime	UAV	$3,840-5,456$	✓	5 – 260 m	✓	0 – 90°	✓
Single-object tracking	Env.	#Clips	Frame widths	Altitude	Range	Angle	Range	Other meta
UAV123 (Mueller *et al.* (2016))	traffic	123	1,280	×	5 – 50 m*	×	0 – 90°*	✓
DTB70 (Li and Yeung (2017))	sports	70	1,280	×	0 – 10 m*	×	0 – 90°*	×
UAVDT-SOT (Du *et al.* (2018))	traffic	50	1,024	×	5 – 200 m*	×	0 – 90°*	✓
VisDrone (Zhu et. al. (2018))	traffic	167	$960-2,000$	×	5 – 200 m*	×	0 – 90°*	✓
SeaDronesSee	maritime	208	3,840	✓	5 – 150 m	✓	0 – 90°	✓
Multi-object tracking	Env.	#Frames	Frame widths	Altitude	Range	Angle	Range	Other meta
UAVDT-MOT (Du *et al.* (2018))	traffic	40.7 k	1,024	×	5 – 200 m*	×	0 – 90°*	✓
VisDrone (Zhu et. al. (2018))	traffic	40 k	$960-2,000$	×	5 – 200 m*	×	0 – 90°*	✓
SeaDronesSee	maritime	54 k	3,840	✓	5 – 150 m	✓	0 – 90°	✓

Table 5.1: Comparison of SeaDronesSee with the most prominent annotated aerial data sets. 'Altitude' and 'Angle' indicate whether or not there are precise altitude and angle view information available. 'Other meta' refers to time stamps, GPS, and IMU data and in the case of object tracking can also mean attribute information about the sequences. The values with stars have been estimated based on ground truth bounding box sizes and corresponding real world object sizes (for altitude) and qualitative estimation of sample images (for angle). For DOTA and Airbus Ship the range of altitudes is not available because these are satellite-based data sets, hence the '–'-sign.

In the following, we review major labeled data sets in the field of computer vision from UAVs and in maritime scenarios which are usable for supervised learning models.

Over the last few years, quite a few data sets captured from UAVs have been published. The most prominent are these that depict traffic situations, such as VisDrone (Zhu et. al. (2018)) and UAVDT (Du *et al.* (2018)). Both data sets focus on object detection and object tracking in unconstrained environments. In Pei *et al.* (2019), the authors collect videos (Stanford Drone Dataset) showing traffic participants on campuses (mostly people) for human trajectory prediction usable for object detection. UAV123 (Mueller *et al.* (2016)) is a single-object tracking data set consisting of 123 video sequences with corresponding labels. The clips mainly show traffic scenarios and common objects. In Hsieh *et al.* (2017) and also Mundhenk *et al.* (2016), the authors capture a data set showing parking lots for car counting tasks and constrained object detection. The authors of Li and Yeung (2017) provide a single-object tracking data set showing traffic, wild life and sports scenarios. Collins et al. capture a single-object tracking data set showing vehicles on streets in rural areas. In Krajewski *et al.* (2018), the authors show vehicles on freeways.

Another active area of research focuses on drone-based wildlife detection. The work van Gemert *et al.* (2014) releases a data set for the tasks of low-altitude detection and counting of cattle. The authors of Ofli *et al.* (2016) release the African Savanna data set as part of their crowd-sourced disaster response project.

Many data sets in maritime environments are captured from satellite-based synthetic aperture radar and therefore fall into the remote sensing category. In this category, the airbus ship data set (Airbus (2018)) is prominent, featuring 40k images from synthetic aperture radars with instance segmentation labels. The paper Li *et al.* (2018) provides a data set of ships with images mainly taken from Google Earth, but also a few UAV-based images. In Xia *et al.* (2018), the authors provide satellite-based images from natural scenes, mainly land-based but also harbors. The most similar to our work is Lygouras *et al.* (2019). They also consider the problem of human detection in open water. However, their data mostly contains images close to shores and of swimming pools. Furthermore, it is not publicly available.

UAVDT (Du *et al.* (2018)) provides coarse meta data for their object detection and tracking data: every frame is labeled with altitude information (low, medium, high), angle of view (front-view, side-view, bird-view) and light conditions (day, night, foggy). Some additional manual work is done in Wu *et al.* (2019), where the authors manually label VisDrone after its release with the same annotation information for the object detection track. Mid-Air (Fonder and Van Droogenbroeck (2019)) is a synthetic multi-modal data set with images in nature containing precise altitude, GPS, time, and velocity data but without annotated objects. Blackbird Antonini *et al.* (2018) is a real-data indoor data set for agile perception also featuring these meta information. In Majdik *et al.* (2017), street-view images with the same meta data are captured to benchmark appearance-based localization. The paper Bozcan and Kayacan (2020) released a low-altitude ($< 30\ m$) object detection data set containing images showing a traffic circle and provide meta data

Camera	Resolution	Video
Hasselblad L1D-20c	3,840×2,160	30 FPS
MicaSense RedEdge-MX	1,280× 960	×
Sony UMC-R10C	5,456×3,632	×
Zenmuse X5	3,840×2,160	30 FPS
Zenmuse XT2	3,840×2,160	30 FPS

Table 5.2: Overview of employed cameras.

such as altitude, GPS, and velocity but exclude the import camera angle information. As the angle is mostly very acute ($< 45\ ^\circ$), no information about object's sizes can be inferred from the provided meta-data.

Tracking data sets often provide meta data (or attribute information) for the clips. However, in many cases these do not refer to the environmental state in which the image was captured. Instead, they abstractly describe the way in which a clip was captured: UAV123 (Mueller *et al.* (2016)) label their clips with information such as aspect ratio change, background clutter, and fast motion, but do not provide frame-by-frame meta data. The same observation can be made for the tracking track of VisDrone (Fan *et al.* (2020b)). See Table 5.1 for an overview of annotated aerial data sets.

5.2 Data Set Generation

We gathered the footage on several days to obtain variance in light conditions. Taking into account safety and environmental regulations, we asked over 20 test subjects to be recorded in open water. Boats transported the subjects to the area of interest, where quadcopters were launched at a safe distance from the swimmers. At the same time, the fixed-wing UAV Trinity F90+ was launched from the shore. We used waypoints to ensure a strict flight schedule to maximize data collection efficiency. Care was taken to maintain a strict vertical separation of the UAVs at all times. Subjects were free to wear life jackets, of which we provided several differently colored pieces (see also Figure 5.2).

To diminish the effect of camera biases within the data set, we used multiple cameras, as listed in Table 5.2, mounted to the following drones: DJI Matrice 100, DJI Matrice 210, DJI Mavic 2 Pro, and a Quantum Systems Trinity F90+. With the video cameras, we captured videos at 30 FPS. For the object detection task, we extracted at most three frames per second of these videos to avoid having redundant occurrences of frames. See Section 5.3 for information on the distribution of images with respect to different cameras.

Lastly, we captured top-down looking multi-spectral imagery at 1 FPS. We used a MicaSense RedEdge-MX, which records five wavelengths (475 nm, 560 nm, 668 nm, 717 nm, 842 nm). Therefore, in addition to the RGB channels, the recordings also contain a

Data	Unit	Min. value	Max.value
Time since start	ms	0	∞
Date and Time	ISO 8601	–	–
Latitude	degrees	-90	$+90$
Longitude	degrees	-90	$+90$
Altitude	meters	0	∞
Gimbal pitch	degrees	0	90
UAV roll	degrees	-90	$+90$
UAV pitch	degrees	-90	$+90$
UAV yaw	degrees	-180	$+180$
x-axis speed	m/s	0	∞
y-axis speed	m/s	0	∞
z-axis speed	m/s	0	∞

Table 5.3: Meta data that comes with every image/frame.

RedEdge and a Near Infrared channel. The camera was referenced with a white reference before each flight. As the RedEdge-MX captures every band individually, we merge the bands using the development kit provided by MicaSense.

5.2.1 Meta Data Collection

Accompanied with every frame there is a meta stamp, that is logged at 10 hertz. To align the video data (30 FPS) and the time stamps, a nearest neighbor method was performed. The data in Table 5.3 was logged and provided for every image/frame read from the onboard clock, barometer, IMU and GPS sensor, and the gimbal, respectively.

Note that $\alpha = 90^\circ$ corresponds to a top-down view, and $\alpha = 0^\circ$ to a horizontally facing camera. The date format is given in the extended form of ISO 8601. Furthermore, note that the UAV roll/pitch/yaw-angles are of minor importance for meta-data-aware vision-based methods as the onboard gimbal filters out movement by the drone such that the camera pitch angle is roughly constant if it is not intentionally changed (Jedrasiak *et al.* (2013)). Note that the gimbal yaw angle is not included, as we fix it to coincide with the UAV's yaw angle.

We need to emphasize that the meta values lie within the error thresholds introduced by the different sensors, but an extended analysis is beyond the scope of this dissertation (see e.g. Zimmermann *et al.* (2017); Sitemark (2020); Kulhavy *et al.* (2017) for an overview).

5.2.2 Annotation Method

Using the non-commercial labeling tool DarkLabel (DarkLabel (2020)), we manually and carefully annotated all provided images and frames with the categories swimmer (person in water without life jacket), floater (person in water with life jacket), life jacket, swimmer† (person on boat without life jacket), floater† (person on boat with life jacket), and boats. We note that it is not sufficient to infer the class floater by the location from swimmer and life jacket as this can be highly ambiguous. Subsequently, all annotations were checked by experts in aerial vision. We choose these classes as they are the hardest and most critical to detect in SAR missions. Furthermore, we annotated regions with other objects as ignored regions, such as boats on land. Moreover, the data set also covers unlabeled objects, which may not be of interest, like driftwood, birds or the coast such that detectors can be robust to distinguish from those objects. See Figure 5.2 for examples of objects.

5.2.3 Data Set Split

Object Detection

To ensure that the training, validation, and testing set have similar statistics, we roughly balance them such that the respective subsets have similar distributions with respect to altitude and angle of view, two of the most important factors of appearance changes. Of the individual images, we randomly select $4/7$ and add it to the training set, add $1/7$ to the validation set and another $2/7$ to the testing set. In addition to the individual images, we randomly cut every video into three parts of length $4/7$, $1/7$, and $2/7$ of the original length and add every 10-th frame of the respective parts to the training, validation, and testing set. This is done to avoid having subsequent frames in the training and testing set such that a realistic evaluation is possible. We release the training and validation set with all annotations and the testing set's images, but withhold its annotations. Evaluation is available via an evaluation server[1], where the predictions on the test set can be uploaded.

Object Tracking

Similarly, we take $4/7$ of our recorded clips as the training clips, $1/7$ as the validation clips and $2/7$ as the testing clips. As for the object detection task, we withhold the annotations for the testing set and provide an evaluation server[1].

[1] https://seadronessee.cs.uni-tuebingen.de/

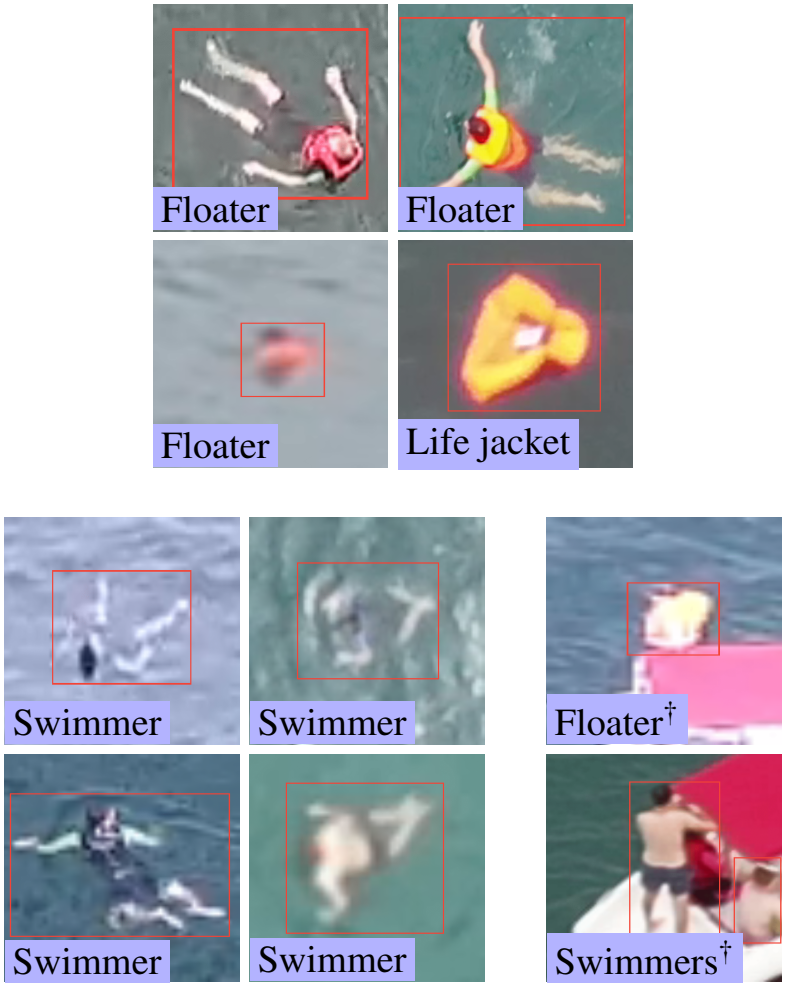


Figure 5.2: Examples of objects. Note that these examples are crops from high-resolution images. However, as the objects are small and the images taken from high altitudes, they appear blurry.

5.3 Data Set Tasks

There are many works on UAV-based maritime SAR missions, focusing on unified frameworks describing the process of how to search and rescue people, for example Mishra *et al.* (2020); Gallego *et al.* (2019); Lvsouras and Gasteratos (2020); Lygouras *et al.* (2019); Queralta *et al.* (2020); Roberts *et al.* (2016); Ghazali *et al.* (2016). These works answer questions corresponding to path planning, autonomous navigation and efficient signal transmission. Most of them rely on RGB sensors and detection and tracking algorithms to actually find people of interest. This commonality motivates us to extract the specific tasks of object detection and tracking, which pose some of the most challenging issues in this application scenario.

Maritime environments from a UAV's perspective are difficult for a variety of reasons: Reflective regions and shadows resulting from different cardinal points (such as in Fig. 5.1) that could lead to false positives or negatives; people may be hardly visible or occluded by waves or sea foam (see Supplementary material); typically large areas are overseen such that objects are particularly small (Mishra *et al.* (2020)). We note that these factors are on top of general UAV-related detection difficulties.

Now, we proceed to describe the specific tasks.

5.3.1 Object Detection

There are 5,630 images (training: 2,975; validation: 859; testing: 1,796). See Figure 5.3 for the distribution of images/frames with respect to cameras and the class distribution. We recorded most of the images with the L1D-20c and UMC-R10C, having the highest resolution. Having the lowest resolution, we recorded only 432 images with the RedEdge-MX. Note, for the Object Detection Task only the RGB-channels of the multi-spectral images are used to support a uniform data structure.

Furthermore, the class distribution is slightly skewed towards the class 'boat', since safety precautions require boats to be nearby. We emphasize that this bias can easily be diminished by blackening the respective regions, as is common for areas which are not of interest or undesired (such as boats here; see e.g. Du *et al.* (2018)). Right after that, swimmers with life jacket are the most common objects. We argue that this scenario is very often encountered in SAR missions. This type of class is often easier to detect than just swimmer as life jackets mostly are of contrasting color, such as red or orange (see Fig. 5.2 as well as Tables 5.4 and 5.5). However, as it is also a likely scenario to search for swimmers without life jacket, we included a considerable amount. There are also several different manifestations/visual appearances of that class, which is why we recorded and annotated swimmers with and without adequate swimwear (such as wet suit). To be able to discriminate between humans in water and humans on boats, we also annotated humans on boats (with and without life jackets). Lastly, we annotated a small amount of life jackets only. However, we note that the discrimination between life jackets and humans in life jackets can become visually ambiguous, especially at higher

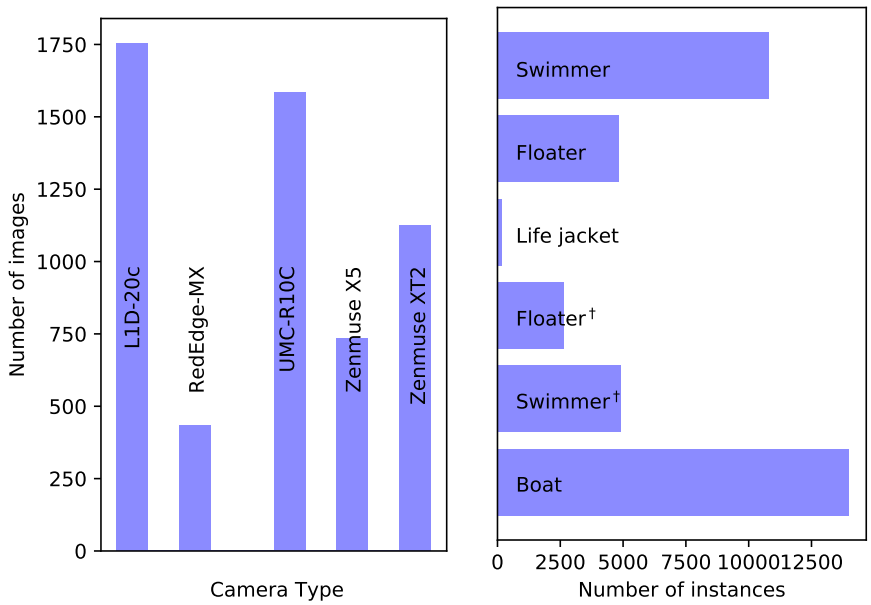


Figure 5.3: Distribution of training images over camera types (left) and distribution of objects over classes (right). As mentioned in section 5.2.2, a † symbol denotes a corresponding human on a boat, outside the water, as opposed to instances inside the water.

altitudes. See also Fig. 5.2.

Figure 5.4 shows the distribution of images with respect to the altitude and viewing angle they were captured at. Roughly 50% of the images were recorded below 50 m because lower altitudes allow for the whole range of available viewing angles ($0-90°$). That is, to cover all viewing angles, more images at these altitudes had to be taken. On the other hand, there are many images facing downwards (90°), because images taken at greater altitudes tend to face downwards since acute angles yield image areas with tiny pixel density, which is unsuitable for object detection. Nevertheless, every altitude and angle interval is sufficiently represented.

5.3.2 Single-Object Tracking

We provide 208 short clips (>4 seconds) with a total of 393,295 frames (counting the duplicates), including all available objects labeled. We randomly split the sequences into 58 training, 70 validation and 80 testing sequences. We do not support long-term tracking. The altitude and angle distributions are similar to these in the object detection section since the origin of the images of the object detection task is the same.

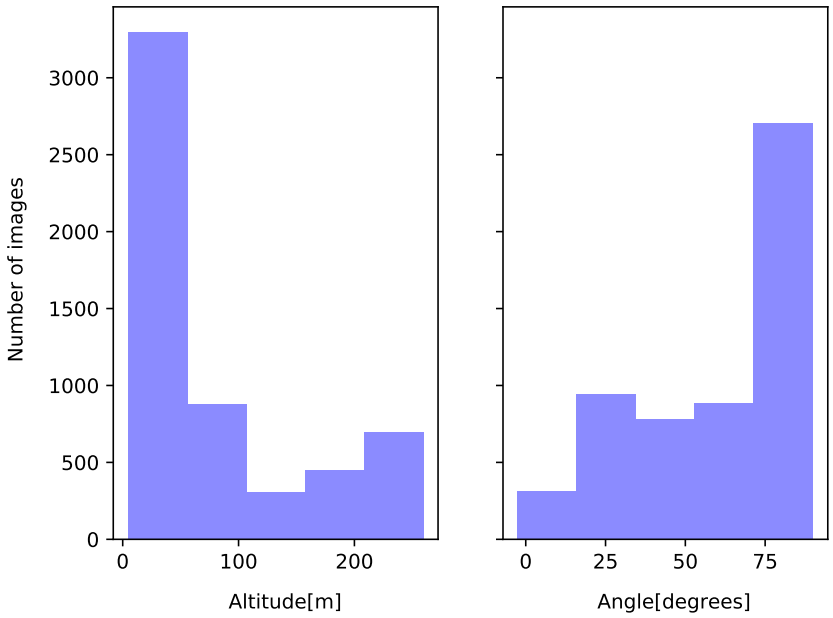


Figure 5.4: Distribution of images over altitudes (left) and angles (right), respectively.

5.3.3 Multi-Object Tracking

We provide 22 clips with a total of 54,105 frames and 403,192 annotated instances, the average consists of 2,460 frames. We differentiate between two use-cases. In the first task, only the persons in water (floaters and swimmers) are tracked, it is called MOT-Swimmer. In the second task, all objects in water are tracked (also the boats, but not people on boats), called MOT-All-Objects-In-Water. In both tasks, all objects are grouped into one class. The data set split is performed as described in section 5.2.3.

5.3.4 Multi-Spectral Footage

Along with the data for the three tasks, we provide multi-spectral images. We supply annotations for all channels of these recordings, but only the RGB-channels are currently part of the Object Detection Task. There are 432 images with 1,901 instances. See Figure 5.1 for an example of the individual bands.

5.4 Evaluations

We evaluated current state-of-the-art object detectors and object trackers on SeaDrones-See. All experiments can be reproduced by using our provided code available on the evaluation server.

5.4.1 Object Detection

The detectors used can be split into two groups. The first group consists of two-stage detectors, which are mainly built on Faster R-CNN (Girshick (2015)) and its improvements. Built for optimal accuracy, these models often lack the inference speed needed for real-time employment, especially on embedded hardware, which can be a vital use-case in UAV-based SAR missions. For that reason, we also evaluate on one-stage detectors. In particular, we perform experiments with the best performing single-model (no ensemble) from the workshop report Zhu *et al.* (2018): a Faster R-CNN with a ResNeXt-101 64-4d (Xie *et al.* (2017)) backbone with P6 removed. For large one-stage detectors, we take the recent CenterNet (Zhou *et al.* (2019)). To further test an object detector in real-time scenarios, we choose the current best model family on the COCO test-dev according to Papers with Code (2021), i.e. EfficientDet (Tan *et al.* (2020)), and take the smallest model, *D*0, which can run in real-time on embedded hardware, such as the Nvidia Xavier (see e.g. Chapter 3).

Model	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	FPS
F. ResNeXt-101-FPN Xie *et al.* (2017)	30.4	54.7	29.7	18.6	42.6	2
F. ResNet-50-FPN Girshick (2015)	14.2	30.1	7.2	6.4	17.7	14
CenterNet-Hourglass104 Zhou *et al.* (2019)	25.6	50.3	22.2	17.7	40.1	6
CenterNet-ResNet101 Zhou *et al.* (2019)	15.1	36.4	10.8	9.6	21.4	22
CenterNet-ResNet18 Zhou *et al.* (2019)	9.9	21.8	9.0	7.2	19.7	78
EfficientDet–*D*0 Tan *et al.* (2020)	20.8	37.1	20.6	11.5	29.1	26

Table 5.4: Average precision results for several baseline models. All reported FPS numbers are obtained on a single NVIDIA RTX 2080 Ti. The abbreviation 'F.' stands for Faster R-CNN.

Similar to the VisDrone benchmark (Zhu et. al. (2018)), we evaluate detectors according to the COCO json-format (Lin *et al.* (2014)), i.e. average precision at certain intersection-over-union-thresholds. More specifically, we use $AP = AP^{IoU=0.5:0.05:0.95}$, $AP_{50} = AP^{IoU=0.5}$ and $AP_{75} = AP^{IoU=0.75}$. Furthermore, we evaluate the maximum recalls for at most 1 and 10 given detections, respectively, denoted $AR_1 = AR^{max=1}$, and $AR_{10} = AR^{max=10}$. All these metrics are averaged over all categories (except for "ignored region"). We furthermore provide the class-wise average precisions. Moreover, similar to Chapter 3, we report AP_{50}-results on different equidistant levels of altitudes 'low' = 5-

Model	S	F	S†	F†	B	LJ	FPS
F. ResNeXt-101-FPN Xie *et al.* (2017)	78.1	82.4	25.9	44.3	96.7	0.6	2
F. ResNet-50-FPN Girshick (2015)	24.6	54.1	4.9	7.5	89.2	0.3	14
CenterNet-Hourglass104 Zhou *et al.* (2019)	65.1	73.6	19.1	48.1	95.8	0.3	6
CenterNet-ResNet101 Zhou *et al.* (2019)	16.8	39.8	0.8	1.7	74.3	0	22
CenterNet-ResNet18 Zhou *et al.* (2019)	20.9	21.9	2.6	3.3	81.9	0.4	78
EfficientDet–*D*0 Tan *et al.* (2020)	65.3	55.1	3.1	3.3	95.5	0.1	26

Table 5.5: AP_{50}–results for each class individually. All reported FPS numbers are obtained on a single NVIDIA RTX 2080 Ti. The abbreviation 'F.' stands for Faster R-CNN. For visualization purposes, the classes are abbreviated as swimmer(†) $\rightarrow$ S(†), floater(†) $\rightarrow$ F(†), boat $\rightarrow$ B, life jacket $\rightarrow$ LJ.

56 m (L), 'low-medium' = 55-106 m (LM), 'medium' = 106-157 m (M), 'medium-high' = 157-208 m (MH), and 'high' = 208-259 m (H). To measure the universal cross-domain performance, we report the average over these domains, denoted AP_{50}^{avg}. Similarly, we report AP_{50}-results for different angles of view: 'acute' = 7-23° (A), 'acute-medium' = 23-40° (AM), 'medium' = 40-56° (M), 'medium-right' = 56-73° (MR), and 'right' = 73-90° (R). Ultimately, it is the goal to have robust detectors across all domains uniformly, which is better measured by the latter metrics.

Table 5.4 shows the results for all object detection models. As expected, the large Faster R-CNN with ResNeXt-101 64-4d backbone performs best, closely followed by CenterNet-Hourglass104. Medium-sized networks, such as the ResNet-50-FPN, and fast networks, such as CenterNet-ResNet18 and EfficientDet-*D*0, expectedly perform worse. However, the latter can run in real-time on an Nvidia Xavier (compare with Chapter 3). Table 5.5 displays the detection results (in the AP_{50} metric) individually for each class. Swimmers are detected significantly worse than floaters by most detectors. Notably, life jackets are very hard to detect since from a far distance these are easily confused with swimmers† (see Fig. 5.2). Since there is a heavy class imbalance with many fewer life jackets, detectors are biased towards floaters.

Table 5.6 and 5.7 show the performances for different altitudes and angles, respectively. These evaluations help assess the strength and weaknesses of individual models. For example, although ResNeXt-101-FPN performs overall better than Hourglass104 in AP_{50} (54.7 vs. 50.3), the latter is better in the domain of medium angles (45.2 vs.

Model	L	LM	M	MH	H	AP_{50}^{avg}
ResNeXt-101-FPN	56.8	54.6	49.2	65	78.3	60.8
ResNet-50-FPN	32.8	29.8	23.5	40.5	48.9	35.1
Hourglass104	50.6	52.0	47.5	64.9	73.2	57.6
ResNet101	20.2	30.4	24.1	35.1	38.0	29.6
ResNet18	23.8	20.3	19.2	29.3	31.9	24.9
EfficientDet–*D*0	39.6	38.0	30.4	42.5	54.5	41.0

Table 5.6: Results on different altitude-domains. E.g. ResNeXt's AP_{50} performance in low-medium (LM) altitudes is 54.6 AP_{50}.

Model	A	AM	M	MR	R	AP_{50}^{avg}
ResNeXt101-FPN	68.3	55.1	45.2	63.6	51.5	56.7
ResNet50-FPN	32.8	35.5	32.7	35.7	27.6	32.9
Hourglass104	66.4	42.1	49.7	58.7	46.9	52.76
ResNet101	7.4	35.8	20.5	33.6	21.7	23.8
ResNet18	9.6	29.5	26.3	27.9	22.1	23.1
EfficientDet–*D*0	26.9	47.0	40.5	40.3	36.8	38.3

Table 5.7: Results on different angle-domains. For example, ResNeXt's AP_{50} performance in medium-right (MR) angles (57-73°) is 63.6 AP_{50}.

49.7). Furthermore, the great performance discrepancy between CenterNet-ResNet101 and CenterNet-ResNet18 in AP_{50} (36.4 vs. 21.8) vanishes when averaged over angle domains (23.8 vs. 23.1 AP_{50}^{avg}) possibly indicating ResNet101's bias towards specific angle domains.

Model	MOTA	IDF1	MOTP	MT	ML	FP	FN	Recall	Prcn	ID Sw.	Frag
FairMOT-D34 Zhang *et al.* (2020b)	39.0	44.8	23.6	17	17	3,604	9,445	57.2	77.8	307	1,687
FairMOT-R34 Zhang *et al.* (2020b)	15.2	27.6	33.7	6	37	2,502	12,592	30.1	68.4	181	807
Tracktor++ Bergmann *et al.* (2019)	55.0	69.6	25.6	62	4	7,271	3,550	85.5	74.2	165	347

Table 5.8: Multi-Object Tracking evaluation results for the **Swimmer** task.

Model	MOTA	IDF1	MOTP	MT	ML	FP	FN	Recall	Prcn	ID Sw.	Frag
FairMOT-D34 Zhang *et al.* (2020b)	36.5	43.8	20.9	28	49	3,788	20,867	47.2	83.1	447	1,599
FairMOT-R34 Zhang *et al.* (2020b)	30.5	40.8	27.3	29	127	4,401	28,999	40.2	81.6	285	1,588
Tracktor++ Bergmann *et al.* (2019)	71.9	80.5	20.1	123	5	7,741	5,496	88.5	84.5	192	438

Table 5.9: Multi-Object Tracking evaluation results for the **All-Objects-in-Water** task.

Model	L	LM	M	MH	H	AP_{50}^{avg}
F. ResNet-50-FPN	**32.8**	29.8	23.5	40.5	**48.9**	35.1
5×Altitude@3 (Chapter 3)	**32.8**	**29.9**	**26.2**	**41.5**	**48.9**	**35.9**

Model	A	AM	M	MR	R	AP_{50}^{avg}
F. ResNet-50-FPN	32.8	**35.5**	32.7	**35.7**	27.6	32.9
5×Angle@3 (Chapter 3)	**42.0**	**35.5**	**39.3**	**35.7**	**27.7**	**36.0**

Table 5.10: Results on different altitude- and angle-domains. The corresponding expert models share layers up until the third stage (@3) and are then split into 5 expert models (5×) beyond that point. For additional details, see Section 3.4.

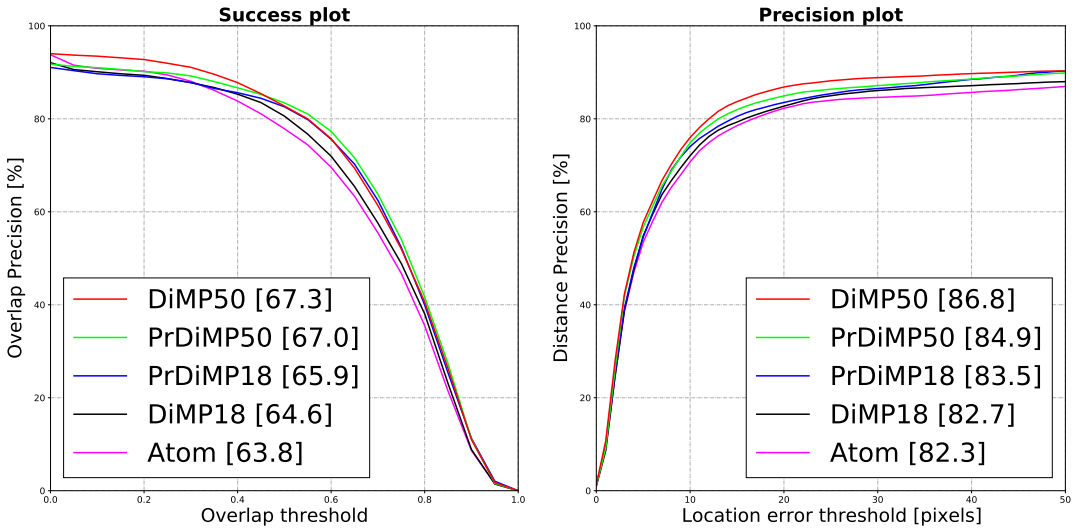


Figure 5.5: Success and precision plots for single-object tracking task.

5.4.2 Single-Object Tracking

Like VisDrone (Zhu *et al.* (2020a)), we provide the success and precision curves for single-object tracking and compare models based on a single number, the success score. As comparison trackers, we choose the DiMP (Discriminative Model Prediction) family (DiMP50, DiMP18, PrDiMP50, PrDiMP18) (Bhat *et al.* (2019); Danelljan *et al.* (2020)) and Atom (Danelljan *et al.* (2019)) because they were the foundation of many of the submitted trackers to the 2020 VisDrone workshop (Fan *et al.* (2020b)).

Figure 5.5 shows that the PrDiMP- and DiMP-family expectedly outperform the older Atom tracker in both success and precision. Surprisingly, PrDiMP50 slightly trails the accuracy of its predecessor DiMP50. Furthermore, all trackers' performances on SeaDronesSee are similar or worse than on UAV123 (e.g. Atom with 65.0 % success) (Bhat *et al.* (2019); Danelljan *et al.* (2020, 2019)), for which they were heavily optimized. We argue that in SeaDronesSee there is still room for improvement, especially considering that the clips feature precise meta information that may be helpful for tracking. Furthermore, in our experiments, the faster trackers DiMP18 and Atom run at approximately 27.1 FPS on an NVIDIA RTX 2080 Ti. However, we note that they are not capable of running in real-time on embedded hardware, a use-case especially important for UAV-based SAR missions.

5.4.3 Multi-Object Tracking

We use a similar evaluation protocol as the MOT benchmark (Milan *et al.* (2016)). That is, we report results for Multiple Object Tracking Accuracy (MOTA), Identification F1 Score (IDF1), Multiple Object Tracking Precision (MOTP), number of false positives

(FP), number of false negatives (FN), recall (R), precision (P), ID switches (ID sw.), fragmentation occurrences (Frag). We refer the reader to Ristani *et al.* (2016) for a thorough description of the metrics.
We train and evaluate FairMOT (Zhang *et al.* (2020b)), a popular tracker, which is the base of many trackers submitted to the challenge (Fan *et al.* (2020a)). FairMOT-D34 employs a DLA34 (Yu *et al.* (2018)) as its backbone while FairMOT-R34 makes use of a ResNet34. Another SOTA tracker is Tracktor++ (Bergmann *et al.* (2019)), which we also use for our experiments. It performed well on the MOT20 (Dendorfer *et al.* (2020)) challenge and is conceptually simple.
Surprisingly, Tracktor++ was better than FairMOT in both tasks. One reason for this may be the used detector. Tracktor++ utilizes a Faster-R-CNN with a ResNet50 backbone. In contrast, FairMOT is using a CenterNet with a DLA34 and a ResNet34 backbone, respectively.

5.4.4 Meta-Data-Aware Object Detector

Developing meta-data-aware object detectors is difficult since there are no large-scale data sets to evaluate their performances. However, some works provide promising preliminary results using this metadata (e.g. Wu *et al.* (2019) or Chapters 3 and 4 in this work). We provide an initial baseline from Chapter 3 incorporating the meta data. We evaluate the performances of 5×Altitude@3- and 5×Angle@3-experts, which are constructed on top of a Faster R-CNN with ResNet-50-FPN, respectively. Essentially, these experts make use of meta-data by allowing the features to adapt to their responsible specific environmental domains.

As Table 5.10 shows, meta data can enhance the accuracy of an object detector considerably. For example, 5×Angle@3 outperforms its ResNet-50-FPN baseline by 3.1 AP_{50}^{avg} while running at the same inference speed. The improvements are especially significant for underrepresented domains, such as $+9.2$ and $+6.4$ AP_{50}^{avg} for the acute angle (A) and the medium angle (M), respectively, which are underrepresented as can be seen from Fig. 5.4.

5.5 Conclusions

This chapter introduces a benchmark for UAV-based computer vision problems in maritime scenarios. We built the first large-scale data set for detecting and tracking humans in open water. Furthermore, it is the first large-scale benchmark providing full environmental information for every frame, offering great opportunities in the so-far underdeveloped area of multi-modal object detection and tracking. We offer three challenges: object detection, single-object tracking, and multi-object tracking by providing an evaluation server. We hope that the development of meta-data-aware object detectors and trackers can be accelerated by means of this benchmark. Moreover, we provide multi-

spectral imagery for detecting humans in open water. These images are very promising in maritime scenarios, having the ability to capture wavelengths, which set objects apart from the water background.
So far in this dissertation, we put a huge emphasis on computer vision on UAVs. Since we want to provide a holistic approach to maritime searching of people in distress, the next chapter deals with the trajectory planning of a drone in the presence of non-zero water current and wind flow.

Chapter 6

UAV Path Planning Algorithms for Maritime Search and Rescue Missions

6.1 Introduction

The increasing adoption of Unmanned Aerial Vehicles (UAVs) for maritime search and rescue (SAR) missions has introduced novel operational capacities. Specifically, UAVs provide enhanced endurance and real-time data transmission, which can complement traditional human-led SAR efforts. Nonetheless, devising efficient path planning for UAVs in this context remains challenging, primarily due to the variable and unpredictable characteristics of the maritime environment.

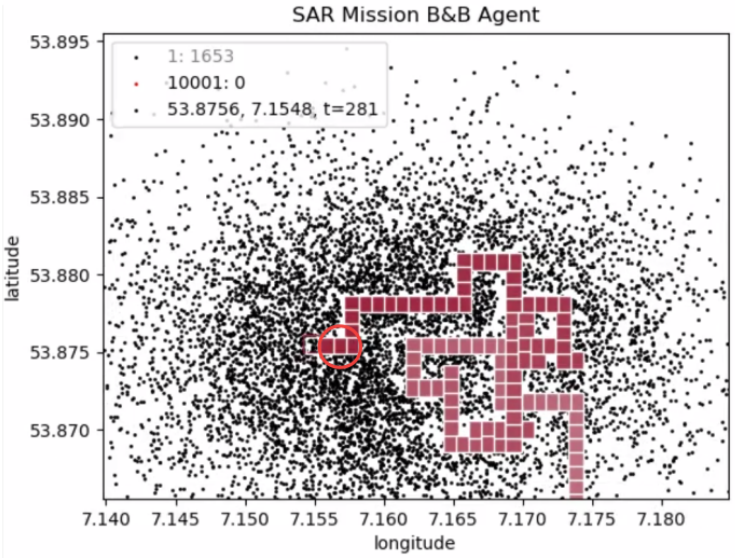


Figure 6.1: Example of a trajectory of the branch and bound agent (which we are proposing in this chapter) right after finding a search target. Here, it succeeded in doing so after approximately 30 minutes after take-off. The search target's position is highlighted by a circle. The plot is taken from our framework.

In maritime SAR scenarios, efficient coverage of the search area and accurate target localization are pivotal. Many popular existing path planning algorithms, like A* or Dijkstra (Russell (2010)), operate under the assumption of known and static environmental conditions. These assumptions are not applicable to maritime SAR for multiple reasons: Factors, such as water currents and wind dynamics, determine the search targets' trajectory, affecting their position and velocity. However, these factors are known very imprecisely at best. Furthermore, imprecise knowledge about locations of search targets in distress calls for probabilistic trajectory planning algorithms.

Extensive research in the area of maritime search and rescue utilizing Unmanned Aerial Vehicles has predominantly centered on computer vision, with a specific emphasis on object detection. This focus resulted in the publication of dedicated datasets (e.g. Chapter 5 in this work), works that delve into the intricacies of detecting small objects (Lee *et al.* (2018); Varga and Zell (2021)), exploration into the integration of supplemental sensor data to enhance detection efficacy (e.g. Chapters 3 and 4 in this work), and contributions from control theory (Raap *et al.* (2019)).

Therefore, there is an evident need for a more adaptive approach in UAV path planning for maritime SAR missions – one that considers both target location uncertainty and dynamic environmental factors. Hence, we analyze competitive path planning algorithms. All of them integrate probabilistic models to account for target location uncertainty and use real-time meteorological data to adapt to changing water currents and wind conditions.

Furthermore, this paper introduces a novel path planning algorithm, building upon the foundational principles of branch-and-bound (BnB) techniques (Sato (2008)). Our proposed method aims to bridge the gap from theory to practical application by leveraging the strengths of existing theory on BnB algorithms while tailoring them for real-world application. It takes into account environmental factors, such as water current and wind flow. Yet, in contrast to the existing literature on BnB-based path planning (Sato (2008); Raap *et al.* (2017, 2019)), our approach is designed to be computationally lightweight.

The main contributions of this chapter are the following:

- We propose a novel trajectory planning algorithm that aims at bridging the gap from easily applicable algorithms that take almost no environmental data into account to computation-heavy and theory-backed algorithms like branch-and-bound algorithms.
- The algorithms in this chapter were evaluated using a newly developed framework for researching and testing maritime SAR algorithms for UAVs. It is available on GitHub[1]. It is fully written in Python, which makes it easy to use.
- We compare and evaluate multiple trajectory planning algorithms and discuss their results.

© 2024 IEEE. Reprinted, with permission, from Messmer and Zell (2024).

[1] `https://github.com/cogsys-tuebingen/pathplanningrepository`

The structure of this chapter is as follows: Section 6.2 reviews the current research in this area. Section 6.3 explains some background and the algorithms and methods used here, including those related to the change of search targets over time and the main trajectory planning algorithms. Section 6.4 presents the experimental results and a subsequent discussion. Finally, Section 6.5 discusses possible future research and the limitations of the approaches presented in this chapter.

6.2 Related Work

In Martinez-Alpiste *et al.* (2021), the authors build a pipeline to perform search operations from a UAV equipped with a smartphone. They also record a dataset for the training of their neural network. While this is interesting work towards UAV-based SAR missions, they do not investigate the path planning problem. Similarly, there is a vast number of publications on computer vision from UAVs (Varga and Zell (2021); Du *et al.* (2019)) and also Chapters 2, 3, 4, and 5 in this work. While this is very important for automated SAR scenarios, it doesn't address the problem of how to compute the UAV's trajectory. The authors in Sato (2008) construct bounds for a branch and bound algorithm for the search problem with a single UAV. This finds an optimal solution to the problem. However, the algorithm works, as shown, merely on problems in the range of 10×10 to 20×20 grids, which is far too small for any realistic application. While Morin *et al.* (2010) uses an ant-colony optimization method, it suffers from the same problem. In Berger *et al.* (2013) the authors propose a mixed integer linear programming approach to solve this problem. While this also delivers exact solutions, it is again computationally too intensive for application in practice. In Riehl *et al.* (2007), the authors use graph-based model-predictive search to solve problems in the range of roughly 34×34 grids. That is already larger but still too small for most real-world applications. For comparison, in our experiments we usually used a grid size of 2500×2500, see section 6.4. The authors in Dagestad *et al.* (2018) developed OpenDrift, a framework to efficiently simulate drift of objects or substances in the ocean, such as oil spills, floating debris, life-rafts or vessels in distress. We will employ this work for the latter. Similarly, Wu *et al.* (2023) model the leeway drift of people in water. While this is very important, they don't investigate trajectory planning for maritime SAR missions. In Guoxiang and Maofeng (2010) the authors describe a full application to plan maritime SAR missions. They use trajectory planning methods similar to the spiral and boustrophedon search used later in this chapter.
The authors of Ghazali *et al.* (2016) describe an effective use of a swarm of quadcopters in a maritime search and rescue scenario where the wind flow and water current are unknown. While this is also interesting in some realistic application scenarios, we prefer to focus on the case where those are known in order to be able to develop better-informed algorithms. In Kratzke *et al.* (2010), the authors describe the planner model used by the United States Coast Guard; their environmental model is basically equal to the method

of OpenDrift. How the planning algorithm works, however, is not disclosed in detail, leaving the need for further research in this field.
In Roberts *et al.* (2016) the authors present a framework to analyze mSAR missions with UAVs. This might be of interest to practitioners when deciding whether or not to dispatch UAV to aid the search. We were not able to find their framework published, however. Furthermore, they do not develop any new path planning methods. Instead, they re-implement the ones used by the US Coast Guard similar to the boustrophedon and expanding spiral used later in this chapter.
In the paper Bourgault *et al.* (2006) the authors also address the single searcher problem using a parametric representation for the search target, which is appealing from a theoretical perspective. However, we argue a particle-based representation provides a more general probabilistic description of the search targets' movements. Additionally, the employed algorithm optimizes in a greedy manner, selecting the most favorable step at each point in time, which may not always lead to the best overall outcome. The authors of Li *et al.* (2023) investigate the sensor, communication, and control subsystems of a UAV platform potentially employed for maritime SAR missions. This is highly relevant for SAR missions, yet it gives no insight into path planning methods for the problem at hand. The work Tiemann *et al.* (2018) investigates the problem where the search target in distress is continuously sending a distress signal and use this to enhance the estimated target position. While distressed ships might be able to provide such a search aid, live rafts may not. Hence we explore the case, where the search target is not transmitting signals.

6.3 Method

Since our proposed method employs a branch-and-bound–type algorithm, the next subsection aims at recalling the necessary basics. The method we propose is then introduced in subsection 6.3.2.

6.3.1 Background – Branch and Bound for Path Planning Problems

Branch-and-bound (Land and Doig (1960)) algorithms are an abstract class of algorithms designed to solve a wide range of optimization problems from the field of discrete optimization, sometimes also referred to as mixed-integer linear programming (MILP). Some examples, which are usually solved with these algorithms, are job scheduling or the Knapsack problem (Taha (2007)). Additionally, branch-and-bound algorithms are applicable as well to more complex path planning problems, like the traveling salesperson problem (TSP) (Taha (2007)), which cannot be directly addressed by algorithms like A* (Russell (2010)) due to its special requirement of optimizing a path rather than reaching a specific goal position. Recall that other path planning algorithms like A* require a specific start and goal node as input. This makes a branch-and-bound–type algorithm

the preferred choice to solve the problem we are facing in the trajectory planning for maritime search and rescue drones. Here, we also need to optimize a flight trajectory for its probability of finding the search targets rather than compute a path from start to goal.

In the following, we will try to give an intuition of how branch-and-bound algorithms work on a concrete example (see Figures 6.2 and 6.3) as well as pseudo-code for a branch-and-bound algorithm adapted to path planning problems (see Algorithm 1). This is an adapted version of the one presented in Sato (2008) which we include here to allow for a complete understanding.

In general, any optimization problem must have the following form to be solvable by branch-and-bound: given some objective function $f\colon X \to \mathbb{R}$ and some additional constraints represented via $g\colon X \to \mathbb{R}^m$ and $c \in \mathbb{R}^m$, find $x \in X$ with

$$\begin{aligned} f(x) &= \max_{z \in X} f(z), \\ g(x) &\leq c. \end{aligned} \tag{6.1}$$

Here, the search space X is a discrete set consisting of candidate solutions in which the algorithm searches the optimal one. In the later path planning scenario X will be a set of potential flight trajectories for the UAV. The objective $f\colon X \to [0,1]$ will map a trajectory to the probability of finding the search target(s), and g will denote constraints on the trajectories, like the UAV's flight time.
Then, in a nutshell, any branch-and-bound algorithm consist of two parts:

- Branch: In the branching step, the current problem (or subproblem) is divided into smaller, more manageable subproblems, until solving it becomes possible. The goal is to make the subproblems simpler to solve or bound (also known as divide and conquer). How the division is implemented depends entirely on the specific problem.

- Bound: Calculate an upper bound for the objective function on the currently investigated subproblem. This upper bound can be compared to the best yet observed candidate solution. If the upper bound is smaller, the subproblem under investigation can be pruned. The algorithm is able to disregard the whole branch originating from this subproblem. Like for the branching step, how to actually calculate an upper bound, preferably computationally efficiently, is highly dependent on the problem at hand.

These two foundational principles of the branch-and-bound algorithm make for a striking resemblance to the well-known α-β–pruning algorithm as described in Russell (2010). In both approaches child nodes are generated from the current node and evaluated based on a heuristic. Subsequently, nodes yielding a low heuristic value compared to others are pruned (i.e. not further expanded). In both cases, this pruning is central in making either

algorithm a preferred choice. It essentially makes the difference in between an efficient algorithm and a brute-force one.

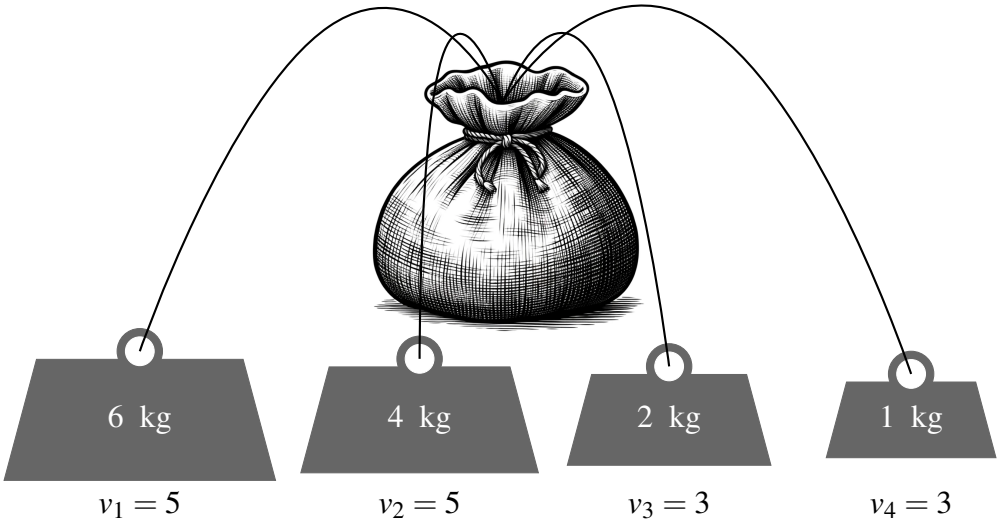


Figure 6.2: The Knapsack problem. What is the best way to maximize the bag's content's value if the bag can hold at most $W_{\max}$ kg?

To demonstrate the functionality of branch-and-bound algorithms, we illustrate its performance on the well known Knapsack problem (Taha (2007)). The challenge is: Given $n \in \mathbb{N}$ objects with weights and values $w_j, v_j > 0$ and a sack which can hold at most a weight of $W_{\max} > 0$. How can we pack the objects such that the value of the packed items is maximized while the total weight does not exceed the sack's capacity?

Formally, we have $X = \{0,1\}^n = \big\{(x_1,\ldots,x_n) : x_k \in \{0,1\}\big\}$, where each $(x_1,\ldots,x_n) = x \in X$ denotes the decision whether to put any of the items into the sack via a binary representation. That means, the kth item has been placed in the sack if $x_k = 1$ and has not if $x_k = 0$. Then the objective function and constraints from Equation 6.1 take the form

$$f(x) = \sum_{k=1}^{n} x_k \cdot v_k,$$
$$g(x) = \sum_{k=1}^{n} x_k \cdot w_k \leq W_{\max}.$$

To provide an example, we will use Algorithm 1 to solve the problem with $n = 4$, $(w_1,\ldots,w_4) = (6,4,2,1)$ and $(v_1,\ldots,v_4) = (5,5,3,3)$, as illustrated in Figure 6.2 and Figure 6.3. The branch-and-bound algorithm explores branches of a decision tree, where

each node represents a state with a subset of considered items. The decision at each node is whether to include the current item in the knapsack. This binary choice leads to two branches: one where the item is included and one where it is excluded. To denote this branching, we use the notation $(0,1,\_,\_)$ to indicate the state where the first item is excluded, the second included, and the rest is not yet decided. The process begins with the first item ($k = 1$). The algorithm keeps the best solution yet discovered, initialized with an empty solution that has a total value of 0 and a total weight of 0, represented as $(\_,\_,\_,\_)$. At each step, the algorithm considers the next item. It generates two new nodes:

- Include k: If adding the current item does not exceed the weight limit, create a new node with this item included, updating the total weight and value.
- Exclude k: Create a new node where the current item is not included, keeping the total weight and value unchanged.

This branching effectively splits the problem into two subproblems: one considering the solution with the item and one without it. The remaining search space is $\{0,1\}^{n-k}$, the tail of the binary vector. For each generated node, the algorithm calculates a bound to determine if further exploration might lead to a better solution than the best found so far. This bound is highly problem specific. For the Knapsack problem, it makes sense to add the values of all items that could potentially still be included, disregarding the weight constraint. Formally, let $x = (x_1,\ldots,x_k,\_,\ldots,\_)$ denote the state, then we define the bound $\bar{q}(x)$ as

$$\bar{q}(s) = \sum_{j=1}^{k} x_j v_j + \sum_{j=k+1}^{n} v_j.$$

If the calculated bound of a node is lower than the value of the best solution found so far, the node is pruned; that is, it is not explored further as it cannot lead to a better solution. The algorithm iterates through all items, branching and bounding at each step. It keeps track of the best solution encountered so far. The algorithm terminates when all nodes have been either fully explored or pruned. The solution associated with the highest value among the final nodes is the optimal set of items to include in the knapsack. Figure 6.3 visually shows the entire solving process conducted by the branch-and-bound algorithm as described here.

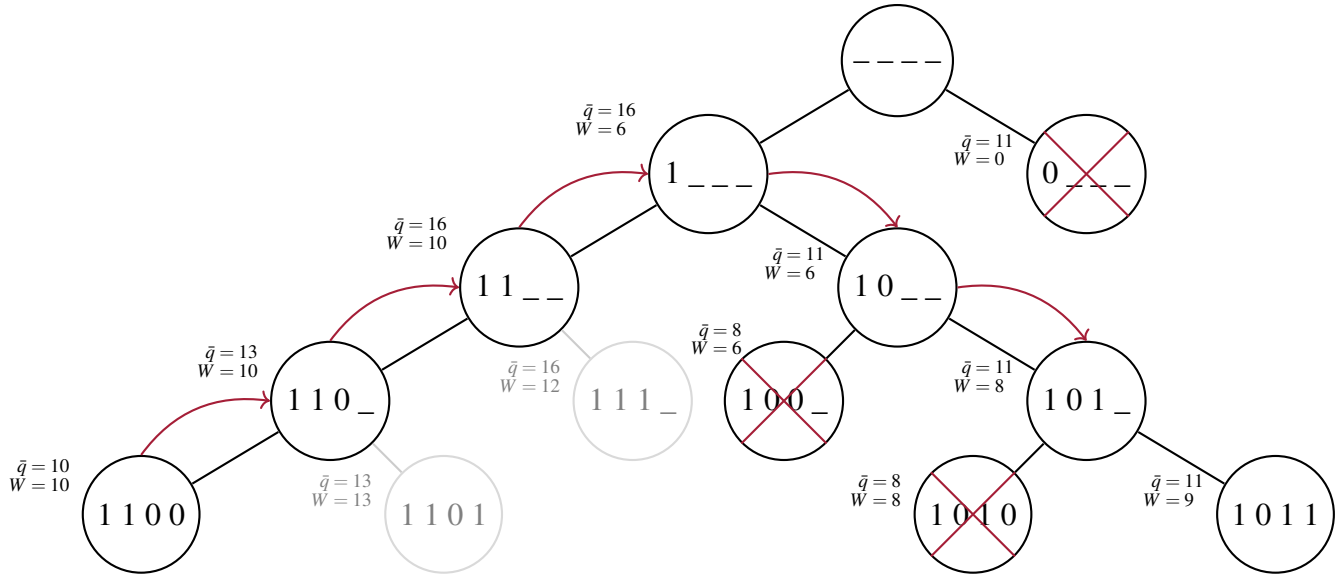


Figure 6.3: The search graph of a branch-and-bound algorithm solving the Knapsack problem from Figure 6.2 demonstrates how the algorithm operates. By design of the bound function, it is expanding nodes that include the respective items (adding a '1' instead of a '0') until further expansion is not possible. This leads the algorithm to descend to '1 1 0 0' in the bottom left (the grey nodes are invalid child nodes, as their weight is larger than $10 = W_{\max}$). Then, it propagates the best seen solution, with a value of $\hat{q} = 10$, back upwards as indicated by the red arrows. During backtracking, the algorithm explores the previously ignored node '1 0 _ _'. Here, it prunes the node '1 0 0 _' (because of its bound $\bar{q}(\text{'1 0 0 \_'}) = 8 \leq 10 = \hat{q}$) and instead explores '1 0 1 _'. At this point, the node '1 0 1 0' is also pruned, as its bound is worse than the best solution observed so far, leading to the discovery of the optimal solution '1 0 1 1'. The best observed value is updated to $\hat{q} = 11$. This is backtracked (not shown in the graph for simplicity) up to the root node, where the last remaining node '0 _ _ _' is pruned because $\bar{q} = 11 \leq 11 = \hat{q}$ and the algorithm terminates. Observe, that the returned solution '1 0 1 1' has an optimal value of $\bar{q} = 11$, but it is not unique. Another possible solution is '0 1 1 1', which also has a value of $\bar{q} = 11$ but a lower weight of $W = 7$. This solution could be found in the pruned subtree originating from '0 _ _ _'. However, due to the design of the branch-and-bound algorithm, it is indifferent to the notion of a 'better' solution based on lower weight.

Algorithm 1 Branch and Bound Algorithm for path planning problems. This is an adapted version of the one presented in Sato (2008).

Require: Graph (V,E), where V is the set of vertices and E is the set of edges.
Require: UAV position $v_0 = (x,y)$, maximum flight time $T \in \mathbb{N}$.
$t \leftarrow 0, \hat{q} \leftarrow 0, \mathcal{Q}(0) \leftarrow \{(v_0, 0, \infty)\}, \mathcal{Q}(t) \leftarrow \emptyset \; \forall t > 0.$ ▷ Initialization
do
  if $\mathcal{Q}(t)$ is empty **then** ▷ Backtracking, after full exploration at time t.
    $t \leftarrow t - 1$
  else
    $(v_t, t, \bar{q}(v_t,t)) \leftarrow \text{pop}\big(\mathcal{Q}(t)\big)$ ▷ The triple with largest bound $\bar{q}$ is popped.
    if not $\bar{q}(v_t,t) \leq \hat{q}$ **then** ▷ Pruning, if no improvement possible.
      if $t = T$ **then**
        $\hat{q} \leftarrow \bar{q}(v_t,t)$ ▷ Update best observed probability.
      else
        for $v \in \mathcal{N}(v_t)$ **do** ▷ Branching; $\mathcal{N}(v_t)$ is the set
          Append $\big(v, t+1, \bar{q}(v,t+1)\big)$ to $\mathcal{Q}(t+1)$. ▷ of neighbors of v_t.
        end for
        $t \leftarrow t + 1$
      end if
    end if
  end if
while $t > 0$

Unlike the simple Knapsack problem (Figure 6.2), where branches in the solution graph (Figure 6.3) can never meet again, *transpositions* – such as moving left then down resulting in the same position as moving down then left – can occur. However, in an open maritime environment, where search targets may also be moving, differently ordered sequences of the same movements by the drone might lead to varying probabilities of finding all search targets. Consequently, in Algorithm 1, it is necessary to consider the time t as well, to distinguish between different trajectories. Therefore, there is a queue $\mathcal{Q}(t)$ for each point in time t. It contains triples $(v,t,\bar{q})$, where v is a node in the search graph representing a potential drone location with additional meta data like the search target's probability distribution at time t and its parent node. The variable t denotes the point in time, while $\bar{q}$ is the calculated bound for the detection probability on the currently investigated trajectory. Further details on these quantities will be discussed in the next subsection.

6.3.2 Solving the mSAR Path Planning Problem

To run and test the planning algorithms at hand, we first developed a framework to simulate the flight trajectory of an UAV. It is available on GitHub[2]. Plots produced by our framework are shown in Figures 6.1, 6.4, and 6.7. Briefly summarized, it contains the following features:

- The first step of every simulation in our framework is to run an OpenDrift (Dagestad *et al.* (2018)) simulation. OpenDrift (Fig. 6.5, section 6.3.4) is an open-source framework that models drift trajectories of objects or substances in the ocean, such as oil spills, floating debris, or in our case, targets for search and rescue (modeled as life-rafts). By leveraging the capabilities of OpenDrift, our framework distributes particles in the simulated maritime environment. These particles are a non-parametric model of the probability distribution describing the potential location of the search target as this is usually not precisely known for maritime SAR missions. The distribution location of these particles is defined by the user to model the specifics of the search scenario.

- Subsequent to the particle distribution, our framework creates a grid in the search area. This grid serves as a defined movement space for the UAVs. The dimensions of the grid as well as the grid tile's size are freely specified by the user. Depending on the sensor, larger or smaller grid tiles might be adequate for the scenario.

- The main component of our framework is its ability to constantly monitor and update the state of each particle within the simulation. Once a particle is observed, our framework deletes it, see section 6.3.3. That is a key feature as observed particles need not be taken into account by the UAV for subsequent planning of the trajectory.

We compare distinct UAV path planning strategies specifically tailored for maritime search and rescue. Recognizing the dynamic nature of maritime environments, each strategy under consideration incorporates varying degrees of water current and wind information to enhance search efficiency. To provide a comprehensive evaluation, we looked into three strategies from the literature; the first two of them are used in practice (CCG Manual (2023); Roberts *et al.* (2016)), while the third is building on existing work (Sato (2008)) and aims to bridge the gap from theory to application. Fig. 6.6 shows a schematic drawing of their functionality.

1. Expanding Spiral Method: This strategy, already implemented by the Canadian and US Coast Guard (CCG Manual (2023); Roberts *et al.* (2016)), begins its search from the presumed location of the search target. The UAV then follows an outward spiral pattern, progressively increasing the radius of the spiral as the search continues. The idea is to

[2] `https://github.com/cogsys-tuebingen/pathplanningrepository`

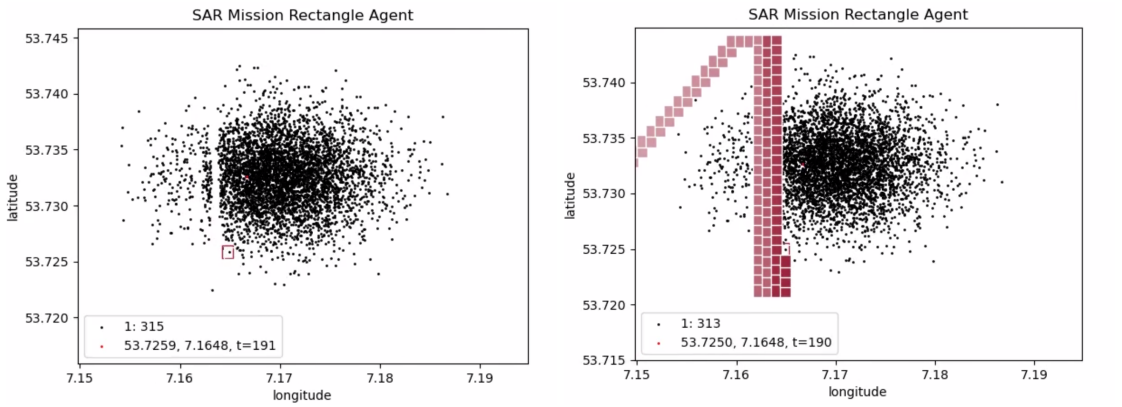


Figure 6.4: Two example plots taken from our framework. The agent performs the boustrophedon rectangle method. On the right hand side, the agent is plotted with the trace of its trajectory for better overview of its performance. Recently visited grid cells are plotted in dark red while cells which were visited longer ago are lighter. On the left, the plot contains no trace to have a better look at the particles. The legends contain position, time, and found particles.

start the search close to the last known or estimated location and then expand outwards in a systematic manner. Furthermore, the estimated location is computed by using the particles, which incorporate water current and wind data to model their trajectory.

2. Boustrophedon Rectangle Planner: This approach entails the allocation of rectangular search zones across areas designated as high-probability zones. The UAV then conducts its search in a boustrophedon (back-and-forth) manner within each rectangle before moving to the next. Similar to the expanding spiral method, particles are used in tracking the estimated position of the search targets, adjusting for the influence of water current and wind dynamics over time. Again, this approach is already used in practice, see Kratzke *et al.* (2010).

3. Global-Local Branch-and-Bound Method: Our proposed approach leverages a hybrid model for trajectory planning. On a global scale, similar to the boustrophedon method, this planner starts by identifying and estimating rectangular zones that encapsulate a predetermined portion of particles. The UAV first heads to these regions. Upon reaching the designated area, the approach shifts to a local scale. At this level, the UAV employs a modified version of the branch and bound algorithm (Sato (2008); Clausen (1999)), specifically leveraging the smaller search space. This two-tier system ensures the UAV covers vast areas quickly while maintaining the precision necessary for finer, more detailed searches. A more in-depth explanation of the details will be given later in this section.

Fig. 6.6 illustrates the first two planning methods. Specifically, the spiral planner computes the center of gravity for each of the particle clouds. Then, it calculates the

shortest path visiting all of them and moves from one to the next. Given that real-world situations typically have a small number of targets, one can comprehensively examine all possible permutations. Once arrived, it starts searching for a search target by performing an outward spiral until it observes a target. Next, it proceeds to the next search target. Over time, all search targets' estimated locations are updated according to the modeled probability distribution given by the particles.

The boustrophedon planner acts in a similar fashion by placing rectangular search areas on the map. For each particle cloud, it constructs a rectangle large enough to contain a predefined portion η of each collection of particles. Later, in our experiments (section 6.4), we chose $\eta = 0.75$, because this choice empirically gave the best results. Once arrived at a rectangle, the UAV traverses it in a boustrophedon fashion, meaning it is flying back and forth to cover the whole ground area in the rectangle.

Algorithm 2 Global-local Branch and Bound Planner.

Require: Number of targets $n \in \mathbb{N}$
Require: Targets with associated particles $L = \{(x_k, P_k)\}_{k=1}^{n}$
Require: Containment percentage $0 < \eta < 1$
Require: UAV position $u = (x, y)$
1: $L \leftarrow \text{OrderedList}(L)$ ▷ Order L to form shortest path
2: $R \leftarrow \{R_k\}_{k=1}^{n}$ ▷ Rectangles R_k enclosing η particles of P_k
3: **for** $k \leq n$; $k++$ **do**
4: **if** u is at R_k **then**
5: $u \leftarrow$ `Branch&BoundSearch`(u, R_k)
6: **else**
7: $u \leftarrow$`Advance`(u, R_k)
8: **end if**
9: $R \leftarrow$`UpdateSearchAreas`$(\{R_k\})$
10: **end for**

Algorithm 2 shows the pseudo code for our branch and bound planner. The subprocedure '`Advance`' flies the UAV from its current position into the direction of the next rectangle in the queue. The call to '`UpdateSearchAreas`' recalculates the rectangular search areas to account for particle drift during the course of the simulation. Finally, '`Branch&BoundSearch`' performs a branch and bound algorithm on the rectangular search area. For details of this algorithm see Section 6.3.1 or Clausen (1999); Land and Doig (1960), for its specific application to trajectory planning problems see Sato (2008) and Algorithm 1.

By only applying an actual branch-and-bound search to the designated search area, like in Algorithm 2, we reduce the search space for the problem drastically. Depending on the chosen containment percentage η this effect can be influenced. But even for $\eta = 1$, the effect is already remarkable: for the experiments done in Section 6.4 the grid size inside the search rectangle could sometimes be reduced to 50×50 tiles instead of the initial

2500×2500. Furthermore, instead of computing a precise upper bound q in Algorithm 1 – which is computationally expensive – we employ a heuristic, accepting a possibly sub-optimal solution while making the algorithm applicable in practice. Specifically, the heuristic we employ merely computes a sum of the particles in a vicinity to the investigated position weighted by their distance, achieving that locations with a higher number of particles nearby are valued higher to the algorithm than others. This is a greedy approach, yet it worked well in our experiments (see Section 6.4).
More formally, given a particle $p = (p_x, p_y)$ and UAV position $u = (u_x, u_y)$. Let

$$r_p = \exp\left(-\alpha \cdot |p_x - u_x|\right) + \exp\left(-\alpha \cdot |p_y - u_y|\right).$$

Now we can define the bound as

$$q = \frac{1}{|P|} \sum_{p \in P} r_p,$$

where P is the set of all associated particles with the search target. Summarized, we sum over the particles weighted by their distance and normalize by the number of particles. Above, $\alpha > 0$ is a hyper parameter. In our experiments, setting $\alpha = 1/2$ yielded the best results. Thus, this value was selected.

6.3.3 Particle Filter with negative Measurements

Particle filters, commonly used for state estimation, rely on representing a system's uncertainty through a set of $N \in \mathbb{N}$ weighted samples (particles) that collectively describe the system's probabilistic belief over its state, usually denoted $\mathcal{M} = \{(x_k, \omega_k) | 1 \leq k \leq N\}$ (Choset *et al.* (2005); Elfring *et al.* (2021)). An integral component of the particle filter process is the update step, where the weight ω_k of each particle is adjusted based on the likelihood of an observed measurement y given that particle's state x_k, that is

$$\omega_k \leftarrow P(y|x_k).$$

The measurement process in this work's setup is special in the sense, that the UAV either observes a search target in a grid cell, or it does not. This translates to a particle filter in the following way. Assume our UAV is observing grid cell (i, j) (denoted $\text{cell}_{(i,j)}$) at time t. Then either we have $y = 1$, if the search target is contained in $\text{cell}_{(i,j)}$, or $y = 0$ otherwise. Hence, the relevant cases for the particle update are

$$P\left(y = 0 | x_k \in \text{cell}_{(i,j)}\right) = \varepsilon, \tag{6.2}$$

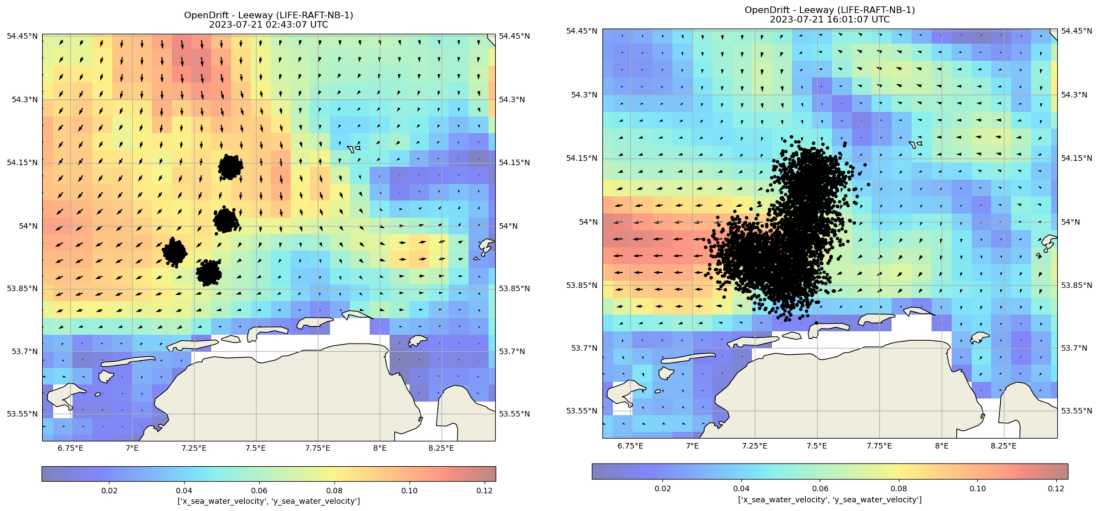
$$P\left(y = 0 | x_k \notin \text{cell}_{(i,j)}\right) = 1 - \varepsilon. \tag{6.3}$$

Here $0 \leq \varepsilon < 1$ accounts for sensor detection errors. Later, in our experiments (section 6.4), we chose the flight altitude low enough that we can safely assume $\varepsilon = 0$. Expression

(6.2) equals zero, because particle $x_k \in \text{cell}_{(i,j)}$ at time t, yet we observed, that no search target is present. The weight update for particles which are not in $\text{cell}_{(i,j)}$ is given by equation (6.3). In the cases, where $y = 1$, the UAV found the search target and we stop searching. Therefore, in the resampling step of the particle filter, the particles contained in $\text{cell}_{(i,j)}$ are resampled with probability 0, thus being erased.

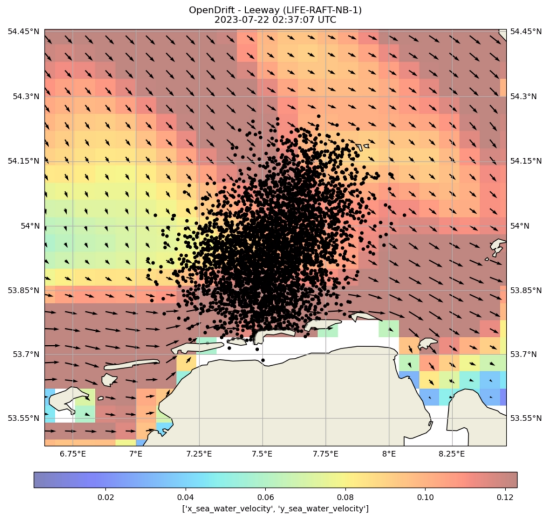
6.3.4 Search Targets' Movement Model

In examining the dynamics of distressed search targets and their movement patterns, this study utilizes the OpenDrift framework (Dagestad *et al.* (2018)). This open-source tool offers a reliable simulation platform for a variety of floating objects, like ships and life rafts, while also including debris and oil spills in aquatic settings. Notably, OpenDrift's design draws from the simulation models employed by the United States Coast Guards' planning software (Kratzke *et al.* (2010)). The following provides a brief summary of the elements relevant to this research. Figure 6.5 shows examples from an OpenDrift simulation. In maritime search and rescue operations, accurately predicting the movement of search targets in the ocean is of high importance. At the beginning of a search mission, represented as $t = 0$, a total of $n \in \mathbb{N}$ particles are sampled from a bivariate Gaussian distribution surrounding the initial belief for the position of each search target. The variance of this distribution corresponds to the inherent uncertainty in our initial beliefs regarding the precise location of the targets. If the uncertainty is smaller or larger, so is the variance from which we sample the particles. To model the trajectory of these particles over subsequent time intervals, we employ the Lagrangian movement model (Breivik and Allen (2008)) embedded within the OpenDrift framework. This model offers a robust estimation of each particle's path, taking into account the influences of both water currents and wind flow. To more comprehensively mirror the real-world scenario, OpenDrift incorporates a minor diffusion factor into the trajectory of each particle. This factor serves to simulate the progressively increasing uncertainty associated with the position of the search targets over time. The wind and water flow information is gathered from HYCOM (2003) and Iwamoto *et al.* (2016), respectively. Their respective provided resolutions are roughly 5 km^2 and 67×33 km^2.



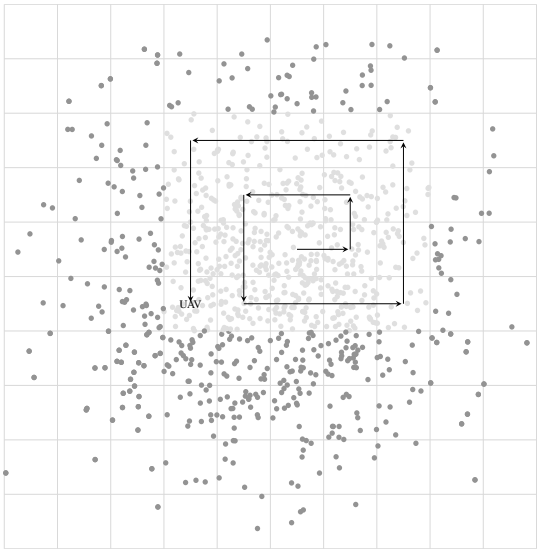
(a) At the start of the simulation.

(b) After twelve hours of the simulation.

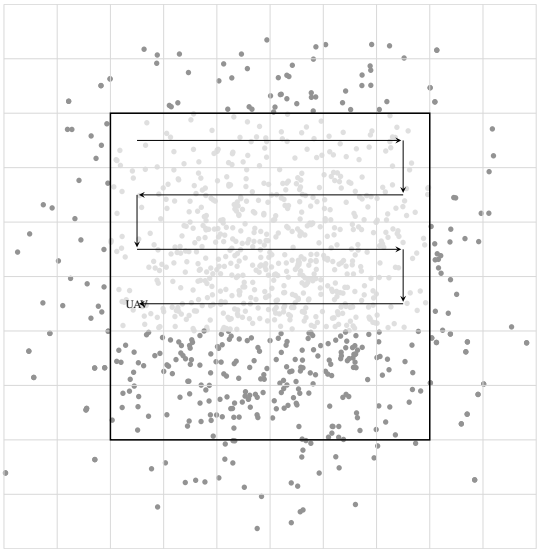


(c) After 24 hours of the simulation.

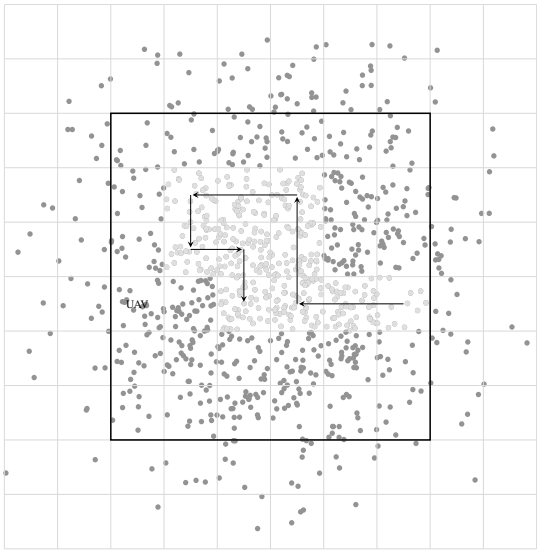
Figure 6.5: Three output plots from the OpenDrift framework. It simulates four search targets, observable as four particle clouds in the top left image, showing the start of the simulation. The top right and bottom images show the evolution of the simulation after twelve and 24 hours. The background shows the underlying water flow, changing over time. The location is roughly at 54.0 N, 7.5 E.



(a) Example of an expanding spiral path.



(b) Example of a boustrophedon path.



(c) Schema of a possible B&B trajectory.

Figure 6.6: Schematic drawing of the three algorithms under investigation.

6.4 Experiments

The main goal of this chapter is to emulate real-world conditions as closely as possible. We aim to bridge the significant gap between algorithms, which are good in theory but not applicable, and their tangible, practical applications in maritime SAR operations. The following experiments reflect that as well.

One of the critical factors we had to consider was the grid tile's size. For our simulations to reflect real-life conditions, it is essential to capture the realistic surface area that a standard optical sensor would cover when deployed on a UAV. Chapters 3 and 5 show, that any human in the water should be detected by an object detector running on the UAV when flying at around 100 m altitude. Therefore, we've determined that a square tile measuring approximately 100 m on each side represents the area a UAV would typically survey when flying at this altitude. That ensures, that we can safely mark a particle as observed once found in the same grid tile as the UAV.

Furthermore, the UAV's speed is based on the capabilites of smaller fixed-wing drones. With a modeled speed set at 18 $^m/_s$ (ElevonX (2023); Quantum (2020b)), our simulations mirror the typical operational speeds of these drones, which we argue are the most fitting for maritime SAR mission – they are affordable, relatively easy to use, and compromise in between the flexibility of multi copters and the endurance and speed of small air crafts while not requiring a pilot. For further details, see the discussion in 2.

air speed	altitude	field of view	flight time
18 $^m/_s$	100 m	100×100 m^2	$1-5$ h

Table 6.1: Specifications of simulated UAV.

Smaller fixed-wing drones equipped with electrical engines have approximately $1-2$ hours of average battery life span (ElevonX (2023); Quantum (2020b)) while employing a combustion engine this class of UAVs can reach a flight time of 5 hours (ElevonX (2023)). In practice, electrical engines are less error-prone and require less maintenance. Also, in case of an accident, the environmental burden is smaller compared to engines running on gasoline.

To cover both cases in our simulations, we conducted experiments with a battery life span of $1, 2$, and 5 hours, trying to be as realistic and close to practical application as possible. Table 6.1 shows an overview of the technical specifications of the simulated UAV.

In all described experiments contained in this section, we used $10,000$ particles per search target for the OpenDrift simulation. They were sampled roughly 2 km around the simulated position of each target. The grid size used is 250×250 km^2 with a tile size of 100 m, resulting in a 2500×2500 grid.

In all experiments, the UAV's take-off area is right on shore. Precisely, it takes off at 53.722827 $N, 7.192965$ E. Search targets were sampled at a distance of $10, 20$, or 30 kilometers in the sea. The experiments become more challenging for the agents as the distance increases because the drone requires more time to reach the search area. This delay allows the particles to disperse more before the UAV begins its search. For example, at a speed of 18 $^m/_s$ (see Table 6.1), the UAV needs roughly 9.5 minutes to fly 10 kilometers, but almost 28 minutes to travel 30 kilometers. In all experiments, there are

either one or two targets. First, we randomly drew whether to sample one or two targets based on a uniform distribution. Next, we sampled the specified number of search targets at the respective distance. We report the success rates, the time to detect the first search target, and the success rate for finding the first search target only. These metrics are calculated as the average across all runs with identical distances to the targets. For each table and agent, the reported numbers are averages taken over roughly 500 runs.

Of course the most important metric is the success rate, defined as the number of found targets divided by the total number of search targets that could have been found. The other two additional metrics provide further valuable insights. Although secondary, it is also crucial to optimize for finding search targets quickly; for example, if the targets are humans, a quicker detection helps prevent them from cooling down too much. The success rate for only the first search targets, when compared with the overall success rate, shows whether the agent was able to detect the second target as well. This comparison is valuable because it reveals how the agents handle higher uncertainty, as these targets have been adrift for a longer period, causing their corresponding particles to disperse more. This is similar to the scenarios on search targets at larger distances. Roughly equal values for 'success rate' and 'success rate 1st' suggest, that the agent was generally successful in finding the second search target. On the other hand, if there is a significant difference between the two – relative to the overall success rate – it indicates that the second search target was often missed.

The tables are arranged in ascending order based on the distance from the UAV's take-off to the search targets. The lines 'Spiral' and 'Boustrophedon' show the results for the respective agents as described in section 6.3. In these experiment, for the boustrophedon agent the portion of particles that is contained in a rectangle to be searched is 0.75. 'B&B 15', 'B&B 35', and 'B&B 50' denote the branch and bound planners from section 6.3; the number indicates the maximum duration, in minutes, that the UAV dedicates to searching for a target inside a rectangle using the branch and bound method. Specifically, the first seeks for 15 minutes, the second for up to 35 minutes, and the last for 50 minutes. Once the UAV identifies a search target, it immediately halts the search and moves on to the next target, indifferent to the time already spent.

Table 6.2 shows the results for the experiments where the search targets were sampled roughly 10 *km* away from the UAV's take off. Judging by their performance, it is the easiest task for the planning algorithms. This makes sense, as the UAV does not need to fly far before arriving at the position of the first targets. Hence the uncertainty about the position while searching is relatively low. Especially the spiral agent profits from that, as it starts its search at the point of highest probability, the center of the particle cloud, making it the best performing planner in this case. Its time of success for the first target is close to optimal. We were surprised by the boustrophedon planner's worse performance compared to the spiral. We argue that is due to the rectangular agent starting its search at the edge of the particle cloud, a low probability area. This allows the particles to disperse before exploring areas with a higher likelihood. The branch-and-bound agents, B&B15, B&B35, and B&B50 perform well, each outperforming the boustrophe-

	success rate ↑	time 1st ↓	success rate 1st ↑
Spiral	**0.73**	15.0 min.	**0.80**
Boustrophedon	0.48	46.1 min.	0.60
B&B 15	0.51	21.0 min.	0.65
B&B 35	0.52	23.0 min.	0.67
B&B 50	0.62	26.9 min.	0.72

Table 6.2: Experimental results for search targets roughly 10 *km* off shore. The 'time 1st' values represent the average number of minutes for the respective agent to locate the first search target. The values for 'success rate 1st' indicate the success rate for finding the first search target only. Each line shows an average over roughly 500 runs.

	success rate ↑	time 1st ↓	success rate 1st ↑
Spiral	0.44	30.2 min.	0.55
Boustrophedon	0.43	54.2 min.	0.57
B&B 15	0.35	32.2 min.	0.46
B&B 35	0.44	35.9 min.	0.60
B&B 50	**0.54**	41.3 min.	**0.68**

Table 6.3: Experimental results for search targets roughly 20 *km* off shore. The 'time 1st' values represent the average number of minutes for the respective agent to locate the first search target. The values for 'success rate 1st' indicate the success rate for finding the first search target only. Each line shows an average over roughly 500 runs.

don planner in this scenario. However, they are unable to unfold their full potential in the easiest scenario. This can be seen by the most capable one, B&B50, still trailing the spiral agent by roughly 10 percentage points. In general, the success rates for the first target are quite close to the overall success rates, indicating that all agents are generally capable of finding the second target as well.

Table 6.3 shows the search results for targets sampled at a distance of roughly 20 *km*. In these experiments, we see the spiral agents' performance deteriorating quickly (compared to Table 6.2) as the uncertainty of the search targets' location grows. It starts searching in the center of a particle cloud, but since this cloud's particles already dispersed by the time the UAV arrives, the search target does not need to be near the center. Therefore, in some cases, the UAV is simply not fast enough to reach the search target because the drone works its way outward in increasingly larger circles. However, in the successful cases, this agent is the quickest at finding the search target. Surprisingly, the boustrophedon agent's performance does not decrease as much. However, the time spent on finding the first search target is significantly higher than the others, indicating that it

	success rate ↑	time 1st ↓	success rate 1st ↑
Spiral	0.10	59.0 min.	0.14
Boustrophedon	0.17	73.8 min.	0.23
B&B 15	0.16	51.2 min.	0.23
B&B 35	0.18	51.3 min.	0.27
B&B 50	**0.30**	58.5 min.	**0.37**

Table 6.4: Experimental results for search targets roughly 30 *km* off shore. The 'time 1st' values represent the average number of minutes for the respective agent to locate the first search target. The values for 'success rate 1st' indicate the success rate for finding the first search target only. Each line shows an average over roughly 500 runs.

generally locates targets through persistence rather than efficiency. For the branch and bound agents, while their performance also decreases, it is not as drastic as for the spiral agent. Also, they deliver the best search performance for this scenario. The ratio between success rate for the first target and overall success rate[3] shows, that the spiral agent and B&B50 are the agents with the best balance for the trade-off between searching for the first target and giving up on it in favour of a chance on finding the second.

Table 6.4 shows the results for experiments with targets at a distance of roughly 30 *km* from the UAV's take-off position. Notably, the performance of all agents is relatively low compared to the results shown in Tables 6.2 and 6.3, confirming the assumption, that this is the hardest task, as the search targets are sampled at the largest distance to the take-off position. We observe that B&B50, the agent investing the most resources in finding each target, achieves the highest performance, both overall and for the first search target. The ratio between success rate for the first target and overall success rate is the lowest for B&B50, meaning that it is the agent most successful at finding the second search targets[3]. That is, although it spends quite some time on the first search target, making the agent's belief about the second target's location very uncertain. Notably, the spiral agent's search time is not the smallest for this task.

6.5 Conclusion and Outlook

We have developed an improved path planning algorithm for UAVs in maritime search and rescue missions. Recognizing the challenges of dynamic maritime conditions and

[3]For each run, we sampled either one or two search targets with uniform probability. Therefore, there are approximately as many runs with two targets as there are with one target or, roughly speaking, twice as many first search targets as there are second search targets over all runs. Hence, for an agent never finding the second search target, we would expect a ratio of $3/2$ between 'success rate 1st' and 'success rate'. Conversely, an agent with comparable performance in finding both the first and the second search targets would have a ratio close to 1 between these two quantities.

uncertain target locations, our method integrates real-time meteorological data and probabilistic models. This offers a more adaptive and effective approach than existing solutions. Our research also highlighted the subtle differences of traditional particle filters when primarily faced with negative measurements in maritime contexts. Looking forward, we aim to further refine the presented algorithms. Investigating coordinated multi-UAV missions could also improve maritime search and rescue operations.

The next chapter gives concluding remarks for this dissertation and provides an outlook on what steps might be interesting to take from here to extend this research.

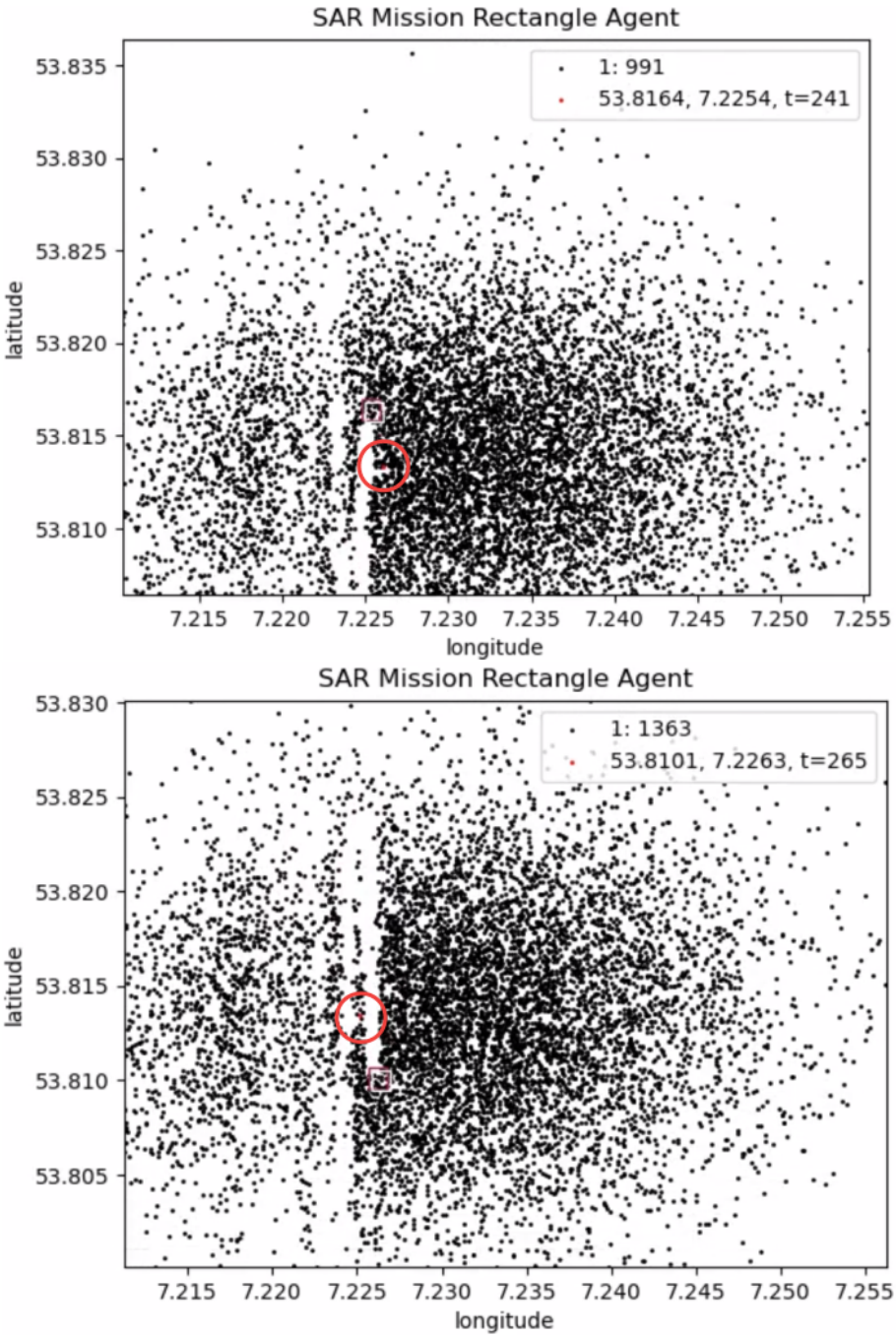


Figure 6.7: An unfavorable case for boustrophedon search: The two images show two closely consecutive moments in a target search, as can be seen by the simulation time in the top right corner of either image. In the top image, the UAV (red and white square) is moving north in a straight line, then turning at the northern end of the search area to fly south afterwards – this is shown in the bottom image. The search target (red particle, highlighted by a circle around it) moves west of the UAV's position while the drone is turning around at the northern edge of its rectangular search pattern. Plots are taken from our framework.

Chapter 7

Conclusion

The goal of this dissertation is to improve UAV–based maritime search and rescue (mSAR) operations, particularly the search part. In order to achieve this, we explored various topics, including computer vision and path planning. The results in the different areas are discussed in this summarizing chapter.

This work began with Chapter 2 by investigating the practical implementation of a maritime search and rescue mission employing a UAV. Our investigation focused on small fixed-wing drones and quadcopters to ensure the feasibility of our researched approaches for small SAR organizations operating on a tight budget. Additionally, we developed and published a software framework designed for mSAR operations. It enables preliminary detection of regions of interest onboard the drone, followed by more powerful object detectors on a ground station. This framework is able to stream the RoIs in high resolution while transmitting the rest of the image in lower quality, thereby addressing the challenge of limited bandwidth. Finally, we explored various RoI proposal methods under different bandwidth constraints. The key question was, how to utilize the limited bandwidth optimally to maximize detections.

Focusing on the computer vision aspects, Chapter 3 analyzed domain imbalances in various popular UAV object detection data sets. We found that many data sets are heavily imbalanced with respect to various environmental circumstances like altitude, viewing angle, and lighting conditions. We went on to show that these imbalances cause models to perform well in certain domains while failing in others, ultimately hindering performance. To mitigate these confounders, the chapter proposed a multi-domain learning approach that utilizes domain labels available from the UAV's sensors. We trained expert models on specific domains to enhance their domain-specific performance. Adaptively utilizing these specialized models on their corresponding domains led to an overall increase in performance across all domains.

Learning from the last chapter that the bird's eye view (BEV) domain is especially hard for object detectors, Chapter 4 introduced the Adaptive Resizer, a method designed to address scale variance in BEV imagery. This technique resizes images in a preprocessing step based on the UAV's capture altitude. This ensures that objects of the same class appear at a consistent scale across all images, thereby simplifying the detection task for the deep learning model. As a byproduct, the Adaptive Resizer is capable of trans-

ferring knowledge between different heights, allowing models trained on images from specific altitudes to generalize well to images captured at different altitudes. This alleviates the burden of domain imbalances, as discussed in Chapter 3, while also addressing the problem of small data sets for UAV computer vision, as discussed in Chapter 1.

To address these issues not only from the algorithmic perspective but also from the data set side, Chapter 5 introduced the SeaDronesSee data set. It features three tracks: single-object tracking, multi-object tracking, and object detection. With over $54,000$ annotated frames for the tracking track and over $5,500$ images for the object detection challenge featuring instances of humans and boats, this is the first UAV maritime computer vision data set of this scale. Furthermore, it is the first to include dense meta-data gathered from the UAV's various sensors. We introduced baseline models for all the tracks and provide an evaluation web server for researchers to test their models. Furthermore, we tested some of the meta-data-aware approaches from Chapters 3 and 4 on this data set and showed that these could enhance the results.

After focusing on the computer vision aspects in the last few chapters, Chapter 6 explores different trajectory planning techniques for UAVs in maritime search and rescue (mSAR) missions. In particular, we present a more effective path planning strategy based on the well-known branch-and-bound algorithm. It takes into account real-time meteorological data like wind flow and water currents, and uses this data to model the search targets' movement employing a particle filter. Compared to environmental-agnostic planners, this approach enhances performance. To test these approaches, we again developed and published a software framework that downloads the meteorological data, runs the path planning algorithm, and evaluates its performance using drift simulation of the search targets.

In conclusion, this dissertation presents various approaches for enhancing UAV-based maritime search and rescue missions. The research addressed challenges in object detection, path planning, and the technical implementation, both algorithmically, through the collection of data sets, and the development of software solutions. We achieved improvements in multiple aspects of mSAR operations. We emphasized the importance of integrating environmental data in both detection and trajectory planning.

Future work could explore the coordination of multiple UAVs in mSAR missions or develop ideas on how to integrate angle information into an adaptive detection model.

Bibliography

Adão, T., Hruška, J., Pádua, L., Bessa, J., Peres, E., Morais, R., and Sousa, J. J. (2017). Hyperspectral imaging: A review on uav-based sensors, data processing and applications for agriculture and forestry. *Remote Sensing*, **9**(11), 1110.

Airbus (2018). Airbus Ship Detection Challenge. `https://www.kaggle.com/c/airbus-ship-detection`. Accessed: 2021-03-01.

Albanese, A., Sciancalepore, V., and Costa-Pérez, X. (2020). Sardo: An automated search-and-rescue drone-based solution for victims localization. *arXiv preprint arXiv:2003.05819*.

Antonini, A., Guerra, W., Murali, V., Sayre-McCord, T., and Karaman, S. (2018). The blackbird dataset: A large-scale dataset for uav perception in aggressive flight. In *International Symposium on Experimental Robotics*, pages 130–139. Springer.

Bashmal, L., Bazi, Y., AlHichri, H., AlRahhal, M. M., Ammour, N., and Alajlan, N. (2018). Siamese-gan: Learning invariant representations for aerial vehicle image categorization. *Remote Sensing*, **10**(2), 351.

Berger, J., Lo, N., and Noel, M. (2013). Exact solution for search-and-rescue path planning. *International Journal of Computer and Communication Engineering*, **2**(3), 266.

Bergmann, P., Meinhardt, T., and Leal-Taixé, L. (2019). Tracking without bells and whistles. In *The IEEE International Conference on Computer Vision (ICCV)*.

Bhat, G., Danelljan, M., Gool, L. V., and Timofte, R. (2019). Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6182–6191.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. F. (2006). Optimal search for a lost target in a bayesian world. *Field and Service Robotics: Recent Advances in Reserch and Applications*, pages 209–222.

Bozcan, I. and Kayacan, E. (2020). Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8504–8510. IEEE.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Breivik, Ø. and Allen, A. A. (2008). An operational search and rescue model for the norwegian sea and the north sea. *Journal of Marine Systems*, **69**(1-2), 99–113.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Caruana, R. (1997). Multitask learning. *Machine learning*, **28**(1), 41–75.

CCG Manual (2023). Canadian Coast Guard Auxiliary Search & Rescue Crew Manual. `https://ccga-pacific.org/files/library/Chapter_9_Search.pdf`. Accessed: 2023-08-16.

Chattopadhay, A., Sarkar, A., Howlader, P., and Balasubramanian, V. N. (2018). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE.

Chen, Q., Wang, Y., Yang, T., Zhang, X., Cheng, J., and Sun, J. (2021). You only look one-level feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13039–13048.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., and Burgard, W. (2005). *Principles of robot motion: theory, algorithms, and implementations*. MIT press.

Clausen, J. (1999). Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30.

Coles-Brennan, C., Sulley, A., and Young, G. (2019). Management of digital eye strain. *Clinical and Experimental Optometry*, **102**.

Corbane, C., Najman, L., Pecoul, E., Demagistri, L., and Petit, M. (2010). A complete processing chain for ship detection using optical satellite imagery. *International Journal of Remote Sensing*, **31**(22), 5837–5854.

Council, V. (2016). Eyes overexposed: the digital device dilemma. *Alexandria, VA: The Vision Councile*.

Crisp, D. J. (2004). The state-of-the-art in ship detection in synthetic aperture radar imagery. *DSTO Information Sciences Laboratory Edinburgh, Australia*.

Dagestad, K.-F., Röhrs, J., Breivik, Ø., and Ådlandsvik, B. (2018). Opendrift v1. 0: a generic framework for trajectory modelling. *Geoscientific Model Development*, **11**(4), 1405–1420.

Danelljan, M., Bhat, G., Khan, F. S., and Felsberg, M. (2019). Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4660–4669.

Danelljan, M., Gool, L. V., and Timofte, R. (2020). Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7183–7192.

DarkLabel (2020). Darklabel video/image labeling and annotation tool. `https://github.com/darkpgmr/DarkLabel`. Accessed: 2021-01-11.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., Leal-Taixé, L., and Taixé, T. (2020). MOT20: A benchmark for multi object tracking in crowded scenes. Technical report.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, **29**(6), 141–142.

Ding, J., Xue, N., Long, Y., Xia, G.-S., and Lu, Q. (2018). Learning roi transformer for detecting oriented objects in aerial images. *arXiv preprint arXiv:1812.00155*.

Ditty, M., Karandikar, A., and Reed, D. (2018). Nvidia's xavier soc. In *Hot chips: a symposium on high performance chips*.

DJI (2018). Data Sheet DJI Matrice M210. `https://shop.solectric.de/media/pdf/5a/b4/41/spezifikation_matrice210v2.pdf`. Accessed: 2023-11-02.

Du, D., Qi, Y., Yu, H., Yang, Y., Duan, K., Li, G., Zhang, W., Huang, Q., and Tian, Q. (2018). The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 370–386.

Du, D., Zhu, P., Wen, L., Bian, X., Lin, H., Hu, Q., Peng, T., Zheng, J., Wang, X., Zhang, Y., *et al.* (2019). VisDrone-DET2019: The vision meets drone object detection in image challenge results. In *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 213–226. Institute of Electrical and Electronics Engineers Inc.

ElevonX (2023). Data Sheet ElevonX SkyEye. `https://www.elevonx.com/wp-content/uploads/2022/10/ElevonX.pdf`. Accessed: 2023-08-17.

Elfring, J., Torta, E., and van de Molengraft, R. (2021). Particle filters: A hands-on tutorial. *Sensors*, **21**(2), 438.

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., and Dehmer, M. (2020). An introductory review of deep learning for prediction models with big data. *Frontiers in Artificial Intelligence*, **3**.

Epstein, R. A. and Baker, C. I. (2019). Scene perception in the human brain. *Annual review of vision science*, **5**, 373–397.

Everingham et al., M. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, **111**(1), 98–136.

Fan, H., Du, D., Wen, L., Zhu, P., Hu, Q., Ling, H., Shah, M., Pan, J., Schumann, A., Dong, B., *et al.* (2020a). Visdrone-mot2020: The vision meets drone multiple object tracking challenge results. In *European Conference on Computer Vision*, pages 713–727. Springer.

Fan, H., Wen, L., Du, D., Zhu, P., Hu, Q., Ling, H., Shah, M., Wang, B., Dong, B., Yuan, D., *et al.* (2020b). Visdrone-sot2020: The vision meets drone single object tracking challenge results. In *European Conference on Computer Vision*, pages 728–749. Springer.

Farhadi, A. and Redmon, J. (2018). Yolov3: An incremental improvement. *Computer Vision and Pattern Recognition*.

Fonder, M. and Van Droogenbroeck, M. (2019). Mid-air: A multi-modal dataset for extremely low altitude drone flights. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, **2019-June**, 553–562.

Frontex (2024). Detections of illegal border-crossings statistics. `https://www.frontex.europa.eu/what-we-do/monitoring-and-risk-analysis/migratory-map/`. Accessed: 2024-05-21.

Gallego, A.-J., Pertusa, A., Gil, P., and Fisher, R. B. (2019). Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras. *Journal of Field Robotics*, **36**(4), 782–796.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, **32**(11), 1231–1237.

Geraldes, R., Goncalves, A., Lai, T., Villerabel, M., Deng, W., Salta, A., Nakayama, K., Matsuo, Y., and Prendinger, H. (2019). Uav-based situational awareness system using deep learning. *IEEE Access*, **7**, 122583–122594.

Ghazali, S. N. A. M., Anuar, H. A., Zakaria, S. N. A. S., and Yusoff, Z. (2016). Determining position of target subjects in maritime search and rescue (msar) operations using rotary wing unmanned aerial vehicles (uavs). In *2016 International Conference on Information and Communication Technology (ICICTM)*, pages 1–4. IEEE.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, **63**(11), 139–144.

Guoxiang, L. and Maofeng, L. (2010). Sargis: A gis-based decision-making support system for maritime search and rescue. In *2010 International Conference on E-Business and E-Government*, pages 1571–1574. IEEE.

Han, Y., Roig, G., Geiger, G., and Poggio, T. (2020). Scale and translation-invariance for novel objects in human vision. *Scientific reports*, **10**(1), 1–13.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Höfer, T., Kiefer, B., Messmer, M., and Zell, A. (2023). Hyperposepdf-hypernetworks predicting the probability distribution on so (3). In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2369–2379.

Hong, S., Kang, S., and Cho, D. (2019). Patch-level augmentation for object detection in aerial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0.

Hsieh, M.-R., Lin, Y.-L., and Hsu, W. H. (2017). Drone-based object counting by spatially regularized regional proposal network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4145–4153.

Huang, H., Huo, C., Wei, F., and Pan, C. (2019). Rotation and scale-invariant object detector for high resolution optical remote sensing images. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 1386–1389. IEEE.

HYCOM (2003). HYbrid Coordinate Ocean Model (HYCOM). `https://www.hycom.org/`. Accessed: 2023-08-29.

International Organization for Migration (Missing Migrants Project) (2024). Dead and missing by year. `https://missingmigrants.iom.int/region/mediterranean`. Accessed: 2024-05-14.

Iwamoto, M. M., Langenberger, F., and Ostrander, C. E. (2016). Ocean observing: serving stakeholders in the pacific islands. *Marine Technology Society Journal*, **50**(3), 47–54.

Jedrasiak, K., Bereska, D., and Nawrat, A. (2013). The prototype of gyro-stabilized uav gimbal for day-night surveillance. In *Advanced technologies for intelligent systems of national border security*, pages 107–115. Springer.

Jiang, P., Ergu, D., Liu, F., Cai, Y., and Ma, B. (2022). A review of yolo algorithm developments. *Procedia computer science*, **199**, 1066–1073.

Karaca, Y., Cicek, M., Tatli, O., Sahin, A., Pasli, S., Beser, M. F., and Turedi, S. (2018). The potential use of unmanned aircraft systems (drones) in mountain search and rescue operations. *The American journal of emergency medicine*, **36**(4), 583–588.

Kiefer, B. and Zell, A. (2023). Fast region of interest proposals on maritime uavs. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3317–3324.

Kiefer, B., Messmer, M., and Zell, A. (2021). Diminishing domain bias by leveraging domain labels in object detection on uavs. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 523–530. IEEE.

Kiefer, B., Žust, L., Kristan, M., Perš, J., Teršek, M., Wiliem, A., Messmer, M., Yang, C.-Y., Huang, H.-W., Jiang, Z., *et al.* (2024). 2nd workshop on maritime computer vision (macvi) 2024: Challenge results. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 869–891.

Kim et. al., S. (2020). Height-adaptive vehicle detection in aerial imagery using metadata of eo sensor. In *Automatic Target Recognition XXX*, volume 11394, page 1139404. International Society for Optics and Photonics.

Kokkinos, I. and Yuille, A. (2008). Scale invariance without scale selection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. (2018). The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE.

Kratzke, T. M., Stone, L. D., and Frost, J. R. (2010). Search and rescue optimal planning system. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, **25**.

Kulhavy, D. L., Hung, I., Unger, D., Zhang, Y., *et al.* (2017). Accuracy assessment on drone measured heights at different height levels. Accessed: 2024-04-09.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., *et al.* (2020). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, **128**(7), 1956–1981.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, **28**(3), 497–520.

Lee, H., Eum, S., and Kwon, H. (2019). ME r-cnn: Multi-expert r-cnn for object detection. *IEEE Transactions on Image Processing*, **29**, 1030–1044.

Lee, N., Ajanthan, T., and Torr, P. H. (2018). Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Li, J., Zhang, G., Jiang, C., and Zhang, W. (2023). A survey of maritime unmanned search system: Theory, applications and future directions. *Ocean Engineering*, **285**, 115359.

Li, Q., Mou, L., Liu, Q., Wang, Y., and Zhu, X. X. (2018). Hsf-net: Multiscale deep feature embedding for ship detection in optical remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, **56**(12), 7147–7161.

Li, S. and Yeung, D.-Y. (2017). Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Lin et. al., T.-Y. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Liu, S., Huang, D., and Wang, Y. (2019). Learning spatial fusion for single-shot object detection. *arXiv preprint arXiv:1911.09516*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*.

Lvsouras, E. and Gasteratos, A. (2020). A new method to combine detection and tracking algorithms for fast and accurate human localization in uav-based sar operations. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1688–1696. IEEE.

Lygouras, E., Santavas, N., Taitzoglou, A., Tarchanidis, K., Mitropoulos, A., and Gasteratos, A. (2019). Unsupervised human detection with an embedded vision system on a fully autonomous uav for search and rescue operations. *Sensors*, **19**(16), 3542.

Majdik, A. L., Till, C., and Scaramuzza, D. (2017). The zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, **36**(3), 269–273.

Martinez-Alpiste, I., Golcarenarenji, G., Wang, Q., and Alcaraz-Calero, J. M. (2021). Search and rescue operation using uavs: A case study. *Expert Systems with Applications*, **178**, 114937.

Mayer, S., Lischke, L., and Woźniak, P. W. (2019). Drones for search and rescue. In *1st International Workshop on Human-Drone Interaction*.

Messmer, M. and Zell, A. (2024). Evaluating uav path planning algorithms for realistic maritime search and rescue missions. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 472–479. IEEE.

Messmer, M., Kiefer, B., and Zell, A. (2022). Gaining scale invariance in uav bird's eye view object detection by adaptive resizing. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3588–3594. IEEE.

Messmer, M., Kiefer, B., Varga, L. A., and Zell, A. (2024). Uav-assisted maritime search and rescue: A holistic approach. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 272–280. IEEE.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

Mishra, B., Garg, D., Narang, P., and Mishra, V. (2020). Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, **156**, 1–10.

Morin, M., Lamontagne, L., Abi-Zeid, I., and Maupin, P. (2010). The ant search algorithm: An ant colony optimization algorithm for the optimal searcher path problem with visibility. In *Advances in Artificial Intelligence: 23rd Canadian Conference on Artificial Intelligence, Canadian AI 2010, Ottawa, Canada, May 31–June 2, 2010. Proceedings 23*, pages 196–207. Springer.

Mueller, M., Smith, N., and Ghanem, B. (2016). A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer.

Mundhenk, T. N., Konjevod, G., Sakla, W. A., and Boakye, K. (2016). A large contextual dataset for classification, detection and counting of cars with deep learning. In *European Conference on Computer Vision*, pages 785–800. Springer.

Nasr, I., Chekir, M., and Besbes, H. (2019). Shipwrecked victims localization and tracking using uavs. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 1344–1348. IEEE.

Ofli, F., Meier, P., Imran, M., Castillo, C., Tuia, D., Rey, N., Briant, J., Millet, P., Reinhard, F., Parkan, M., *et al.* (2016). Combining human computing and machine learning to make sense of big (aerial) data for disaster response. *Big data*, **4**(1), 47–59.

Oksuz, K., Cam, B. C., Kalkan, S., and Akbas, E. (2020). Imbalance problems in object detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Orin, N. (2020). NVIDIA Orin NX. `https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-orin/`. Accessed: 2024-01-28.

Papers with Code (2021). Object Detection on COCO test-dev. `https://paperswithcode.com/sota/object-detection-on-coco`. Accessed: 2021-03-01.

Papers with Code (2024a). COCO Benchmark Leaderbord. `https://paperswithcode.com/sota/real-time-object-detection-on-coco?dimension=FPS%20(V100%2C%20b%3D1)`. Accessed: 2024-01-27.

Papers with Code (2024b). Object Detection on COCO test-dev. `https://paperswithcode.com/area/computer-vision`. Accessed: 2024-05-22.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pei, Z., Qi, X., Zhang, Y., Ma, M., and Yang, Y.-H. (2019). Human trajectory prediction in crowded scene using social-affinity long short-term memory. *Pattern Recognition*, **93**, 273–282.

Perreault et. al., H. (2020). SpotNet: Self-Attention Multi-Task Network for Object Detection. pages 230–237. Institute of Electrical and Electronics Engineers Inc.

Prasad, D. K., Dong, H., Rajan, D., and Quek, C. (2019). Are object detection assessment criteria ready for maritime computer vision? *IEEE Transactions on Intelligent Transportation Systems*, **21**(12), 5295–5304.

Quantum, S. (2020a). Camera Overview Trinity F90+. `https://t6y2v7k3.rocketcdn.me/wp-content/uploads/2023/01/QS_Trinity_Overview_Cameras_V01_220711.pdf`. Accessed: 2023-11-02.

Quantum, S. (2020b). Data Sheet Trinity F90+. `https://quantum-systems.com/wp-content/uploads/2023/01/QS_TrinityF90_Overview_220912.pdf`. Accessed: 2023-08-20.

Queralta, J. P., Raitoharju, J., Gia, T. N., Passalis, N., and Westerlund, T. (2020). Autosos: Towards multi-uav systems supporting maritime search and rescue with lightweight ai and edge computing. *arXiv preprint arXiv:2005.03409*.

Raap, M., Zsifkovits, M., and Pickl, S. (2017). Trajectory optimization under kinematical constraints for moving target search. *Computers & Operations Research*, **88**, 324–331.

Raap, M., Preuß, M., and Meyer-Nieberg, S. (2019). Moving target search optimization–a literature review. *Computers & Operations Research*, **105**, 132–140.

Rajesh, R. and Kavitha, P. (2015). Camera gimbal stabilization using conventional pid controller and evolutionary algorithms. In *2015 International Conference on Computer, Communication and Control (IC4)*, pages 1–6. IEEE.

Redmon et. al., J. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, **39**(6), 1137–1149.

Riehl, J. R., Collins, G. E., and Hespanha, J. P. (2007). Cooperative graph-based model predictive search. In *2007 46th IEEE Conference on Decision and Control*, pages 2998–3004. IEEE.

Ringwald et. al., T. (2019). UAV-Net: A Fast Aerial Vehicle Detector for Mobile Platforms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 544–552. IEEE Computer Society.

Ristani, E., Solera, F., Zou, R., Cucchiara, R., and Tomasi, C. (2016). Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer.

Roberts, W., Griendling, K., Gray, A., and Mavris, D. (2016). Unmanned vehicle collaboration research environment for maritime search and rescue. In *30th Congress of the International Council of the Aeronautical Sciences*. International Council of the Aeronautical Sciences (ICAS) Bonn, Germany.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Russell, S. J. (2010). *Artificial intelligence: a modern approach*. Pearson Education, Inc.

San, K. T., Mun, S. J., Choe, Y. H., and Chang, Y. S. (2018). Uav delivery monitoring system. In *MATEC Web of Conferences*, volume 151, page 04011. EDP Sciences.

Sato, H. (2008). *Path optimization for single and multiple searchers: models and algorithms*. Ph.D. thesis, Citeseer.

Ševo, I. and Avramović, A. (2016). Convolutional neural network based automatic object detection on aerial images. *IEEE geoscience and remote sensing letters*, **13**(5), 740–744.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., *et al.* (2016). Mastering the game of go with deep neural networks and tree search. *nature*, **529**(7587), 484–489.

Singh, B. and Davis, L. S. (2018). An Analysis of Scale Invariance in Object Detection - SNIP. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3578–3587. IEEE Computer Society.

Sitemark (2020). Aerial data accuracy – an experiment comparing 4 drone approaches. `https://www.sitemark.com/blog/accuracy`. Accessed: 2021-01-11.

Sommer, L. W., Schuchert, T., and Beyerer, J. (2017). Fast deep vehicle detection in aerial images. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 311–319. IEEE.

Stabernack, B. and Steinert, F. (2021). Architecture of a low latency h. 264/avc video codec for robust ml based image classification. In *Workshop on Design and Architectures for Signal and Image Processing (14th edition)*, pages 1–9.

Sullivan, J. M. (2008). Visual fatigue and the driver. Technical report, University of Michigan, Ann Arbor, Transportation Research Institute.

Suzuki, S. *et al.* (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, **30**(1), 32–46.

Taha, H. A. (2007). *Operations research: an introduction*. Pearson Prentice Hall.

Tan, M. and Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *36th International Conference on Machine Learning*, pages 10691–10700.

Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790.

Tian et. al., Z. (2019). FCOS: Fully convolutional one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9626–9635.

Tiemann, J., Feldmeier, O., and Wietfeld, C. (2018). Supporting maritime search and rescue missions through uas-based wireless localization. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE.

UN Trade and Development (2023). World seaborne trade by types of cargo and by group of economies, annual. `https://unctadstat.unctad.org/datacentre/dataviewer/US.SeaborneTrade`. Accessed: 2024-05-21.

Unel et. al., O. (2019). The power of tiling for small object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.

van Gemert, J. C., Verschoor, C. R., Mettes, P., Epema, K., Koh, L. P., and Wich, S. (2014). Nature conservation drones for automatic localization and counting of animals. In *European Conference on Computer Vision*, pages 255–270. Springer.

Vanholder, H. (2016). Efficient inference with tensorrt. In *GPU Technology Conference*, volume 1.

Varga, L. A. and Zell, A. (2021). Tackling the background bias in sparse object detection via cropped windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2768–2777.

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

Varga, L. A., Messmer, M., Benbarka, N., and Zell, A. (2023). Wavelength-aware 2d convolutions for hyperspectral imaging. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3788–3797.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, **30**.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475.

Wang, X., Cai, Z., Gao, D., and Vasconcelos, N. (2019). Towards universal object detection by domain attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7289–7298.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, **3**, 1–40.

Wu, J., Cheng, L., and Chu, S. (2023). Modeling the leeway drift characteristics of persons-in-water at a sea-area scale in the seas of china. *Ocean engineering*, **270**, 113444.

Wu, Z., Suresh, K., Narayanan, P., Xu, H., Kwon, H., and Wang, Z. (2019). Delving into robust object detection from unmanned aerial vehicles: A deep nuisance disentanglement approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1201–1210.

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., and Zhang, L. (2018). Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Yang, F., Fan, H., Chu, P., Blasch, E., and Ling, H. (2019). Clustered object detection in aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8311–8320.

Yeong, S., King, L., and Dol, S. (2015). A review on marine search and rescue operations using unmanned aerial vehicles. *International Journal of Marine and Environmental Sciences*, **9**(2), 396–399.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

Yu, F., Wang, D., Shelhamer, E., and Darrell, T. (2018). Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412.

Yu et al., H. (2020). The Unmanned Aerial Vehicle Benchmark: Object Detection, Tracking and Baseline. *International Journal of Computer Vision*, **128**(5), 1141–1159.

Zhang, S., Chi, C., Yao, Y., Lei, Z., and Li, S. Z. (2020a). Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9756–9765.

Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2020b). Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv e-prints*, pages arXiv–2004.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, **30**(11), 3212–3232.

Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as Points.

Zhu, P., Wen, L., Du, D., Bian, X., Ling, H., Hu, Q., Nie, Q., Cheng, H., Liu, C., Liu, X., *et al.* (2018). Visdrone-det2018: The vision meets drone object detection in image challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Zhu, P., Wen, L., Du, D., Bian, X., Hu, Q., and Ling, H. (2020a). Vision meets drones: Past, present and future. *arXiv preprint arXiv:2001.06303*.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020b). Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.

Zhu et. al., P. (2018). Vision Meets Drones: A Challenge. *arXiv*.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, **109**(1), 43–76.

Zimmermann, F., Eling, C., Klingbeil, L., and Kuhlmann, H. (2017). Precise positioning of uavs-dealing with challenging rtk-gps measurement conditions during automated uav flights. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, **4**.