Leveraging Metadata for Computer Vision on Unmanned Aerial Vehicles

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

vorgelegt von
M.Sc. Benjamin Kiefer
aus Denzlingen

Tübingen
2023

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

Tag der mündlichen Qualifikation:	07.11.2023
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	Prof. Dr. Andreas Zell
2. Berichterstatter/-in:	Prof. Dr. Andreas Schilling

Dedicated to the person reading this. Good choice! Only a few more pages.

Abstract

The integration of computer vision technology into Unmanned Aerial Vehicles (UAVs) has become increasingly crucial in various aerial vision-based applications. Despite the great significant success of generic computer vision methods, a considerable performance drop is observed when applied to the UAV domain. This is due to large variations in imaging conditions, such as varying altitudes, dynamically changing viewing angles, and varying capture times resulting in vast changes in lighting conditions. Furthermore, the need for real-time algorithms and the hardware constraints pose specific problems that require special attention in the development of computer vision algorithms for UAVs.

In this dissertation, we demonstrate that domain knowledge in the form of meta data is a valuable source of information and thus propose domain-aware computer vision methods by using freely accessible sensor data. The pipeline for computer vision systems on UAVs is discussed, from data mission planning, data acquisition, labeling and curation, to the construction of publicly available benchmarks and leaderboards and the establishment of a wide range of baseline algorithms. Throughout, the focus is on a holistic view of the problems and opportunities in UAV-based computer vision, and the aim is to bridge the gap between purely software-based computer vision algorithms and environmentally aware robotic platforms.

The results demonstrate that incorporating meta data obtained from onboard sensors, such as GPS, barometers, and inertial measurement units, can significantly improve the robustness and interpretability of computer vision models in the UAV domain. This leads to more trustworthy models that can overcome challenges such as domain bias, altitude variance, synthetic data inefficiency, and enhance perception through environmental awareness in temporal scenarios, such as video object detection, tracking and video anomaly detection.

The proposed methods and benchmarks provide a foundation for future research in this area, and the results suggest promising directions for developing environmentally aware robotic platforms. Overall, this work highlights the potential of combining computer vision and robotics to tackle real-world challenges and opens up new avenues for interdisciplinary research.

Kurzfassung

Für die weitreichende Verbreitung und Nutzung von Drohnen sind Algorithmen für Maschinelles Sehen unabdingbar. Viele der Fortschritte zu Algorithmen zum Maschinellen Sehen lassen sich nicht direkt auf den Drohnen-Bereich übertragen, da dieser besondere Herausforderungen in den zugehörigen Bilddaten darstellt, wie zum Beispiel eine große Höhenvarianz und die daraus resultierenden unterschiedlich aufgelösten Bilder mit Objekten verschiedener Größe, dynamische Veränderungen in den Aufnahmewinkeln und verschiedene Aufnahmezeiten, welche dynamischen Lichtveränderungen zur Folge haben. Desweiteren sind Hardware-Beschränkungen und Echtzeit-Anforderungen weitere große Herausforderungen.

In dieser Dissertation zeigen wir, dass Wissen über den Kontext in Form von Metadaten eine wertvolle Informationsquelle ist und schlagen Methodes des Machinelles Sehens vor, welche den Kontext berücksichtigen, indem sie auf frei zugängliche Sensordaten zugreifen. Dahingehend beschreiben wir viele Aspekte von Pipelines für Computer-Vision-Systeme auf UAVs, von der Datenmissionsplanung, Datenerfassung, -annotierung und -kuration bis hin zur Konstruktion von öffentlich verfügbaren Benchmarks und Leaderboards und dem Aufbau einer breiten Palette an Basis-Algorithmen. Dabei steht stets ein holistischer Blick auf die Probleme und Chancen der UAV-basierten Computer-Vision im Fokus und das Ziel ist es, die Lücke zwischen rein softwarebasierten Computer-Vision-Algorithmen und kontext-bewussten Roboter-Plattformen zu schließen.

Die Ergebnisse zeigen, dass die Einbeziehung von Metadaten von Sensoren, wie z.B. GPS, Barometern und Inertial-Messsystemen die Robustheit und Interpretierbarkeit von Computer-Vision-Modellen im UAV-Bereich erheblich verbessern kann. Dies führt zu zuverlässigeren Modellen, welche die Herausforderungen wie Domänen-Bias, Höhenvariation und (In-)effizienz synthetischer Daten überwinden und ein besseres Verständnis von ihrer Umwelt bekommen.

Zusammenfassend zeigt diese Dissertation die Bedeutung des Nutzens von Domänenwissen in Form von Metadaten in UAV-basiertes Maschinelles Sehen auf und schlägt Methoden vor, die Sensordaten nutzen, um die Wahrnehmung zu verbessern und Herausforderungen wie Domänenbias und Höhenvarianz zu überwinden. Die vorgeschlagenen Methoden und Benchmarks bilden eine Grundlage für zukünftige Forschung in diesem Bereich, und die Ergebnisse deuten auf vielversprechende Ansätze zur Entwicklung von intelligenten Robotik-Plattformen hin. Insgesamt verdeutlicht diese Arbeit das Potenzial der Kombination von Maschinellem Sehen und Robotik zur Bewältigung realer Herausforderungen und eröffnet neue Möglichkeiten für interdisziplinäre Forschung.

Acknowledgments

I would first like to thank my thesis advisor Prof. Dr. Zell for giving me the opportunity to write this thesis in the Cognitive Systems Group and the continuous support he provided. Furthermore, I would like to thank the Cognitive Systems Group for the welcoming atmosphere and the diverse and helpful discussions and conversations throughout the entire time. In particular, I would like to thank Martin Meßmer for the engaging discussions, Timon Höfer for the helpful feedback and Leon Varga for his critical opinions.

Contents

Chapter 1

Introduction

The vast field of artificial intelligence has improved drastically over the last few years. Continuously improving hardware along with major scientific advancements contributed to the fast development and use of artificial intelligent methods. In particular, the field of data-driven models, i.e. machine learning models, has seen a great ascent.

The development in hardware and scientific advancements has greatly benefited computer vision methods. Early algorithms relied on manual construction of feature kernels for most tasks in machine vision applications (Zhou *et al.* (2009)), but today's methods predominantly rely on data-driven neural networks or related machine learning models (Voulodimos *et al.* (2018)). However, most computer vision research has focused on standard benchmarks in generic scenarios, such as the ILSVRC image classification benchmark, ImageNet (Russakovsky *et al.* (2015)), and the object detection benchmark Common Objects in Context (COCO) (Lin *et al.* (2014)). While there has been significant transfer from the generic to the aerial domain, the latter is significantly different due to various reasons, such as different viewpoints with typically smaller objects, the need for embedded algorithms, and real-time methods, which require special considerations to develop robust models.

The UAV-based domain poses many challenges, but it also presents the concept of an intelligent agent perceiving the world through its onboard sensors. By treating the problem of UAV-based computer vision holistically, the aerial agent (i.e., the UAV) can achieve a more sophisticated understanding of its surrounding environment. Inspiration for this approach can be drawn from the way humans understand their surroundings. For instance, when searching for people from a helicopter, humans unconsciously impose a prior on many aspects, such as object size, visual appearance, and motion type. This internal prior enables efficient and accurate searches.

Taking inspiration from human environmental awareness, this work seeks to enhance UAV-based computer vision methods with a notion of environmental intelligence. This intelligence is obtained through onboard sensors such as GPS sensors, barometers, and inertial measurement units. By analyzing this data, we demonstrate how intelligent computer vision methods can be developed for various applications and tasks. Chapter 1 provides a brief overview of general and UAV-based computer vision and explains the importance of metadata. The chapter concludes with an outline of the entire work.

1.1 General Computer Vision

The field of computer vision is a branch of artificial intelligence that is concerned with the development of algorithms and sensors to enable computers to interpret and understand visual data captured from the environment. This technology has a broad range of applications, including object detection and recognition, image processing, and robotics, and it has the potential to greatly increase the efficiency and intelligence of various systems.

As an ever-evolving field, computer vision continues to grow and develop, with new advancements and applications constantly emerging. Its transformative potential is far-reaching, and it has already made significant strides in domains such as healthcare, transportation, surveillance and retail (O'Mahony *et al.* (2020)).

Computer Vision has a long history, dating back as far as 1957 with Gilbert Hobrough demonstrating an analog implementation of stereo image correlation (Blikharskyy *et al.* (2022)) [1]. Today, computer vision continues to be an active area of research and development, with new advancements and applications being developed constantly. Some of the most recent driving forces for the advancement in computer vision include:

- The development of convolutional neural networks, which are a type of machine learning algorithm that has proven effective for many computer vision tasks, such as image classification and object detection.

- The development of deep learning, which is a subset of machine learning that involves the use of multiple layers of interconnected neurons to analyze data and make predictions.

- The availability of large amounts of data and powerful computing resources, which have enabled the development of more sophisticated and accurate computer vision algorithms.

- The development of new sensors, such as lidars, which can provide more detailed and accurate data about the environment for computer vision algorithms to analyze.

- The widespread adoption of computer vision technology in many industries, such as healthcare, transportation, and retail, which has led to the development of many new applications and use cases.

- The emergence of new fields, such as augmented reality and robotics, which have driven significant advancements in computer vision research and development.

[1]For a comprehensive computer vision history please see Ikeuchi (2021)

1.2 Specific UAV challenges

Unmanned Aerial Vehicles (UAVs) employ Computer Vision methods for perceiving and comprehending their environment through advanced sensors and algorithms. This allows UAVs to accomplish various tasks, such as object detection, tracking, and image recognition.

Computer vision on UAVs involves the use of cameras and lidars, among other sensors, to capture images and data about the surrounding environment. These data are then analyzed by algorithms, such as convolutional neural networks, to identify relevant objects and information. The visual data is processed either through embedded devices or by streaming to a ground station.

Applications of computer vision on UAVs include navigation, surveillance, delivery services, and search and rescue operations, among others (Cazzato *et al.* (2020)). By enabling UAVs to "see" and understand their environment, computer vision technology can enhance the autonomy and effectiveness of drones in various scenarios.

However, there are several challenges associated with using computer vision on UAVs, including:

- Limited computing power: UAVs are typically small and lightweight, which means they have limited space and power available for computing equipment. This can make it difficult to run complex computer vision algorithms in real-time.

- Variable lighting and weather conditions: The lighting and weather conditions in a UAV's environment can vary significantly, which can make it difficult for algorithms to accurately identify objects.

- Small object sizes: In many applications, the UAV flies high so that perceived objects are small and hard to distinguish from background.

- Variable altitudes and viewing angles: UAVs fly in diverse altitudes and, in modern UAVs, gimbal-based cameras allow for a multitude of different camera tilts and rolls on top of the UAV-induced principle axes changes.

- Moving objects: UAVs are often used in situations where objects are moving, such as in surveillance or delivery applications. This can make it difficult for algorithms to track objects accurately.

These challenges continue to make deployment of generic computer vision algorithms to the UAV domain difficult. However, the UAV-based domain also allowed for casting many problems in a more holistic setting: By incorporating additional information from the UAVs' onboard sensors, we may tackle these problems in a way that allows for more robust, interpretable and trustworthy models.

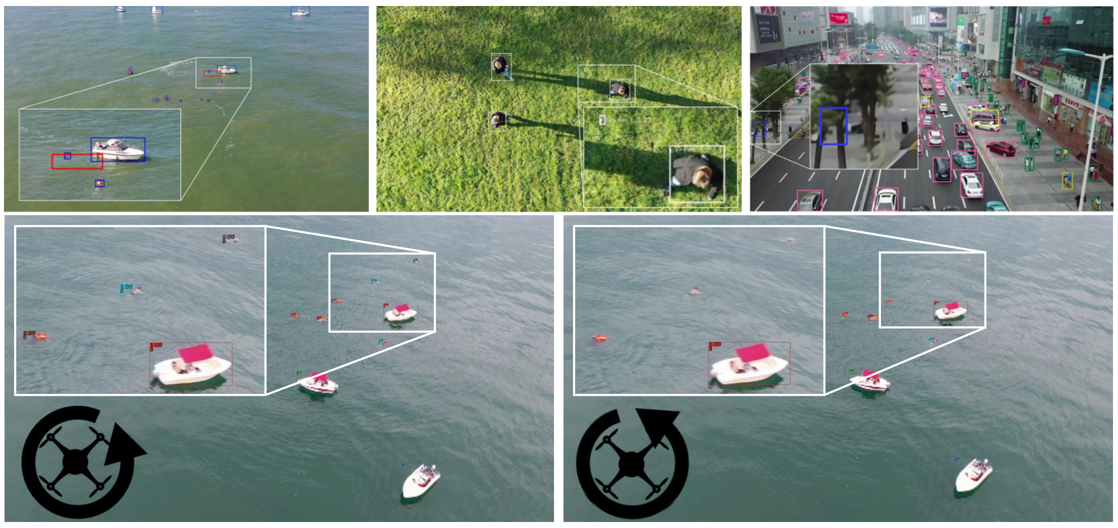


Figure 1.1: Examples of wrong detections by metadata-agnostic object detectors and trackers. The object detectors confidently predict a large swimmer (top left), a tiny person (top center) and a small bus (top right). A tracker fails to track three swimmers from one frame (bottom left) to the next (bottom right) due to a camera heading movement. This could easily be avoided by considering the metadata information onboard the UAV.

1.3 Potential of Metadata

A modern UAV is equipped with a multitude of sensors, such as IMUs, GPS sensors, altitude and heading reference systems, barometers, clocks and more. While these sensors are primarily used for stable control and autopilot of the UAV, they tell us much about the environment, which can be valuable in downstream vision applications. For example, the altitude information tells us about the size of objects. The camera tilt angle allows us to reason about the size distribution for different areas in the image space. Generally, the underlying 3D geometry in conjunction with the camera intrinsics and extrinsics allow for detailed priors on object sizes and appearances. Moreover, by accounting for the movement of the UAV and its camera, we may create a 3D world representation of our environment, which we may leverage in 2D image space to reason about object locations.

In recent years, there has been a growing trend towards self-sufficient models, which require minimal external information (Belmonte *et al.* (2019)). However, we would like to challenge this trend. The domain-agnostic techniques often fail to differentiate between certain types of errors and may exhibit overconfidence in rare or previously unseen circumstances, e.g. see Figure 1.1. Metadata can help ensure certain prediction quality guarantees, resulting in more robust and trustworthy models.

Subsequent chapters show how to include such metadata to help data-driven models attain more robust representations and prediction systems make fewer errors.

1.4 Contribution and Outline

This dissertation provides a comprehensive overview of the topic of UAV-based computer vision using metadata, based on seven first-author publications at high-ranked computer vision and robotics conferences, listed in order of coverage (”*” contributed equally):

1. Leon 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.

2. Benjamin Kiefer et al. ”**1st Workshop on Maritime Computer Vision (MaCVi) 2023: Challenge Results**”. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops 2023.

3. 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.

4. 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.

5. Benjamin Kiefer, David Ott, Andreas Zell. ”**Leveraging Synthetic Data in Object Detection on Unmanned Aerial Vehicles**.” In Proceedings of the IEEE 26th International Conference on Pattern Recognition (ICPR) 2022.

6. Benjamin Kiefer, Andreas Zell. ”**Fast Region of Interest Proposals on Maritime UAVs**”. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2023.

7. Benjamin Kiefer, Yitong Quan, Andreas Zell. ”**Memory Maps for Video Object Detection and Tracking on UAVs**”. Under review for the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2023.

Further, there are two works, which were done during the doctorate program, but will not be covered in this dissertation as they are thematically less relevant.

1. Timon Höfer, Benjamin Kiefer and Andreas Zell. ”**HyperPosePDF: Prediciting the Probability Distribution on SO(3)**.” In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2023.

2. Pablo Ruiz-Ponce, David Ortiz-Perez, Jose Garcia-Rodriguez and Benjamin Kiefer. ”**POSEIDON: A Data Augmentation Tool for Small Object Detection Datasets in Maritime Environments**.” Accepted for Sensors Journal in issue ”Sensor-Based Object Detection and Recognition in Intelligent Surveillance Systems” (2023).

This work is structured as follows:

Chapter 2	First, the acquisition and generation of metadata with corresponding imagery data is discussed. We describe the data generation process from the acquisition campaign planning to the pre- and post-processing and annotation phase. In particular, we discuss the collection of PeopleOnGrass and SeaDronesSee. Lastly, we describe a benchmark webserver that is open for researchers.
Chapter 3	Here, we lay the foundation of multiple baseline methods and discuss metadata-agnostic state-of-the-art methods. Furthermore, we summarize the SeaDronesSee Object Detection v2 and Multi-Object Tracking Challenge Results as part of the 1st Workshop on Maritime Computer Vision. These aimed at establishing and advancing the state-of-the art on this newly captured benchmark.
Chapter 4	The subsequent chapters are a collection of *metadata-aware* methods, each discussing techniques to construct environmentally aware computer vision algorithms. In particular, this chapter discusses a method to leverage metadata by tackling the problem of domain bias. We construct domain-aware expert models to enhance the performance on individual domains and across all domains.
Chapter 5	Here, we present the specific challenge of large altitude variances in object detection on UAVs. We propose a method to introduce scale invariance in bird's eye view imagery.
Chapter 6	This chapter is devoted to synthetic data. We modify and improve a synthetic data capturing engine built on the popular video game GTAV. We demonstrate how to use synthetic data for object detection on UAV and, importantly, show how the use of metadata improves the data efficiency of synthetically generated data.
Chapter 7	Here, we discuss how metadata helps in constructing a weakly supervised anomaly detector for maritime search and rescue applications. We analyze its use in embedded scenarios and show how metadata can help in efficiently training and deploying this model.
Chapter 8	Finally, here we take a closer look at the temporal domain and discusses how we can use metadata to obtain more robust predictions by aggregating information in 3D space. We show that we can improve video object detection and video anomaly detection by considering the underlying 3D geometry.
Chapter 9	We provide a conclusion, discussion and outlook, reviewing the techniques for constructing meta-data aware methods and how to proceed from the current state.

Next, we will discuss the acquisition and generation of multi-modal datasets featuring metadata.

Chapter 2

Generating Image Data and Metadata

Looking at the literature, most of the methods tackling computer vision applications on UAVs ignore metadata (Kanellakis and Nikolakopoulos (2017)), which may be due to them aiming at proposing domain-agnostic methods that do not have to rely on such data. However, it may very well also be due to the lack of benchmarks to test their methods on. Common large-scale benchmarks for computer vision on UAVs solely provide the imagery, but do not capture the environmental capturing conditions as measured by other onboard sensors. While there are benchmarks that capture some of the metadata, they only focus on certain aspects of environmental conditions or only provide a very coarse categorization into certain capturing domains.

Therefore, this chapter discusses the generation and acquisition of data that is accompanied with the environmental capturing condition. Depending on the context, we use the terms *metadata*, *domain data*, *capturing conditions* and *environmental data* interchangeably.

Importantly, metadata would be useless if it was not accompanied by meaningful image data. Therefore, large parts of this chapter are also concerned with the image content and semantics and not just the metadata.

This chapter is divided into two sub-chapters, each discussing the acquisition, generation and labeling of a dataset in a different environment: A simple grass-environment (PeopleOnGrass) and a maritime environment (SeaDronesSee). Before going into the generation of these datasets, we review the literature regarding datasets featuring metadata and, last but not least, define the term metadata.

2.1 What is Metadata?

Metadata is information that describes other data. In the context of computer vision on UAVs, metadata might include information about the location and orientation of the UAV when an image was captured, the type of camera that was used, and any other relevant information that can help interpret the images. This metadata can be used to help identify and analyze objects in the images, as well as to help understand the context in which the images were captured.

Desired variables	Sensor	Units	Accuracy
Altitudes	Barometer	meters	$1-5m$
Angular velocities & acceleration	IMU	degrees/second	$0.1-1^\circ$
Camera tilt angles	Gimbal IMU	degrees	$0.1-1^\circ$
Geographic coordinates	GNSS receiver	degrees	$0.01-5m$
Time	GNSS receiver	ms	$0-30ns$

Table 2.1: Overview of used sensors on UAVs and their properties. Please note that these values only apply for our experiments and depending on the used sensors, they may vary.

In this work, we will consider several types of metadata coming from various sensors. In the following, we will shortly review the metadata we consider and the underlying sensors that are used to acquire this data. As this work focuses on the computer vision aspect, we would like to note that the discussion around the physical aspects of these sensors is limited and we refer to other works that discuss physical properties and their implications (e.g. Balestrieri *et al.* (2021)). In the following, we will shortly discuss the different sensors. Please also see Table 2.1 for a concise overview. Lastly, note that for our data acquisition, we always calibrated the sensors via the manufactures' calibration protocols. This included the IMU, compass and gimbal sensors.

2.1.1 Altitudes

Altitude is a measure of the vertical distance of an object or a point in reference to a specific reference point, usually the sea level or the ground level. In the context of UAVs, altitude is an important factor that determines the performance and capabilities of the UAV.

Altitude can be measured using various methods and instruments, such as barometers, altimeters, or GPS systems. Barometers measure the air pressure at a given altitude, while altimeters use this information to determine the altitude of the UAV, which is why we equate these sensors in our discussion. GPS systems, on the other hand, use satellite signals to calculate the altitude of the UAV with respect to the sea level.

Altimeters are typically smaller, lighter, and less expensive than GPS systems, and they do not require a clear view of the sky to function properly. They can be affected by factors such as air temperature and humidity. GPS systems on the other hand are larger, heavier, and more expensive than altimeters, and they can be affected by factors such as satellite availability and signal interference.

The altitude measurement is a very informative indicator as it tells use about the sizes of objects. We will see in subsequent chapters that this can help significantly in downstream object detection.

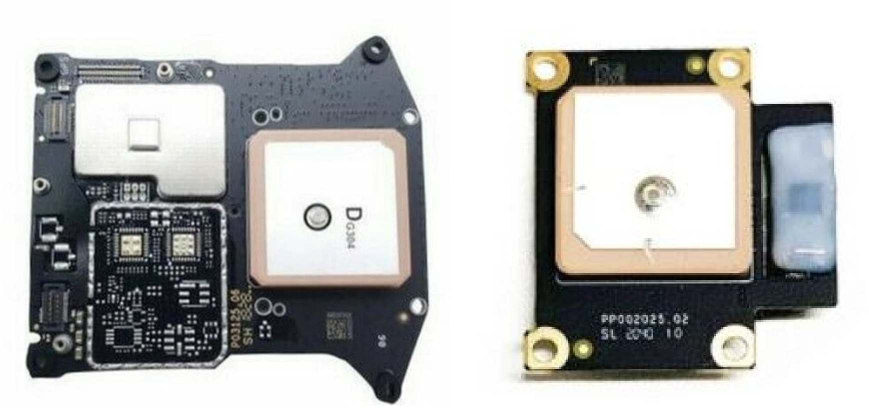


Figure 2.1: GNSS and IMU module (left) and barometer (right) of one of the UAVs we used for our experiments (DJI Mavic 2 Pro (left) and DJI Mini (right)).

2.1.2 Angular velocities

Angular velocity is a measure of the rate of change of an object's orientation in space. It is typically measured in units of degrees per second (deg/s) or radians per second (rad/s), and is represented by a vector quantity with a magnitude and direction. In the context of UAVs, angular velocity is an important parameter that is used to determine the orientation and stability of the vehicle. UAVs typically have sensors, such as gyroscopes or accelerometers, that can measure the angular velocity of the vehicle in real time. This information is then used by the UAV's control system to maintain a stable flight attitude and respond to changes in the vehicle's orientation. Angular velocity is also an important parameter for UAVs that are equipped with gimbal-mounted cameras, as it allows the gimbal to maintain a stable orientation and capture smooth, stable footage.

In our context, we make use of the gimbal angle to know the viewing angle of the scene. However, the other measurements are also important to subtract out these movements in the corresponding footage, e.g. to compute the position and orientation of the horizon in the image plane.

2.1.3 Geographic Coordinates

Geographic coordinates, also known as latitude and longitude, are used to precisely identify a location on the Earth's surface. These coordinates are used in navigation systems used by UAVs to determine the position of the UAV in relation to the Earth's surface. Latitude is a measure of the angle between a specific point on the Earth's surface and the equatorial plane, while longitude is a measure of the angle between a specific point on the Earth's surface and the Prime Meridian. Together, these two coordinates can be used to uniquely identify any location on the Earth's surface. UAVs typically use GPS or other satellite-based navigation systems (GNSS) to determine their geographical coordinates, which can then be used for navigation and mapping purposes. There are many different GNSS receivers, with newer models being as precise as below a centimeter. However, current commonly used models rather have an accuracy of up to five meters. Furthermore, this depends on many other factors, including the number and position of satellites in the sky, atmospheric conditions, and interference from objects such as tall buildings or mountains. Depending on the sensor and the software, the measurements are accompanied with uncertainties, i.e. measurement about variances, allowing for downstream justifications based on these inaccuracies.

GNSS measurements also contain time stamps that yield accurate timestamp, which may be used to determine lighting conditions or for a reliable variable to use for aligning all kinds of measurements. We note that we shortly discuss the altitude measurements in a subsection below.

For our purposes, we use geographic coordinates to obtain an understanding of where we perceived objects which helps in building 3D representations of scenes as will be seen later.

2.2 Labeled Data Sets Captured from UAVs

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.* (2018b)) and UAVDT (Du *et al.* (2018)). Both data sets focus on object detection and object tracking in unconstrained environments.Pei *et al.* (2019) 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. Both, Hsieh *et al.* (2017) and Mundhenk *et al.* (2016) capture a data set showing parking lots for car counting tasks and constrained object detection. Li and Yeung (2017) provide a single-object tracking data set showing traffic, wild life and sports scenarios. Krajewski *et al.* (2018) show vehicles on freeways.

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

2.2.1 Multi-Modal Data Sets Captured from UAVs

UAVDT (Du *et al.* (2018)) provides coarse metadata 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). Wu *et al.* (2019) manually label VisDrone after its release with the same annotation information

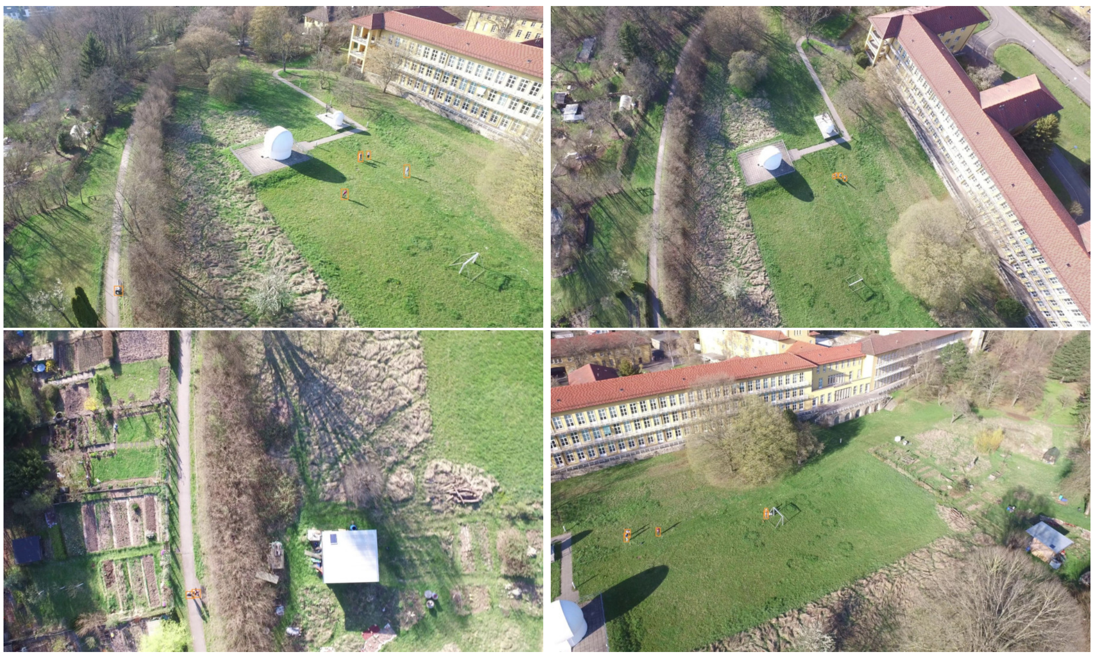


Figure 2.2: Sample images of PeopleOnGrass (POG). Altitudes and viewing angles from top left to bottom right: 50m, 45°; 100m, 85°, 60m, 90°; 40m, 40°.

for the object detection track. Mid-Air (Fonder and 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 metadata are captured to benchmark appearance-based localization. Bozcan and Kayacan (2020) release a low-altitude (< 30 m) object detection data set containing images showing a traffic circle and provide metadata such as altitude, GPS, and velocity but exclude the import camera angle information.

Tracking data sets often provide metadata (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 metadata. The same observation can be made for the tracking track of VisDrone (Fan *et al.* (2020b)). See Table 2.2 for an overview of annotated aerial data sets.

OD	Env.	Plat.	widths	Alt.	Range	Ang.	Range	O.
DOTA	cities	sat.	20,000	–	–	×	90°	×
UAVDT	traffic	UAV	1,024	×	5-200 m*	×	$0-90°$*	×
VisDrone	traffic	UAV	2,000	×	5-200 m*	×	$0-90°$*	×
Airbus Ship	marine	sat.	768	–	–	×	90°	×
AU-AIR	traffic	UAV	1,920	✓	5-30 m	×	$45-90°$	✓
POG	grass	UAV	3,840	✓	5-120 m	✓	$0-90°$	✓
SeaDronesSee	marine	UAV	5,456	✓	5-260 m	✓	$0-90°$	✓

SOT	Env.	#Vids	widths	Alt.	Range	Ang.	Range	O.
UAV123	traffic	123	1,280	×	5-50 m*	×	$0-90°$*	✓
DTB70	sports	70	1,280	×	0-10 m*	×	$0-90°$*	×
UAVDT-SOT	traffic	50	1,024	×	5-200 m*	×	$0-90°$*	✓
VisDrone	traffic	167	2,000	×	5-200 m*	×	$0-90°$*	✓
POG	grass	59	3,840	✓	5-120 m	✓	$0-90°$	✓
SeaDronesSee	marine	208	3,840	✓	5-150 m	✓	$0-90°$	✓

MOT	Env.	#F.	widths	Alt.	Range	Ang.	Range	O.
UAVDT-MOT	traffic	40.7 k	1,024	×	5-200 m*	×	$0-90°$*	✓
VisDrone	traffic	40 k	2,000	×	5-200 m*	×	$0-90°$*	✓
POG	grass	37 k	3,840	✓	5-150 m	✓	$0-90°$	✓
SeaDronesSee	marine	54 k	3,840	✓	5-150 m	✓	$0-90°$	✓

Table 2.2: Comparison with the most prominent annotated aerial data sets in Object Detection (OD), Single-Object Tracking (SOT) and Multi-Object Tracking (MOT). Altitude (Alt.) and angle (Ang.) indicate whether or not there are precise altitude and angle view information available. Other (O.) refers to time stamps, GPS, and IMU data and in the case of object tracking can also mean attribute information about the sequences. The environment (Env.) and platform (plat.) is indicated as well. 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.

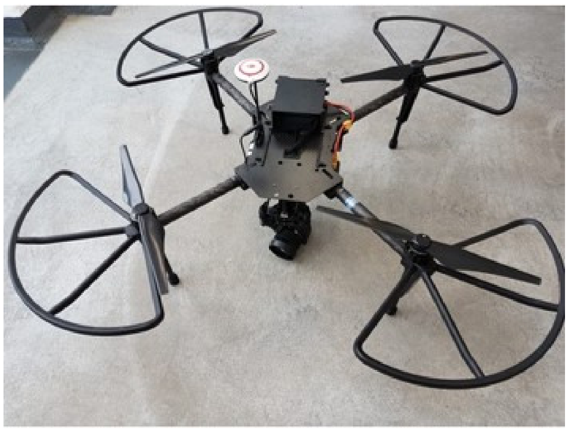


Figure 2.3: DJI Matrice 100 with Zenmuse X5 used for capturing the POG data.

2.3 People On Grass (POG)

First, we record the data set PeopleOnGrass (POG) depicting people walking on grass. We use a DJI Matrice 100 equipped with a Zenmuse X5 for capturing the data. The metadata is obtained through DJI's onboard software developing kit and by using Airdata (App-Airdata (2022)). We flew the UAV manually, trying to capture as many people as possible. 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 capture an object detection and video object detection dataset, respectively.

2.3.1 Object Detection

The object detection dataset contains 2,9k images (3840x2160 pixels resolution), showing people from various angles and altitudes varying from 0° (horizontally facing) to 90° (top-down) and 4m to 103m, respectively, each labeled with the precise altitude and angle it was captured at. See Figure 2.4 for a distribution of objects. Further metadata, such as GPS location, UAV speed and rotation, timestamps and others are also included.

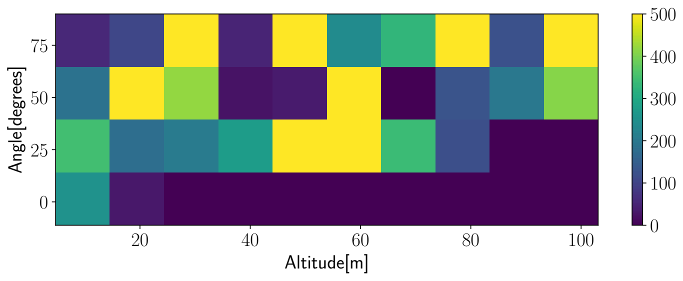


Figure 2.4: Distribution of objects in PeopleOnGrass (POG) across different levels of altitude and camera pitch angles. For visualization purposes only a 4x10 grid is shown.

See Figure 4.1 for example images. We manually and carefully annotate 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.

2.3.2 Video Object Detection

Also for this dataset, we used a Zenmuse X5 camera mounted on a DJI Matrice 100 for collection. We collected 10,633 frames in 30 fps videos with 3840x2160 resolution. We annotated 48,802 instances of people using DarkLabel (Darkpgmr (2022)). We made sure to have a large variance w.r.t. to altitude ($h = 10m$-$100m$) and gimbal pitch angle ($\beta = 17°$-$90°$).

2.4 SeaDronesSee

Unmanned Aerial Vehicles (UAVs) equipped with cameras have grown to an important asset in a wide range of fields, such as agriculture, delivery, surveillance, and search and rescue (SaR) missions (Adão *et al.* (2017); San *et al.* (2018); Geraldes *et al.* (2019)). In particular, UAVs are capable of assisting in SaR missions due to their fast and versatile applicability while providing an overview over the scene (Mishra *et al.* (2020); Karaca *et al.* (2018); Albanese *et al.* (2020)). Especially in maritime scenarios, where wide areas need to be quickly overseen and searched, the efficient use of autonomous UAVs is crucial (Yeong *et al.* (2015)). Among the most challenging issues in this application scenario is the detection, localization, and tracking of people in open water (Gallego *et al.* (2019); Nasr *et al.* (2019)). The small size of people relative to search radii and the variability in viewing angles and altitudes require robust vision-based systems.

Currently, these systems are implemented via data-driven methods such as deep neural networks. These methods depend on large-scale data sets portraying real-case scenarios to obtain realistic imagery statistics. However, there is a lack of large-scale data sets in maritime environments. Most data sets captured from UAVs are land-based, often

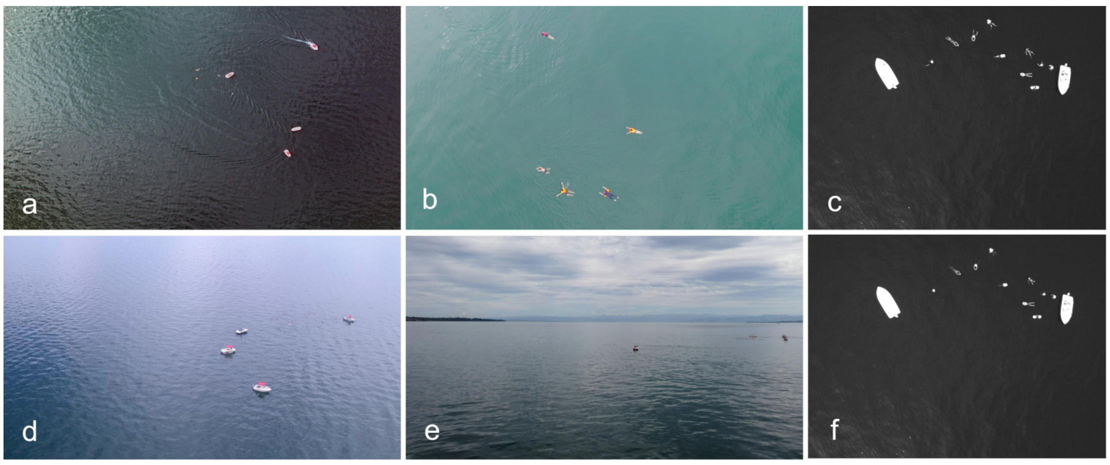


Figure 2.5: Typical image examples with varying altitudes and angles of view: (a) 250 m, 90°; (b) 20 m, 90°; (d) 50 m, 30°; (e) 10 m, 0°. Furthermore, there are examples of the Red Edge (717 nm) (c) and Near Infrared (842 nm) (f) light spectra of an image captured by the MicaSense RedEdge-MX. Note the glowing appearance of the swimmers.

focusing on traffic environments, such as VisDrone (Zhu *et al.* (2018b)) 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 benchmarks also 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 $3840{\times}2160$ px to $5456{\times}3632$ 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 development of multi-modal systems, which take these additional information into account to improve accuracy or speed. Recently, methods that rely on these metadata were proposed. However, they note the lack of large-scaled publicly available data set in that regime (see e.g. Wu *et al.* (2019)). 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.

2.4.1 Labeled Data Sets in Maritime Environments

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 (2022)) is prominent, featuring 40k images from synthetic aperture radars with instance segmentation labels. Li *et al.* (2018b) provide a data set of ships with images mainly taken from Google Earth, but also a few UAV-based images. Xia *et al.* (2018) provide satellite-based images from natural scenes, mainly land-based but also harbors. The most similar to our work is from 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.

2.4.2 Data Set Generation

As this data acquisition mission is more complex, it requires a more thorough planning. Figure 2.6 shows the location of data acquisition at Lake Constance.

We gathered the footage on several days to obtain variance in light conditions. Taking into account safety and environmental regulations, we asked over 20 student volunteers 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 at all times. Subjects were free to wear life jackets, of which we provided several differently colored pieces (see also Figure 2.7).

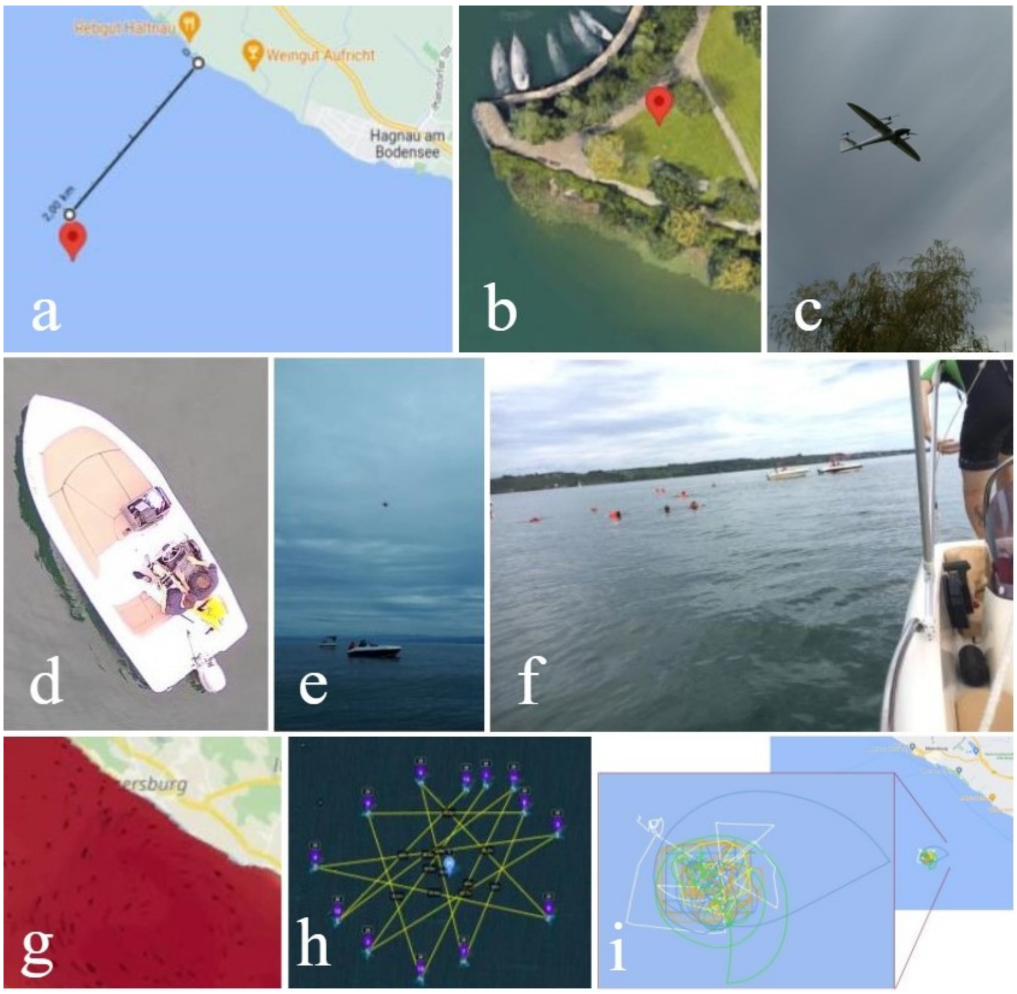


Figure 2.6: (a): Location of data acquisition: Lake Constance, 2km off Hagnau; (b): Launch location of fixed-wing UAV Trinity F90+ (c); (d)+(e): Boat, from which quadcopters were launched; (f): student volunteers from boat viewpoint. (g): Unpredictable currents; (h): flight plans of quadcopters inside the 3rd party app Litchi; (i): actual flight paths from manual control.

UAV	Camera	Resolution	Video
DJI Mavic 2 Pro	Hasselblad L1D-20c	3,840×2,160	30 fps
Trinity F90+/ DJI Matrice 210	MicaSense RedEdge-MX	1,280× 960	×
Trinity F90+	Sony UMC-R10C	5,456×3,632	×
DJI Matrice 100	Zenmuse X5	3,840×2,160	30 fps
DJI Matrice 210	Zenmuse XT2	3,840×2,160	30 fps

Table 2.3: Overview of used UAVs and cameras.

To diminish the effect of camera biases within the data set, we used multiple cameras, as listed in Table 2.3, mounted on 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 extract at most three frames per second of these videos to avoid having redundant occurrences of frames. See Section 2.5 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 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.

2.4.3 Metadata 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 2.4 is logged and provided for every image/frame read from the onboard clock, barometer, IMU and GPS sensor, and the gimbal, respectively.

Note that 90° corresponds to a top-down view, and 0° 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.

Data	Unit	Min. value	Max.value
Time since start	ms	0	∞
Date and Time	ISO 8601	–	–
Latitude	degrees	-90	$+90$
Longitude	degrees	-180	$+180$
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 2.4: Metadata that comes with every image/frame.

2.4.4 Annotation Method

Using the non-commercial labeling tool DarkLabel (Darkpgmr (2022)), 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. Our guidelines for the annotation are described in the appendix. See Figure 2.7 for examples of objects.

2.4.5 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

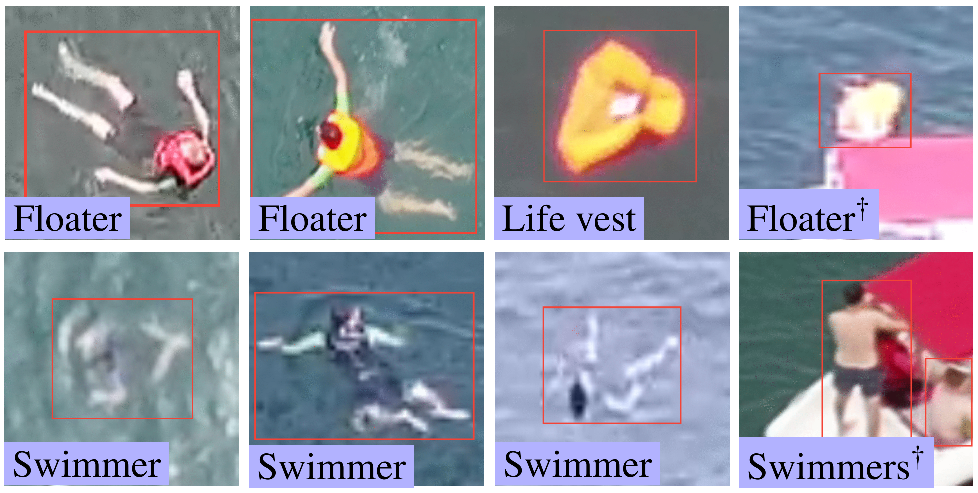


Figure 2.7: Examples of objects. Note that these examples are crops from high-resolution images. However, as the objects are small and the images taken from high altitudes, they appear blurry. Altitude and viewing angle from top left to bottom right: 50m, 70°; 40m, 80°; 20m, 30°; 40m, 40°; 110m, 90°; 60m, 50°; 70m, 40°; 40m, 50°.

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 will be available via an evaluation server, 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.

2.5 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 (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. 2.5) 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.

2.5.1 Object Detection

Given an image, the task of object detection is to predict rectangular, axis-aligned bounding box pixel locations of a pre-defined set of categories. To effectively learn this task, it is inevitable to have a large number of images and instances of each class.

There are 5,630 images (training: 2,975; validation: 859; testing: 1,796). See Figure 2.8 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 often is easier to detect than just swimmer as life jackets mostly are of contrasting color, such as red or orange (see Fig. 2.7 and Table 3.1). 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 in higher altitudes. See also Fig. 2.7.

Figure 2.8 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 ($0-90^\circ$). 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

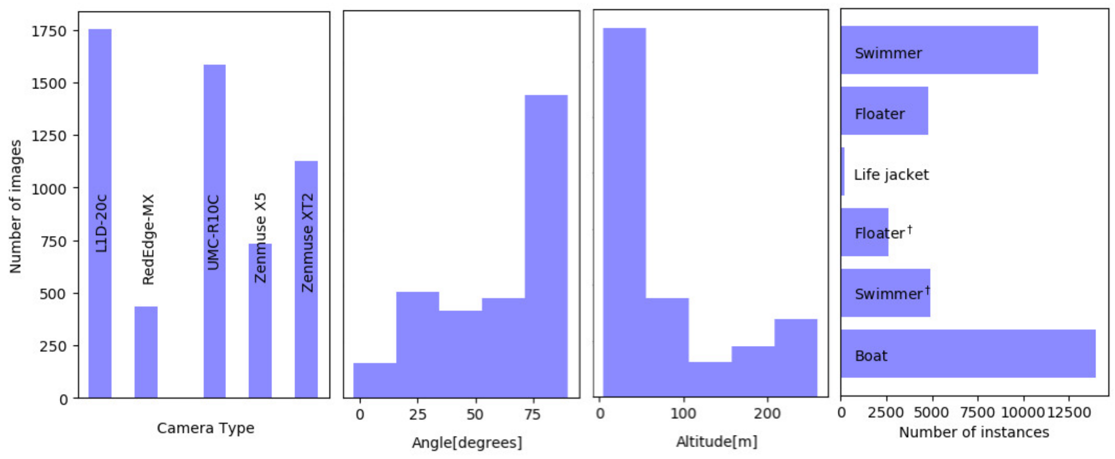


Figure 2.8: Distribution of training images over camera types (left), gimbal pitch angles and altitudes (middle), and distribution of objects over classes (right).

pixel density, which is unsuitable for object detection. Nevertheless, every altitude and angle interval is sufficiently represented.

2.5.2 Single-Object Tracking

The task of single-object tracking is to track a *single* object, whose *initial pixel location is given*, throughout the entire video clip. Boxes for each frame are axis-aligned and we can further distinguis between short-term and long-term tracking. Short-term tracking means that we only track an object for as long as it is visible in the video. Once it leaves, we may forget about it. Long-term tracking on the other hand requires to track an object even if it is not existent in some of the frames.

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.

2.5.3 Multi-Object Tracking

In Multi-Object Tracking, we are not given inital locations of objects and we must detect and track all objects ourselves. We can also distinguish between short-term and long-term tracking. Here, we only consider short-term 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 2.4.5.

2.5.4 Multi-Spectral Footage

Along with the data for the three tasks, we provide multi-spectral images. We supply bounding box 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 2.5 for an example of the individual bands.

2.6 SeaDronesSee Webserver Benchmark

As mentioned earlier, we created a webserver as a resource for researchers and professionals in the field to access and evaluate datasets and methods for various maritime computer vision tasks.

One of the main motivations for creating this webserver was to make these datasets publicly available, as they can be difficult to come by or require specific permissions to access. By hosting the datasets on this webserver, we hope to make it easier for researchers to obtain and use them in their own work.

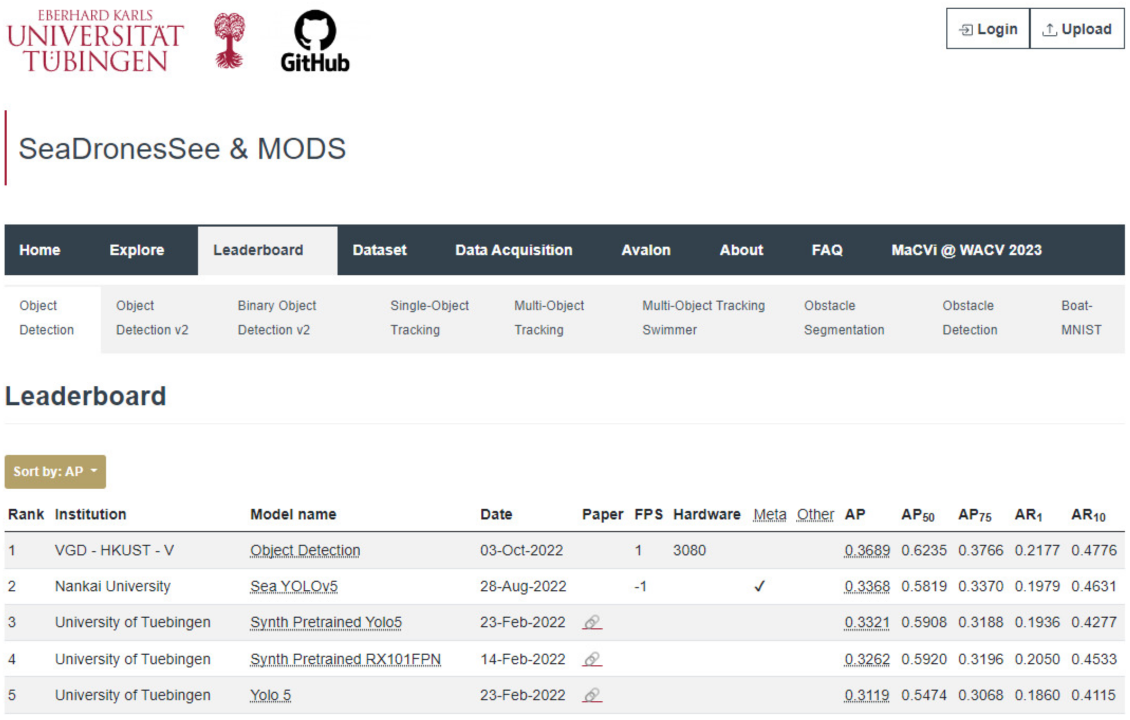
Another important reason for creating this webserver was to allow for fair comparisons of submitted methods. By providing a centralized platform for evaluating and comparing methods, we aim to facilitate the development and advancement of maritime computer vision methods.

We will shortly discuss the technical details of the webpage and the features it supports.

2.6.1 Design and Technical Background

This webserver has been built using JavaScript and Node.js, with a MariaDB database for storing user information and data. The design of the website was created using scratch/-moqups in a wireframe style, and the Bootstrap framework was used for its responsive and easy-to-personalize features. The corporate design of Universität Tübingen has been incorporated throughout the website.

The webserver includes a database of users, with registration, authentication, and login functionality (using encrypted passwords). Logged-in users are able to upload data to the server, and the webserver is also able to execute Python code as needed.



Rank	Institution	Model name	Date	Paper	FPS	Hardware	Meta	Other	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
1	VGD - HKUST - V	Object Detection	03-Oct-2022		1	3080			0.3689	0.6235	0.3766	0.2177	0.4776
2	Nankai University	Sea YOLOv5	28-Aug-2022		-1		✓		0.3368	0.5819	0.3370	0.1979	0.4631
3	University of Tuebingen	Synth Pretrained Yolo5	23-Feb-2022	🔗					0.3321	0.5908	0.3188	0.1936	0.4277
4	University of Tuebingen	Synth Pretrained RX101FPN	14-Feb-2022	🔗					0.3262	0.5920	0.3196	0.2050	0.4533
5	University of Tuebingen	Yolo 5	23-Feb-2022	🔗					0.3119	0.5474	0.3068	0.1860	0.4115

Figure 2.9: Webserver example page: Leaderboard for one of the tracks (SeaDronesSee Object Detection).

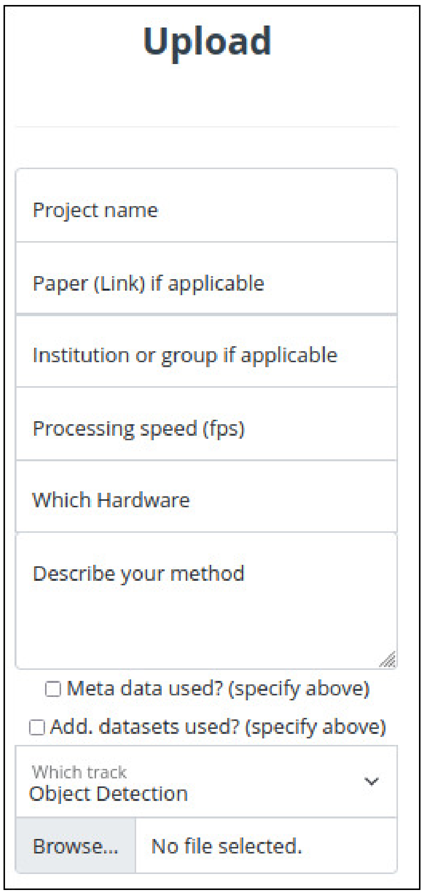
2.6.2 Uploading Feature and Comparison of Methods

One of the main features of this webserver is the ability for users to upload their predictions for each of the challenge subtracks and have them evaluated against the ground truth data, which is not publicly available. This feature is an important resource for researchers and professionals working in the field of maritime computer vision, as it allows them to quickly and easily evaluate their methods on a variety of datasets and tasks without the risk of overfitting. The latter point is ensured by restricting users to only upload a three times per day.

In addition to this, the webserver also includes leaderboards for each of the subtracks. These leaderboards allow users to compare their methods to those of other researchers in the community and see how their approaches stack up. This is a useful tool for identifying areas for improvement and staying up to date with the latest developments in the field.

Upon uploading, users are asked to provide exhaustive information on their methods, as illustrated in Figure 2.10. In view of the focus on metadata, it requires the flag whether metadata was used in the uploaded submission The leadboard for the Object Detection subtrack is depicted in Figure 2.9.

Other features of the webserver are the possibility to explore the datasets, acquire information on dataset acquisition and implemention details, which are not contained in any of the publications.


Upload

Project name

Paper (Link) if applicable

Institution or group if applicable

Processing speed (fps)

Which Hardware

Describe your method

☐ Meta data used? (specify above)

☐ Add. datasets used? (specify above)

Which track
Object Detection

Browse... No file selected.

Figure 2.10: Upload window asking the user to provide information about their method.

2.6.3 Statistics

As of the time of writing this work, the webserver hosts several datasets, such as

- SeaDronesSee Object Detection,
- SeaDronesSee Object Detection v2,
- SeaDronesSee Single-Object Tracking,
- SeaDronesSee Multi-Object Tracking,
- SeaDronesSee Video Anomaly Detection,
- Boat-MNIST.
- MODS - Obstacle Detection,
- Seagull - Sea Monitoring (no leaderboard),
- Synthetic SeaDronesSee (no leaderboard),

- SeaDronesSee Multi-Spectral Detection (no leader-board),

The reader will learn more about some of the other benchmarks in subsequent chapters. The MODS and Seagull Benchmarks are hosted here as part of a collaboration.

In total, there are over 540 uploads, 160 users and the datasets were downloaded over 5500 times. Much of the sparked interest came from the organized workshop, which will be shortly described in the following chapter.

Chapter 3

Establishing State-of-the-Art Baselines

In this chapter, we will first evaluate current state-of-the-art object detectors and tracks on SeaDronesSee. All experiments can be reproduced by using our provided code available on the evaluation server. Then, we will dicuss the challenges as part of the 1st Workshop on Maritime Computer Vision. These will further establish state-of-the-art models.

3.1 Baselines for SeaDronesSee Object Detection

The used detectors can be split into two groups. The first group consists of two-stage detectors, which are mainly built on Faster R-CNN (Girshick (2015a)) 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 of Zhu *et al.* (2018a): 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.* (2019b)). To further test an object detector in real-time scenarios, we choose the current best model family on the COCO test-dev according to PapersWithCode (2022a), i.e. EfficientDet (Tan *et al.* (2020a)), and take the smallest model, $D0$, which can run in real-time on embedded hardware, such as the Nvidia Xavier (Kiefer *et al.* (2021)). We refer the reader to the appendix for the exact parameter configurations and training configurations of the individual models.

Similar to the VisDrone benchmark (Zhu *et al.* (2018b)), 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

$$\mathrm{AP} := \mathrm{AP}^{\mathrm{IoU}=0.5:0.05:0.95},\ \mathrm{AP}_{50} := \mathrm{AP}^{\mathrm{IoU}=0.5} \text{ and } \mathrm{AP}_{75} := \mathrm{AP}^{\mathrm{IoU}=0.75}.$$

Furthermore, we evaluate the maximum recalls for at most 1 and 10 given detections, respectively, denoted

$$\mathrm{AR}_{1} := \mathrm{AR}^{\max=1} \text{ and } \mathrm{AR}_{10} := \mathrm{AR}^{\max=10}.$$

Model	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	FPS
Faster R-CNN ResNeXt-101-FPN	30.4	54.7	29.7	18.6	42.6	2
Faster R-CNN ResNet-50-FPN	14.2	30.1	7.2	6.4	17.7	14
CenterNet-Hourglass104	25.6	50.3	22.2	17.7	40.1	6
CenterNet-ResNet101	15.1	36.4	10.8	9.6	21.4	22
CenterNet-ResNet18	9.9	21.8	9.0	7.2	19.7	78
EfficientDet–*D*0	20.8	37.1	20.6	11.5	29.1	26

Model	S	F	$S^\dagger$	$F^\dagger$	B	LJ
Faster R-CNN ResNeXt-101-FPN	78.1	82.4	25.9	44.3	96.7	0.6
Faster R-CNN ResNet-50-FPN	24.6	54.1	4.9	7.5	89.2	0.3
CenterNet-Hourglass104	65.1	73.6	19.1	48.1	95.8	0.3
CenterNet-ResNet101	16.8	39.8	0.8	1.7	74.3	0
CenterNet-ResNet18	20.9	21.9	2.6	3.3	81.9	0.4
EfficientDet–*D*0	65.3	55.1	3.1	3.3	95.5	0.1

Table 3.1: Average precision results for several baseline models. The bottom part contains AP_{50}–values 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.

All these metrics are averaged over all categories (except for "ignored region"). We furthermore provide the class-wise average precisions. Moreover, similar to Kiefer *et al.* (2021), we report AP_{50}-results on different equidistant levels of altitudes:

Low (L)	Low-Medium (LM)	Medium (M)	Medium-High (MH)	High (H)
5-56 m	55-106 m	106-157 m	157-208 m	208-259 m

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 (A)	Acute-Medium (AM)	Medium (M)	Medium-Right (MR)	Right (R)
7-23°	23-40°	40-56°	56-73°	73-90°

Ultimately, it is the goal to have robust detectors across all domains uniformly, which is better measured by the latter metrics.

Table 3.1 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

Model	L	LM	M	MH	H	AP_{50}^{avg}
Faster R-CNN ResNeXt-101-FPN	56.8	54.6	49.2	65	78.3	60.8
Faster R-CNN ResNet-50-FPN	32.8	29.8	23.5	40.5	48.9	35.1
CenterNet-Hourglass104	50.6	52.0	47.5	64.9	73.2	57.6
CenterNet-ResNet101	20.2	30.4	24.1	35.1	38.0	29.6
CenterNet-ResNet18	23.8	20.3	19.2	29.3	31.9	24.9
Efficient-Det $D0$	39.6	38.0	30.4	42.5	54.5	41.0

Table 3.2: 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}
Faster R-CNN ResNeXt101-FPN	68.3	55.1	45.2	63.6	51.5	56.7
Faster R-CNN ResNet50-FPN	32.8	35.5	32.7	35.7	27.6	32.9
CenterNet-Hourglass104	66.4	42.1	49.7	58.7	46.9	52.76
CenterNet-ResNet101	7.4	35.8	20.5	33.6	21.7	23.8
CenterNet-ResNet18	9.6	29.5	26.3	27.9	22.1	23.1
Efficient-Det $D0$	26.9	47.0	40.5	40.3	36.8	38.3

Table 3.3: Results on different angle-domains. For example, ResNeXt's AP_{50} performance in medium-right (MR) angles (57-73°) is 63.6 AP_{50}.

fast networks, such as CenterNet-ResNet18 and EfficientDet-$D0$, expectedly perform worse. However, the latter can run in real-time on an Nvidia Xavier (Kiefer *et al.* (2021)). 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. 2.7). Since there is a heavy class imbalance with many fewer life jackets, detectors are biased towards floaters.

Table 3.2 and 3.3 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. 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	FP	FN	Rec.	Prcn	ID	Frag
FairMOT-D34	39.0	44.8	23.6	3,604	9,445	57.2	77.8	307	1,687
FairMOT-R34	15.2	27.6	33.7	2,502	12,592	30.1	68.4	181	807
Tracktor++	55.0	69.6	25.6	7,271	3,550	85.5	74.2	165	347

Table 3.4: Multi-Object Tracking evaluation results for the **Swimmer** task.

Model	MOTA	IDF1	MOTP	FP	FN	Rec.	Prcn	ID	Frag
FairMOT-D34	36.5	43.8	20.9	3,788	20,867	47.2	83.1	447	1,599
FairMOT-R34	30.5	40.8	27.3	4,401	28,999	40.2	81.6	285	1,588
Tracktor++	71.9	80.5	20.1	7,741	5,496	88.5	84.5	192	438

Table 3.5: Multi-Object Tracking evaluation results for the **All-Objects-In-Water** task.

3.2 Baselines for SeaDronesSee 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 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 last VisDrone workshop (Fan *et al.* (2020b)).

Figure 3.1 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.

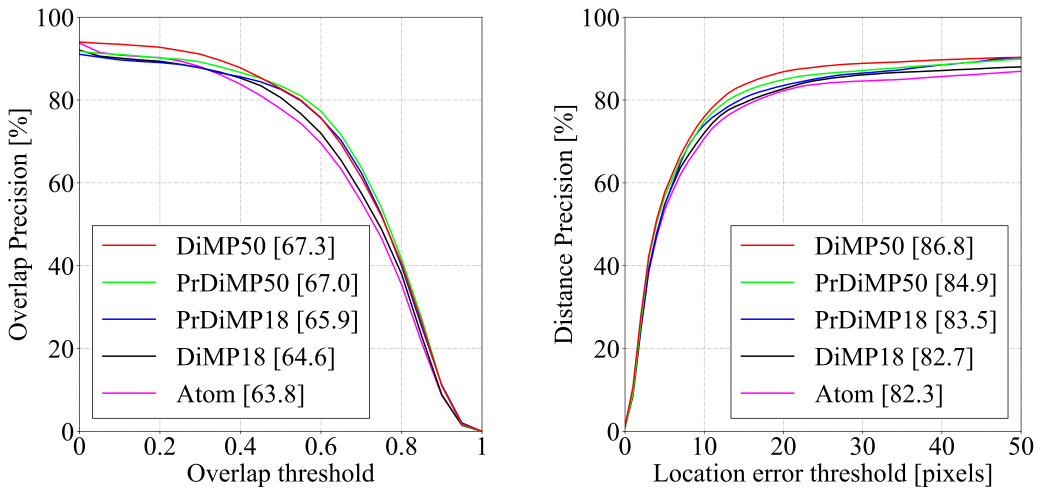


Figure 3.1: Success and precision plots for single-object tracking task.

3.3 Baselines for SeaDronesSee Multi-Object Tracking

We use a similar evaluation protocol as the MOT benchmark (Milan *et al.* (2016a)). 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.

3.4 MaCVi Workshop

Partially responding to the sparked interested in the UAV-based maritime domain, we hosted the 1st Workshop on Maritime Computer Vision (MaCVi) 2023 as part of IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). While this workshop has a broader scope, it is relevant for this work as we hosted two challenges directly tied to the SeaDronesSee benchmark introduced earlier: SeaDronesSee Object Detection v2 and SeaDronesSee Multi-Object Tracking. The latter is the same benchmark we described earlier, while the former is an extension of the SeaDronesSee Object Detection benchmark. These challenges help establish baselines for these two benchmarks and we will refer to some of the baselines in later methods. In this section, we will shortly review the workshop challenges following Kiefer *et al.* (2023).

The 1st Workshop on Maritime Computer Vision (MaCVi) 2023 focused on maritime computer vision for Unmanned Aerial Vehicles (UAV) and Unmanned Surface Vehicle (USV), and organized several subchallenges in this domain: (i) UAV-based Maritime Object Detection, (ii) UAV-based Maritime Object Tracking, (iii) USV-based Maritime Obstacle Segmentation and (iv) USV-based Maritime Obstacle Detection. The subchallenges were based on the SeaDronesSee and MODS benchmarks. This section summarizes the main findings of (i) and (ii). We introduce a further benchmark, called SeaDronesSee Object Detection v2, which extends the previous benchmark by including more classes and footage. We provide statistical and qualitative analyses, and assess trends in the best-performing methodologies of over 130 submissions. The methods are summarized in the appendix. The datasets, evaluation code and the leaderboard are also publicly available on the workshop homepage[1].

3.4.1 Challenge Participation Protocol

The challenge tracks were announced on the 20th of August 2022 and ran until the 25th of October 2022. At the announcement date, participants could download the datasets and evaluation and visualization toolkits from the workshop homepage. Participants could experiment with their methods on this data before they could upload their predictions on the individual tracks' test sets on the webserver from the 14th of September onwards. The predictions were compared with the corresponding ground truth annotations on the server-side. Lastly, participants could choose to show their result on the leaderboard or to delete the submission.

At the start of the uploading phase, participants were allowed to upload predictions three times per day independent of the challenge track. The submitted predictions were said to be subject to further inspection on our side regarding the exact performances and participants were required to provide information on their used methods, and in the USV-based tracks, participants were required to submit their code as well. The respective metrics for the individual challenge tracks decided whether the submission reached a top-3 position. Furthermore, we required every participant to submit information on the speed of their method measured in frames per second wall clock time and their hardware. Lastly, participants needed to indicate which data sets (also for pretraining) they used during training.

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

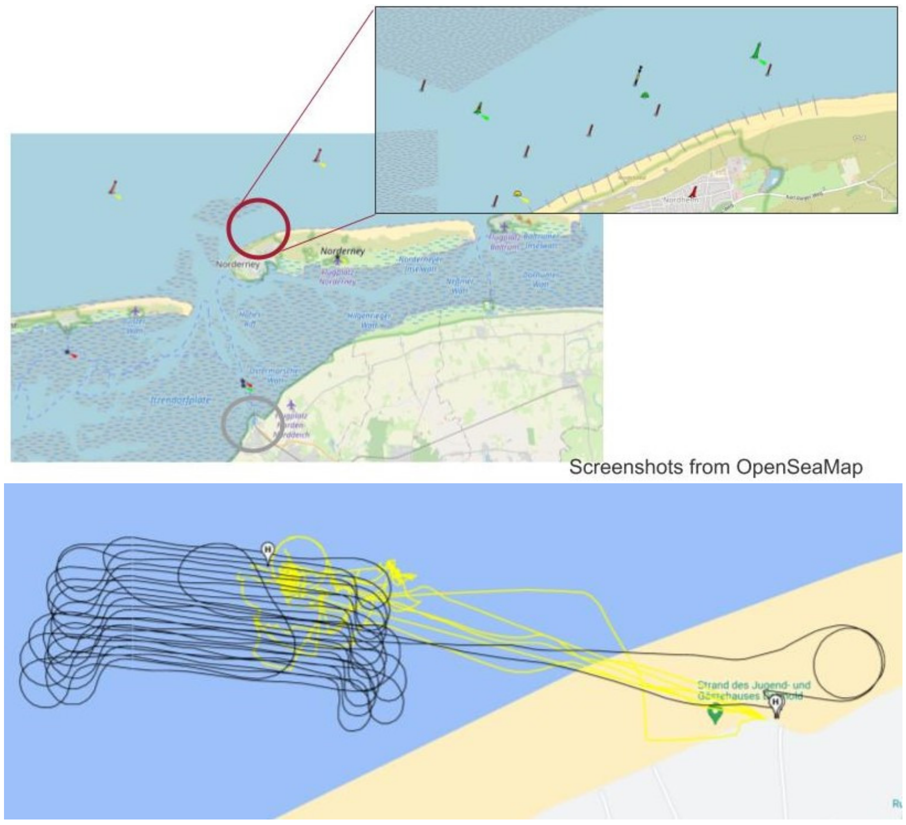

Norderney
Screenshots from OpenSeaMap
Strand des Jugend- und Gästehauses

Figure 3.2: Top: Data mission location at North Sea close to Norderney. Bottom: Both, the Trinity-F90+ (black) and the quadcopters (yellow) were controller from the shore.

Additionally, the teams that reached a performance above our least performing baseline were asked to submit a short technical report, describing their methods and training configurations. Please see Kiefer *et al.* (2023) for more information.

3.4.2 SeaDronesSee Object Detection v2 Challenge

The goal of this challenge was to detect humans, boats and other objects in open water. The task of object detection in maritime SaR is far from solved. For example, the best performing model of the SeaDronesSee object detection track currently achieves 36% mAP, as opposed to the COCO benchmark with the best performer achieving over 60% mAP. SeaDronesSee is more challenging due to lighting conditions and sun reflections, different appearances coming from various altitudes and viewing angles. While the sparsity of object locations often results in false positives, the small sizes of objects along partial occlusion due to water lead to false negatives.

For the challenge of the workshop, we made a few changes from the original SeaDronesSee object detection benchmark. In addition to the already publicly available data,

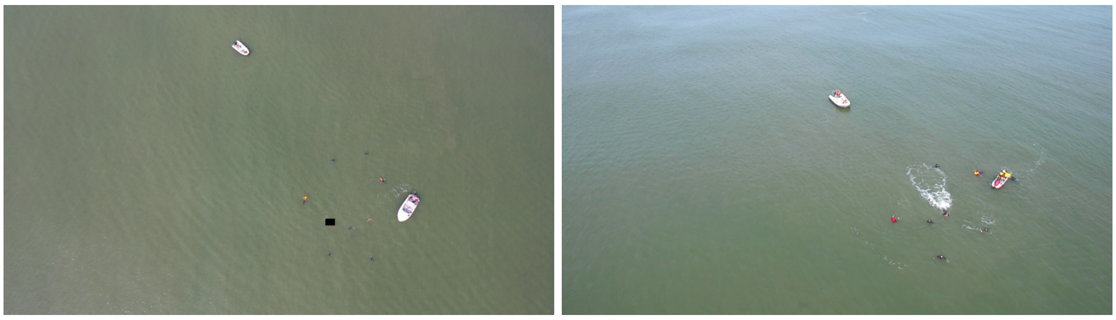


Figure 3.3: Example images of newly generated images in the SDS ODv2 dataset. The black rectangle denotes an ignored region.

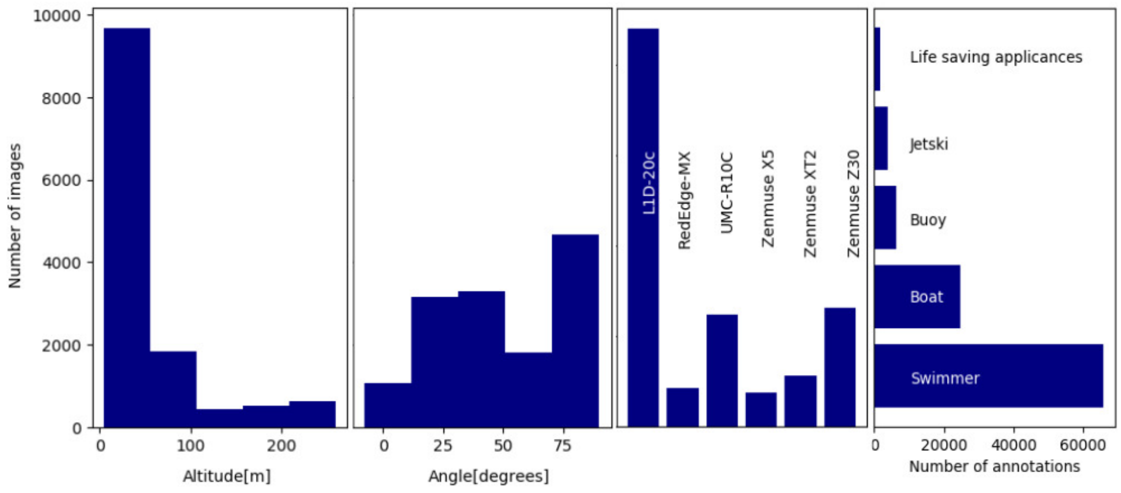


Figure 3.4: Altitude and gimbal pitch angle distribution (left two), and camera and class distribution (right two) of images in SDS OD v2.

we collected further training data, which is included at the start of the challenge. In particular, we extend the object detection track of SeaDronesSee by roughly 9k newly captured images depicting the sea surface from the viewpoint of a UAV. See Figure 3.3 for examples images. The ground-truth bounding boxes are available and the evaluation protocol is based on the standard mean average precision. Owing to the application scenario, we also evaluate the class-agnostic performances, which resembles the use-case of detecting anything that is not water.

Dataset

Similar to the SeaDronesSee Object Detection dataset, we capture this dataset as part of a data acquisition mission at North Sea.

The SeaDronesSee-Object Detection v2 (S-ODv2) dataset contains 14,227 RGB im-

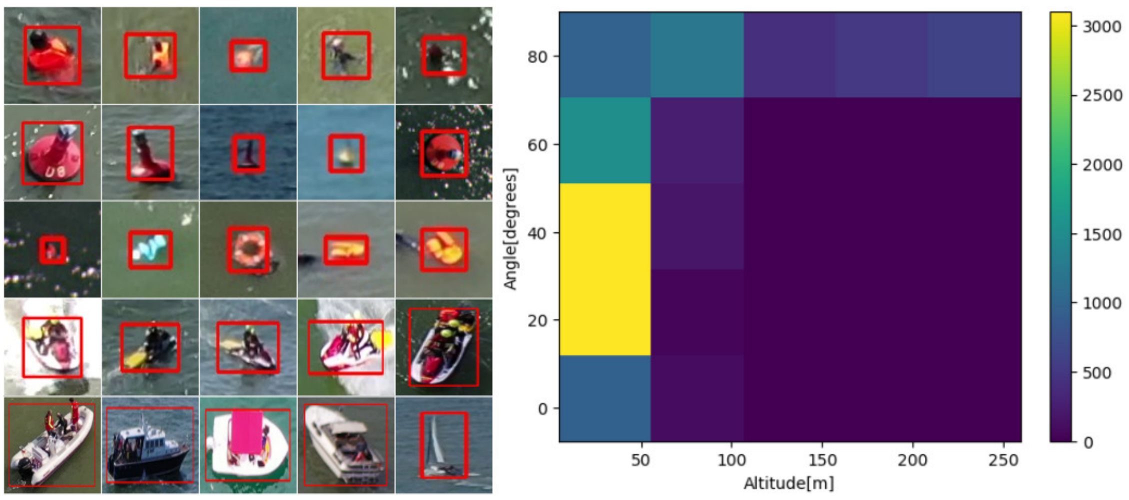


Figure 3.5: Left: Class instances in SDS ODv2. From top to bottom: swimmer, buoy, life vest/life belt, jetski, boat. Right: Altitude-angle distribution of images in SDS ODv2.

ages (training: 8,930; validation: 1,547; testing: 3,750). The images are captured from various altitudes and viewing angles ranging from 5 to 260 meters and 0 to 90° degrees (gimbal pitch angle) while providing the respective meta information for altitude, viewing angle and other metadata for almost all frames. See Figure 3.4 for the altitude and viewing angle distribution. Most images come with additional metadata as depicted in Table 2.4. Note that there are 2,830 images without any metadata labels and 686 images with only gimbal pitch angle labels. The images were captured with six different cameras as depicted in Figure 3.4. Note that we only used the RGB channels if more channels were available. Each image is annotated with labels for the classes

- swimmer
- boat
- jetski
- buoy
- life saving appliance (life vest/belt).

See sample instances of these classes in Figure 3.5. Additionally, there is an ignore class. This region contains difficult to label or ambiguous objects. We blackened out these regions in the images already. Figure 3.4 shows the class distribution and the heavy class imbalance in the dataset. Although the bounding box annotations for the test set are withheld, the metadata labels for the test set were provided.

Evaluation Protocol

We evaluate the predictions on the commonly used AP, AP50, AP75, AR1 and AR10 from the COCO evaluation protocol (Lin *et al.* (2014)). We provided the full evaluation protocol as part of our evaluation kits available online (Kiefer *et al.* (2022)). For the

Model name	Data	Type	Backbone	Module
Maritime-VSA	C, S-O_t	Transf.	DB-Swin-S	Casc. R-CNN
DetectoRS	C, S-O_{all}	2-stg.-CNN	ResNet-50	Casc. R-CNN
YOLOv7-Sea	C, S-O_t	1-stg.-CNN	E-ELAN	SimAM
DyHead	C,S-O_t	Transf.	Swin-L	Dynamic Head
YOLOv7-X	C, S-O_t	1-stg.-CNN	YOLOv7-X	
YOLO-CNS	C, S-O_t	Transf./CNN	Swin Transf.	CBAM, NAM
YOLOv7-W6	C, S-O_t	1-stg.CNN	YOLOv7-W6	
M10	S-O_t	1-stg.CNN	ResNeXt-101	VarifocalNet
YOLOv7-NYU	C, S-O_t	1-stg.CNN	E-ELAN	Super-Res.
YOLOv7-FIT	C, S-O_t	1-stg.CNN	YOLOv7-E6	
DurObj	VD, S-O_t	1-stg.CNN	ResNet-101	TOOD
APX	C, S-O_t	1-stg.CNN	Yolov7	APX
YOLOv7-TILE	C, S-O_t	1-stg.CNN	YOLOv7	

Table 3.6: Object Detection v2 submissions overview. For brevity, we denoted S=SeaDronesSee, O=Object Detection v2, all=(t)rain and (v)al set, VD=VisDrone, C=COCO. For a list of used augmentations, please refer to Kiefer *et al.* (2023).

first subtrack, we average the AP results over all classes. We further analyze the models using other metrics, such as TIDE (Bolya *et al.* (2020)) and by leveraging the available metadata. The determining metric for winning will be AP. In case of a draw, AP50 counts.

Submissions, Analysis and Trends

We received 77 submissions from 18 different teams. We also provided two additional baselines, a YOLOv7 and a Faster R-CNN with ResNet-18 backbone. In our analysis, we will focus on the top 13 models that outperformed both these baselines. None of the methods employed ensembles or were trained on any uncommon dataset. Only some submissions used the SDS ODv2 validation set for training.

Three of the submitted models were transformer-based, which originally were especially hard to tune for small object detection, but was recently found popular also in the aerial object detection domain (Cao *et al.* (2021)). More precisely, the winner of this challenge, Maritime-VSA, the 4th place, DyHead, and the 6th place rely either entirely or partly on transformer-based blocks. Maritime-VSA showcase their recently published varied-size window attention, which is suitable for processing large image resolutions compared to more traditional transformer architectures. In conjunction with the popular Cascade R-CNN as a detection head and test-time augmentations, they obtained a significant lead. DyHead leverage the recent so-called dynamic heads to unify the object detection heads for localization and classification via attention mechanisms (Dai

Model name	FPS	Hardware	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	Bin_{AP}
Maritime-VSA	1	A100	0.62	0.91	0.68	0.48	0.70	0.56
DetectoRS	1	Tesla V100	0.60	0.90	0.66	0.47	0.67	0.54
YOLOv7-Sea	1	Tesla V100	0.59	0.91	0.64	0.46	0.68	0.54
DyHead	1	A100	0.57	0.89	0.62	0.45	0.68	0.52
YOLOv7-X	15	RTX 3090	0.54	0.85	0.57	0.44	0.61	0.50
Yolo-CNS	60	TeslaP6	0.53	0.83	0.56	0.44	0.62	0.49
YOLOv7-W6	10	RTX 3090	0.53	0.84	0.56	0.44	0.62	0.49
M10	1	RTX3090	0.53	0.84	0.55	0.43	0.60	0.47
YOLOv7-NYU	-1	2080	0.52	0.86	0.54	0.43	0.60	0.46
YOLOv7-FIT	6	RTX3090	0.52	0.80	0.55	0.42	0.58	0.49
DurObj	4	TITAN XP	0.50	0.79	0.51	0.42	0.58	0.47
APX	60	RTX 3050	0.50	0.83	0.50	0.41	0.58	0.45
YOLOv7-TILE	3	Nvidia Titan	0.42	0.71	0.44	0.36	0.50	0.44
YOLOv7-BL	66	RTX 3080	0.42	0.72	0.42	0.36	0.49	0.41
FRCNN-RN-BL	29	GTX 1080 Ti	0.24	0.52	0.20	0.24	0.32	0.21

Table 3.7: Final leaderboard for SeaDronesSee Object Detection v2. Bin.=Binary. ”-1” denotes unknown.

et al. (2021)). Test-time augmentations and large image resolutions were employed. The method rightfully mentions the problems with annotation errors, which will be analyzed below.

The remaining models are different types of CNNs. The 2nd place, DetectoRS, base their submission on Cascade R-CNN (Cai and Vasconcelos (2018)), which is well known for its performance in small object detection (see e.g. performance on VisDrone workshop (Cao *et al.* (2021)). A likely significant addition is that they employed large resolutions and multi-scale testing. Several other methods are based on a YOLO-variant, most prominently the current YOLOv7 (Wang *et al.* (2022)) architecture. In fact, the 3rd (Zhao *et al.* (2023)), 5th, 7th, 9th, 10th, 12th and 13th places all base their submissions on YOLOv7. Many YOLOv7 submissions either adapted the architecture to include an attention module or tuned hyperparameters, such as considerably increasing the image size, or included augmentations, such as random cropping, mosaicing or color changes just to name a few.

The remaining methods use more specific architectures, such as VarifocalNet (Zhang *et al.* (2021a)), a single-stage object detector, which itself is based on FCOS (Tian *et al.* (2019)). Further augmentations, such as tiling (also multi-scale) improved the performance significantly. See Table 3.6 for an overview of the submitted methods.

Table 3.7 shows the final standing of this challenge track. Notably, the performance of the top models is above 90 AP_{50}. Owing to the aerial nature and potentially sub-optimal

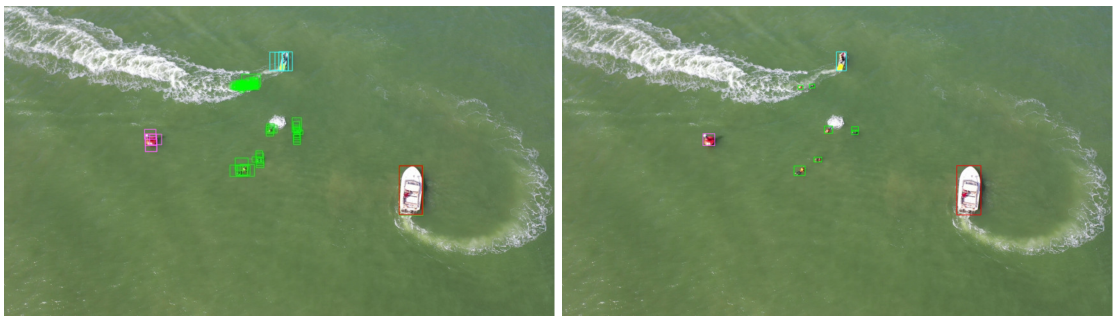


Figure 3.6: Example predictions from Maritime-VSA (leftr) and DetectoRS (right). Note that we did not filter based on the confidence score, which is why the first method has many predictions for each object. The confidence score for most of them is very low, which is why the AP won't suffer from these.

label accuracy (e.g. shifted labels), the averaged AP is far lower, which is also reflected in the lower AR_1 and AR_{10} scores. The binary AP, which measures the foreground vs. background performance, is slightly worse for almost all models which is likely caused by the class imbalance with the majority of the instances being swimmers, which is generally a hard class to predict (Kiefer *et al.* (2023)).

Generally, the classes swimmers and life saving appliances are believed to be the hardest classes as their appearance vary the most and they are the smallest (and thus hardest to predict) objects (see also Figure 3.5). Furthermore, these two classes are harder to distinguish and there are only few instances of life saving appliances. Furthermore, the methods' ranks in performance across different precision levels are consistent as can be seen from Figure 3.8, i.e. every model is more or less better or worse than any other model for all precision scores consistently.

A closer analysis on the type of error can be seen from the TIDE plots in Figure 3.8. There, we plot the different error types of the two best performing submissions, Maritme-VSA and DetectoRS. Both models behave similarly in their error type influence distribution, e.g. most of the errors come from localization errors (roughly 50%). Background errors (falsely predicting background to be any class instance) are a similarly often cause of errors as missing to detect objects in the case of Maritime-VSA. However, DetectoRS takes a different trade-off and mostly only misses objects as opposed to detecting background as foreground objects. Note, however, that the overall magnitude of errors is lower for Maritime-VSA for both types of errors (bottom bar charts). The less common duplicate detections errors only play a role in Maritime-VSA, which aligns with the qualitative prediction example in Figure 3.6. Note, however, that these duplicate predictions have low confidence and hence do not matter too much in the overall AP calculation.

Table 3.8 shows the AP values broken down by different metadata configuration intervals. Again, the models perform mostly consistently across different domains. Generally significantly visible, the performance for acute angles is low across all models. While

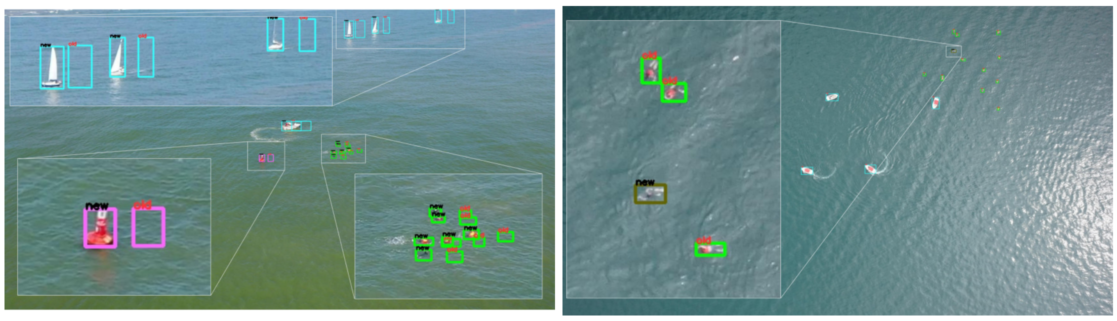


Figure 3.7: Label errors revealed after another iteration of manual annotation. Left: Displaced labels, Right: missing labels (red font: old, black font: new).

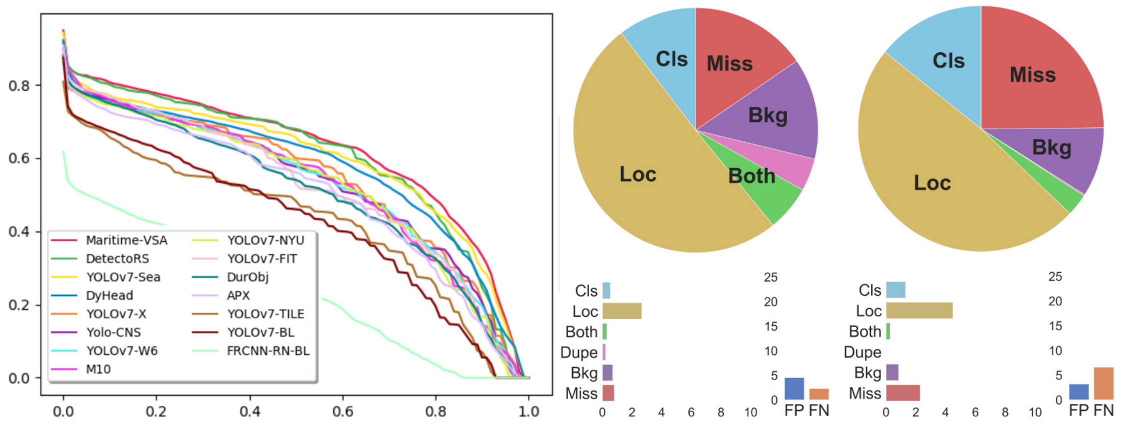


Figure 3.8: Left: Precision (x-axis)-recall (y-axis)-curve, for submitted methods. TIDE evaluations for Maritime-VSA (middle) and DetectoRS (right).

this may simply be the cause of having fewer images in that domain (compare to Fig. 3.4), these images often contain very small objects in the distant horizon. Furthermore, in the case of swimmers, these are hardly visible as only their body parts above the water are visible (see e.g. the first swimmer of Figure 3.5 compared to the third one). Surprisingly, the performance on high altitudes is the highest. This could be the cause of consistent viewpoints, as images from high altitude exhibit viewpoints almost always of close to 90° (looking downwards; see Figure 3.5. The performances broken down by different cameras is not conclusive. The performance for the M210 UAV is very low, which can only be hypothesized to be partly attributed to the M210 UAV carrying the lower resolution Zenmuse Z30 camera, although there exist many images (compare to Fig. 3.4). The high performance for the trinity drone is again believed to be caused by the consistent 90° facing downwards viewpoint as this UAV only has facing downwards cameras.

The dataset contains a fair amount of label errors, which we found upon reiterating a whole manual annotation pass over the dataset. We found 678 missing boxes and

Model name	AP_L	AP_M	AP_H	AP_A	AP_{AR}	AP_R	AP_{Mav}	AP_M	AP_{Tri}
Maritime-VSA	0.62	0.57	0.68	0.23	0.65	0.64	0.61	0.18	0.71
DetectoRS	0.61	0.55	0.70	0.22	0.63	0.63	0.59	0.17	0.69
YOLOv7-Sea	0.61	0.53	0.67	0.21	0.63	0.62	0.59	0.16	0.66
DyHead	0.59	0.49	0.63	0.18	0.62	0.60	0.57	0.17	0.64
YOLOv7-X	0.56	0.48	0.56	0.18	0.59	0.55	0.54	0.16	0.59
Yolo-CNS	0.56	0.42	0.64	0.18	0.58	0.55	0.53	0.13	0.61
YOLOv7-W6	0.55	0.47	0.62	0.17	0.58	0.58	0.53	0.12	0.62
M10	0.55	0.41	0.67	0.14	0.57	0.60	0.52	0.13	0.61
YOLOv7-NYU	0.53	0.49	0.63	0.13	0.57	0.61	0.51	0.14	0.55
YOLOv7-FIT	0.53	0.42	0.67	0.17	0.56	0.58	0.51	0.13	0.63
DurObj	0.52	0.40	0.64	0.13	0.56	0.56	0.49	0.14	0.58
APX	0.54	0.39	0.54	0.13	0.56	0.53	0.50	0.11	0.54
YOLOv7-TILE	0.45	0.36	0.30	0.13	0.49	0.35	0.43	0.10	0.47
YOLOv7-BL	0.44	0.34	0.42	0.06	0.50	0.45	0.42	0.12	0.43
FRCNN-RN-BL	0.26	0.23	0.26	0.00	0.33	0.27	0.25	0.03	0.21

Table 3.8: AP results for subsets, divided by altitude ((L)ow, (M)edium, H(igh)), gimbal pitch ((A)cute, (A)cute to (R)ight, (R)ight), and camera ((Mav)ic, (M)210 and (Tri)nity. We divide the 'Altitudes' and 'Angles' into three equidistant intervals.

615 displaced boxes. See examples of label errors in Figure 3.7. Displaced label errors come from the used annotation tool Darklabel's tracking functionality (Darkpgmr (2022)), which causes a drag in the bounding box labels in scenes where there is a lot of camera or UAV movement. Missing labels mostly occur in static images where there was no underlying video that aided the human annotators in finding objects to label. Rerunning the predictions on this corrected dataset shows that the performances indeed improve across all models but the overall order stays almost the same (Kiefer *et al.* (2023)).

Discussion and Challenge Winners

The challenge results have shown that transformer architectures start to gain traction in the aerial domain as well, while CNN architectures are still the standard choice for such tasks. The easy-to-use and yet strong one-stage detector YOLOv7 is a very popular choice. As is common for these kind of challenges (compare to Aiskyeye (2022)), test-time augmentations are applied and significantly boost the performance at the cost of slower run times. Furthermore, using large resolutions is one of the keys to obtaining high accuracies, be it by means of architecturally supporting large resolutions or by targeted augmentations, such as cropping.

The observation above is exemplified by the winner trio: The first place from The University of Sydney, Maritime-VSA, employed transformers, the second place from

Fraunhofer IOSB, DetectoRS, leveraged the popular two-stage detector Cascade R-CNN, and the third place from Beijing University of Posts and Telecommunications, YOLOv7-Sea, built upon the current YOLOv7 detector.

Furthermore, most submitted object detectors run far from real-time. Moreover, special consideration should be given to the used hardware in that case since in this challenge, participants mostly relied on high-end GPUs, such as V100s.

Therefore, research in this domain needs to consider runtime constraints imposed in real applications of these detectors. In future iterations of MaCVi, this would need to be a focus.

3.4.3 SeaDronesSee Multi-Object Tracking Challenge

Part of the SeaDronesSee benchmark was the Multi-Object Tracking track. This track focuses on tracking objects in water which are of interest in SaR scenarios, while it could also be leveraged for surveillance. In SaR scenarios, it might be of interest to track the detection and position of people or boats over time, so that the found subjects are easily distinguishable. However, tracking small, partly occluded subjects, which change their appearance based on their movement and occlusion level due to water, is non-trivial. Gimbal movement and altitude change cause objects to move quickly within the video frames. For these reasons, we hosted the first SeaDronesSee-MOT challenge track, which will be discussed in the following.

Dataset

The SeaDronesSee-MOT dataset consists of 21 clips in the train set, 17 clips in the validation set and 19 clips in the test set with a total of 54,105 frames and 403,192 annotated instances. Every frame is annotated with the ground-truth bounding boxes along unique ids for the following classes:

- swimmer
- floater
- life jacket
- swimmer on boat
- floater on boat
- boat

Floater denotes a swimmer wearing a life jacket. Following the original definition, for the SeaDronesSee-MOT challenge track, we restrict the task as follows. We only require the objects boats, swimmer and floater to be tracked in a one-class setting, where we do not distinguish between different classes. We note that this is a *short-term* tracking task (Kristan *et al.* (2016)), i.e. objects that disappear from the scene need not be tracked anymore. Each frame comes with precise metadata labels regarding altitude, angles of the UAV and the gimbal, GPS, and more.

Model name	Data	Detector	Modules	FPS	C/GPU
MoveSORT	COCO, S-MOT	YOLOv7	ECC , NSA K.	10 (d+t)	T4
byteTracker	COCO, S-MOT	YOLOX		6 (d+t)	A100
StrongerSORT	S-MOT M1501	pub. det. (YOLOv7)	PCB	10 (t)	M1
MOT	COCO, S-O^{all}	Casc. R-CNN ResNet-50,		1 (d+t)	V100
OCSORT	S-O^{all}, S-MOT^{all}	YOLOX-XL		20 (d+t)	V100
Tracktor Baseline	COCO, S-MOT	F. R-CNN, ResNet-50	ECC	10 (d+t)	RTX 3080

Table 3.9: Multi-Object Tracking submissions overview. For brevity, we denoted d=detector, t=tracker, S=SeaDronesSee, O=Object Detection v2, M1501=Market1501, all=train and val set.

Evaluation Protocol

We evaluate the submissions by using the following metrics: HOTA, MOTA, IDF1, MOTP, MT, ML, FP, FN, Recall, Precision, ID Switches, Frag (Luiten *et al.* (2021); Leal-Taixé *et al.* (2015)). The determining metric for winning is HOTA. In case of a tie, MOTA is the tiebreaker.

Furthermore, we require every participant to submit information on the computational runtime of their method measured in frames per second wall-clock time along their used hardware.

Submissions, Analysis and Trends

We received 18 submissions from 7 different institutions. Additionally, we provided a baseline, i.e. a Tracktor-based tracker using ECC with a Faster R-CNN ResNet-50 detector. We used the mmtracking implementation (Chen *et al.* (2019a)) with default hyperparameters. We also provided public detections so that participants do not need to train their own detectors. These are from a YOLOv7 model pretrained on COCO and trained on SeaDronesSee-MOT train set for 8 epochs yielding an AP of roughly 0.5. For reference, the same model (except for the number of class outputs) has an AP of 0.4181 on Object Detection v2, which is not optimal (compare to best models).

All of the 18 submitted trackers outperformed the baseline. See an overview of the

Model name	HOTA	MOTA	IDF1	MOTP	Re	Pr	IDs	Frag
MoveSORT	0.67	0.80	0.77	0.19	0.89	0.91	44	805
byteTracker	0.65	0.77	0.77	0.21	0.88	0.89	68	841
StrongerSORT	0.63	0.74	0.75	0.20	0.86	0.88	243	1396
MOT	0.62	0.76	0.71	0.19	0.89	0.88	445	672
OCSORT	0.61	0.72	0.69	0.19	0.81	0.91	106	671
Tracktor Baseline	0.46	0.48	0.50	0.21	0.62	0.83	1435	2522

Table 3.10: Multi-Object Tracking results on the SeaDronesSee-MOT test set. The submissions are ranked based on HOTA. The last row indicates the baseline. Gold, silver and bronze denote the first, second and third place, respectively. For MT, ML, FP and FN, refer to Kiefer *et al.* (2023).

submitted methods in Table 3.9. Table 3.10 shows the results of the best submissions of the best five teams. All submissions followed the tracking-by-detection paradigm. Since it was allowed to train on any data, most submissions did so and incorporated stronger detectors as the provided public detection baseline.

MoveSORT performed best in terms of HOTA, MOTA and IDF1 metrics although they only trained on SeaDronesSee-MOT. Being the best model in these metrics suggests that it is a very robust model w.r.t. detection and association accuracy. However, they relied on the recent YOLOv7 Wang *et al.* (2022) detector, which may yield good detection results to work with. Notably, it only made 44 ID switches, which may the cause of the underlying DeepSORT implementation, which focuses specifically on decreasing the number of ID switches. However, also note all models have rather low id switch numbers, which is due to the sparse nature of the dataset where objects are not too cluttered (see e.g. Fig. 3.9). MoveSORT further claims to improve on DeepSORT by using the enhanced correlation coefficient maximization module (ECC) to estimate the global rotation and translation between adjacent frames. Indeed, association between frames for fast moving camera movements is a problem in certain video clips of SeaDronesSee-MOT as exemplified qualitatively in Figure 3.9. Furthermore, they added the NSA Kalman filter module Du *et al.* (2021) from the second place tracker of the VisDrone 2021 MOT challenge Chen *et al.* (2021a).

The method byteTracker placed second basing their submission on the recent ByteTrack implementation (Zhang *et al.* (2022a)). They adapted the tracker's focus on the MOT17 challenge (Milan *et al.* (2016b)) to the maritime setting by removing the vertical bounding box restriction and by changing hyperparameters, such as increasing the non-max-suppression threshold to remove potential false associations and to decrease the number of ID switches. The underlying detector was a large YOLOX-x model, which might explain some of the good performance.

StrongerSORT placed third, mostly owing to its ability to reliably associate tracklets as

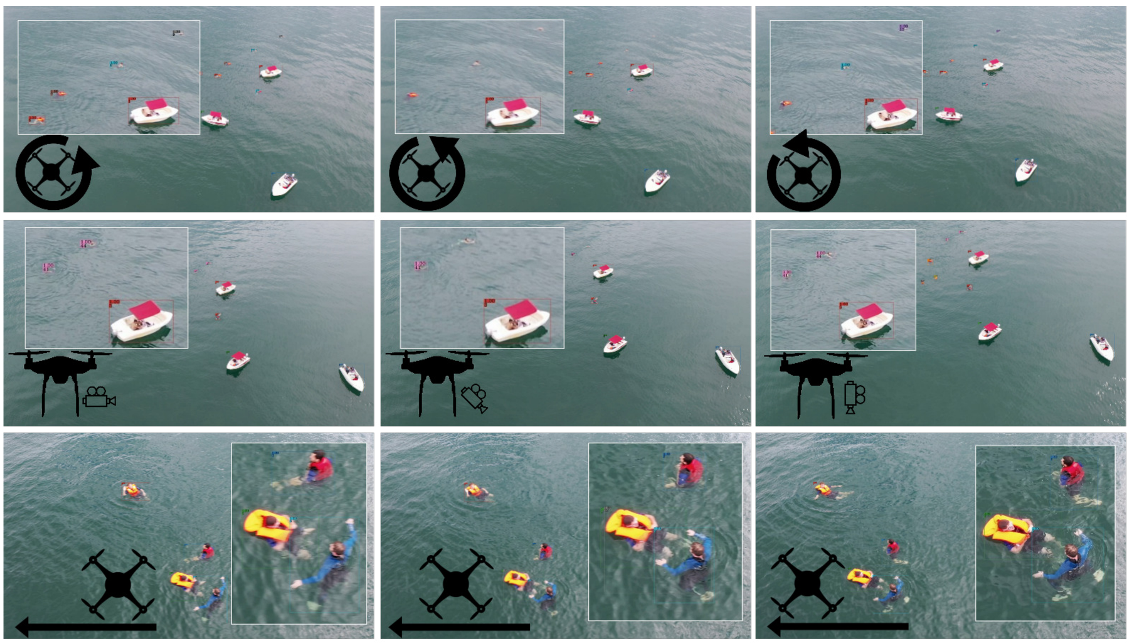


Figure 3.9: Three common types of error causes. Predicted tracks from MoveSORT. Top: Panning by changing the heading angle cause the tracker to lose the three swimmer in the second frame. Middle: Tilting the camera has the same effect. Bottom: Fast movements of the UAV cause a duplicate detection.

indicated by its high IDF1 score and few lost tracklets (ML). They removed the newly introduced GSI and AFLink and added the part-based re-identification model PCB, which is pretrained on Market1501. This submission relied on the sub-optimal provided public detections. Moreover, in a second submission, STI-StrongSORT, they hypothesized that spatio-temporal information is more important than appearance-based information from a re-identification model. They based this hypothesis on the observation that objects have a very similar appearance and because occlusions are rare. With their proposed changes they manage to increase the speed from 10fps to 30fps. With a competitive HOTA score, they managed to decrease the number of ID switches fivefold.

MOT placed fourth as measured in HOTA, but placed third as measured in MOTA and IDF1. They based their submission also on DeepSORT. However, they trained their detector on the whole SeaDronesSee ODv2 dataset (train+val), which has a larger domain/appearance variance, but fewer (yet less correlated) images. Furthermore, their backbone is a ResNet-50 ($\sim$23M parameters), which is small compared to, e.g., a YOLOX-x with almost 100M parameters. They adapted to the aerial domain by setting appropriate scale parameters for the anchors and employed several train and test augmentation strategies while tuning respective hyperparameters. Similar to others, they set hyperparameters so as to ignore occlusion cases They set the detection score for updating tracks to 0, which may come from a similar motivation to that of ByteTrack (Zhang *et al.* (2022b)).

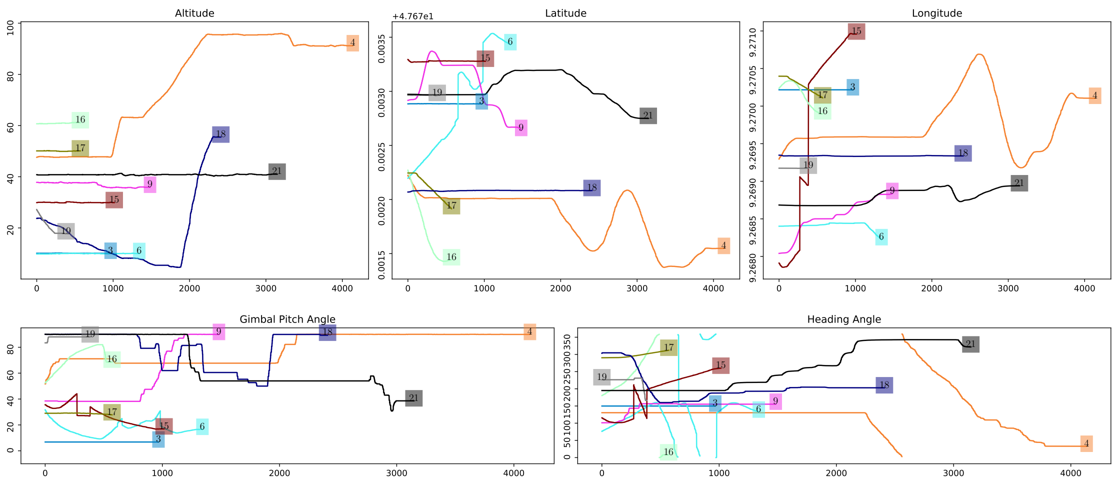


Figure 3.10: Metadata visualization of video clips with lengths longer than 300 frames. For shorter videos refer to Kiefer *et al.* (2023).

OCSORT placed fifth as measured in HOTA, while having the smallest amount of fragmentations and the highest precision. They also employ the large YOLOX-x detector and train on all of SeaDronesSee ODv2 and MOT.

Interestingly, there is no clear best model on the majority of the clips (Kiefer *et al.* (2023)). In the following, we try to explain some of the results on the clips, ordered from easiest to hardest clip.

In clip 19, only a single boat needs to be tracked which explains the high performances of all trackers. Similarly, clip 17 also shows only three boats which have to be tracked in a near static scene (compare to Figure 3.10). Clips 2 and 12 are also static scenes (Kiefer *et al.* (2023)). While clip 5 is also static (small altitude increase), the high altitude causes many trackers to not detect and track the small swimmers. Clip 15 also only features boats although some of them are further away in the horizon and the movement and heading rotation of the drone in addition to the camera pitch angle change cause some trackers to fail to reliably track. See also Figure 3.9 for examples of these errors. Clip 4 is the most dynamic one with camera and UAV panning and tilting and movement of the UAV in x,y and z directions. However, these movements are rather gently such that successful tracking can still be done by most of the trackers. Clip 21 features many swimmers and three boats. Furthermore, there are quick pitch angle changes along with a UAV movement and rotation. Clip 11 only shows a boat and a swimmer while the UAV is rotating around itself, although the swimmer is quite far away and hardly visible. Missing detections are punished relatively hard since the clip is short with few objects. While clip 0 is at very low altitude, there are many swimmer with a fast moving and rotating drone. Clip 16 shows many swimmers and boats and inherits a high dynamic range w.r.t. camera panning and tilting and movement of the UAV. Clip 13 shows a 90° scene where the UAV is moving at quite high altitude at constant speed. The swimmers are close to boats which is why it is hard to detect and track them. Clip 9 shows many swimmers with sudden changes in camera pitch and heading angle, resulting in many fragments and id switches. Clip 10 shows a few swimmers and several boats with a slow minimal camera pan. However, the acute angle lets swimmers appear very small and hard to detect. Similarly, clip 1 shows a scene with slowly rotating UAV and acute pitch angle, which results in many very far away swimmers that are quite small and are failed to detect robustly by most trackers. The hardest clip, 6, shows several boats and swimmers with a great amount of movement and dynamic camera panning and tilting. Furthermore, objects are hardly visible due to sun reflections.

Discussion

The submitted methods are already very strong. Many of the errors are still caused by very hard detections. However, the nature of UAV camera movements also cause several errors. Both, the detection and tracking errors could potentially be mitigated by using the available metadata.

The winner method from Beijing University, MoveSORT, leveraged a recent YOLOv7 detectors but included several modules to enhance the performance. The second place from National University of Defense Technology, byteTracker also employed the recent ByteTrack framework. The third place from EPFL, StrongerSORT, use the sub-optimal provided public detections to achieve the third place.

Further analysis would be necessary to discriminate based on classes and having real-time capable trackers with potentially worse but faster detection backbones. The necessity of certain tracking modules is also questionable in this setting, such as the reidentification module. Also, it is not clear how good the ECC module can really perform in the case of feature-poor maritime sceneries.

3.5 Conclusion

We discussed state-of-the-art baseline methods for multiple datasets and computer vision tasks. These serve as baselines throughout subsequent chapters. While the performances were quite well in some aspects, we will see that we can significantly improve on these results by considering metadata. From here on, the chapters discuss techniques to incorporate metadata to obtain more robust and domain-aware models. In fact, domain imbalance is a major problem, which we are able to tackle using metadata as follows.

Chapter 4

Diminishing Domain Bias

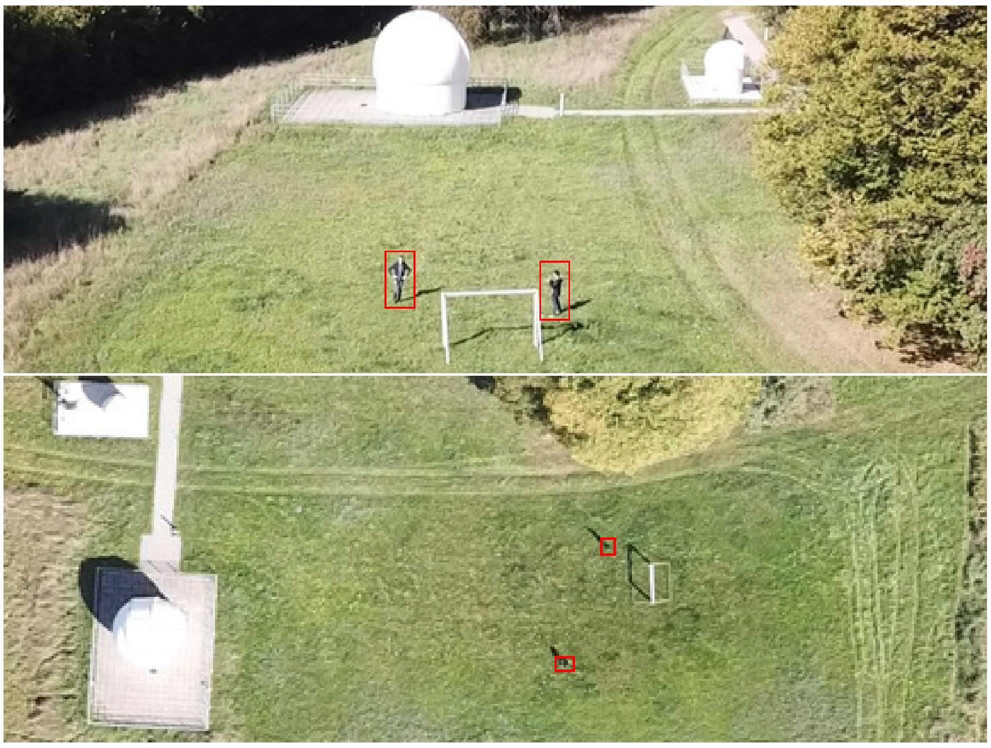


Figure 4.1: Example images of the dataset POG, showing the same scenery taken from different perspectives (top: 10m, 10°, bottom: 100m, 90°).

While generic object detection has improved drastically lately (Zhao *et al.* (2019b)), object detection on images captured from UAVs still poses great challenges (Zhu *et al.* (2020a)). Among these, the variation across domains is particularly challenging.

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. In lower altitudes, objects are captured with more detail while in 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. 4.1 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.* (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 4.2). 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 5.3.

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 this chapter, we discuss the following:

- 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 and on our datasets.

4.1 Related Work

Deep learning-based object detection can roughly be divided into two categories: accurate 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 (Redmon and Farhadi (2018)) or EfficientDet (Tan *et al.* (2020a)). 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.* (2018b)). Furthermore, a common practice in object detection from UAVs is still to use off-the-shelf detectors (Zhu *et al.* (2018a)).

With the release of the UAVDT (Du *et al.* (2018)) and VisDrone (Zhu *et al.* (2018a)) data sets, several works develop 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.* (2019a)).

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.* (2019a)), 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 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

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 4.1: 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.

labels at training and runtime to reduce the domain bias induced by highly imbalanced data sets.

4.2 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.

4.2.1 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 (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.

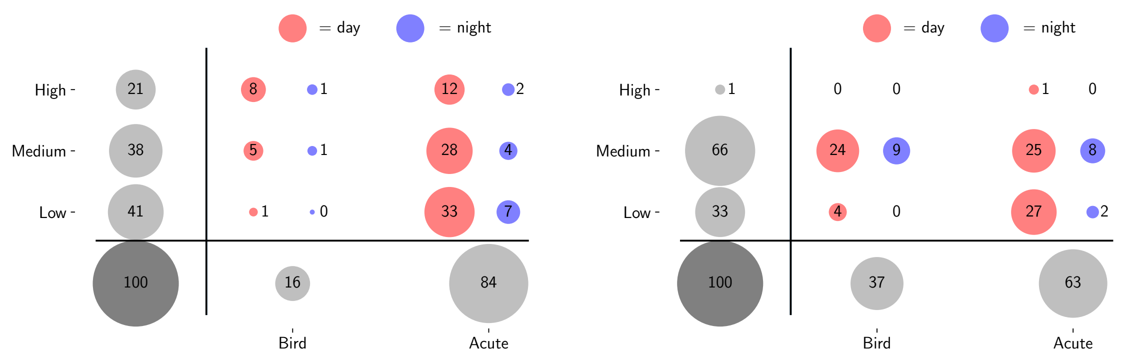


Figure 4.2: Distribution of objects across domains in the VisDrone (left) and UAVDT (right) training set. The lower left circle represents the size of the whole data set (100%), the other circles the relative size to it (rounded to the closest integer). The domains are high, medium, low, bird view, acute viewing angle, day and night and combination thereof.

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 4.1. Although these divisions seem arbitrary, they already help distinguish features in different domains, as will be seen in Section 5.3.

The large amount of varying domains causes data sets to be highly unbalanced with respect to these domains. Figure 4.2 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 5.2, we aim to propose a simple model family to diminish these biases.

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

$$\mathrm{mAP} := \frac{1}{C}\sum_{c=1}^{C}\mathrm{AP(c)} := \frac{1}{C}\sum_{c=1}^{C}\int_0^1 p_c(r)\mathrm{d}r, \tag{4.1}$$

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 4.3, 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, $\mathrm{mAP}_{d_1} = 0.1$ and $\mathrm{mAP}_{d_2} = 1$. 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

$$\mathrm{mAP}^{\mathrm{avg}} := \frac{1}{D}\sum_{j=1}^{D}\mathrm{mAP}_{d_j}, \tag{4.2}$$

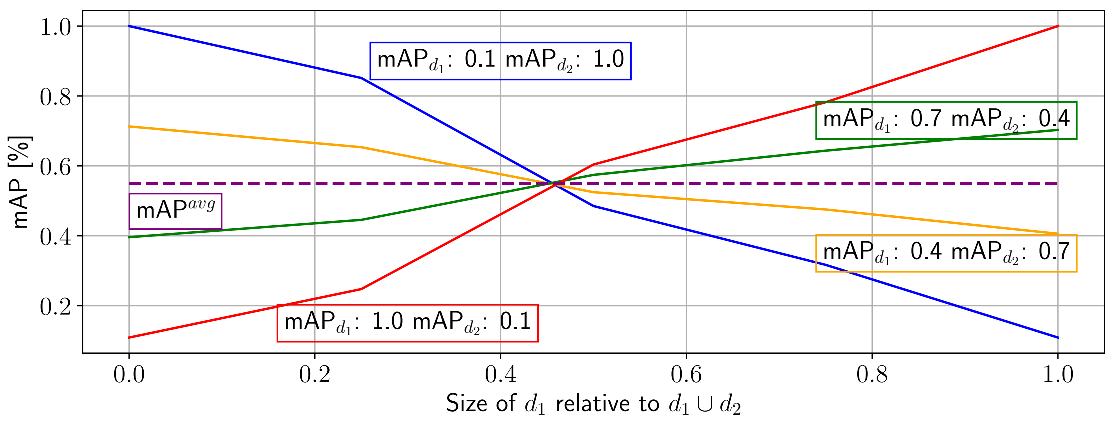


Figure 4.3: Hypothetical mAP values for a two-domain UAV object detection data set. mAP$^{\text{avg}}$ is the average mAP over both domains as defined in equation 4.2.

where mAP$_d$ denotes the mAP on domain $d \in \{d_1, \dots, d_D\}$. To obtain well-calibrated models, we evaluate on both, mAP and mAP$^{\text{avg}}$. 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$^{\text{avg}}$, which is independent of the the sizes of each domain.

4.3 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.

In multi-domain learning, image samples $\{x_j\}$ with corresponding bounding box annotations $\{y_j\}$ are accompanied by a discrete domain indicator $d_x \in \{d_1, \dots, d_D\}$ (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.

4.3.1 Toy Example

To illustrate the method, we fall back to a hypothetical one-dimensional example with two domains. Consider the true one-dimensional function f to be approximated as follows:

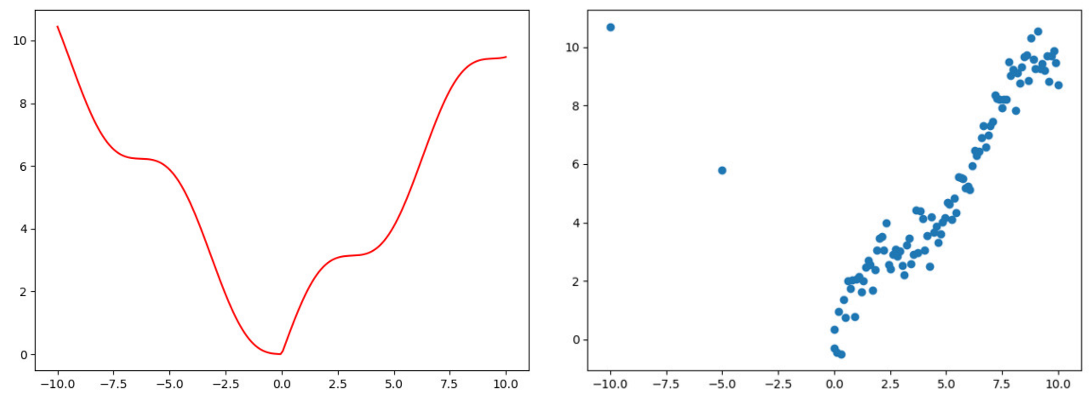


Figure 4.4: Left: True underlying function to be approximated. Right: Unevenly distributed data points sampled from the true function with noise where only a few data points in the left domain are sampled ($x < 0$ and many in the right domain ($x > 0$).

$$f(x) = \begin{cases} x + \sin(x), & \text{if } x > 0 \\ -x + \sin(x), & \text{otherwise.} \end{cases} \tag{4.3}$$

Naturally, in practical settings we only have access to individual data points. Therefore, suppose they are sampled around the true function via a Gaussian distribution with variance $\sigma^2 = 0.5$. Furthermore, to simulate a domain imbalance, suppose the data points are sampled very unevenly, such that we obtain the data points as shown in Figure 4.4.

Finally, assume your feature functions to be restricted to $f_1(x) = x$ and $f_2(x) = \sin(x)$, so that our hypothesis space can restore the true underlying function. Later, we will replace our hypothesis space with the space of all neural networks induced by specific architecture, on which we will experiment. For large neural networks, the assumption of having a powerful hypothesis space therefore seems to be justified. With that formulation, the question arises, how can we best learn the true function?

For that, we consider the following three learning strategies:

1. Learn on all data points globally, i.e. share all features across all domains;

2. Learn on each domain individually, i.e. share no feature across any domain;

3. Share sin feature across all domains and leave x other domain-specific.

Figure 4.5 shows the learned functions for the different training strategies, all fitted via gradient descent and the L_2 loss function. These are as follows:

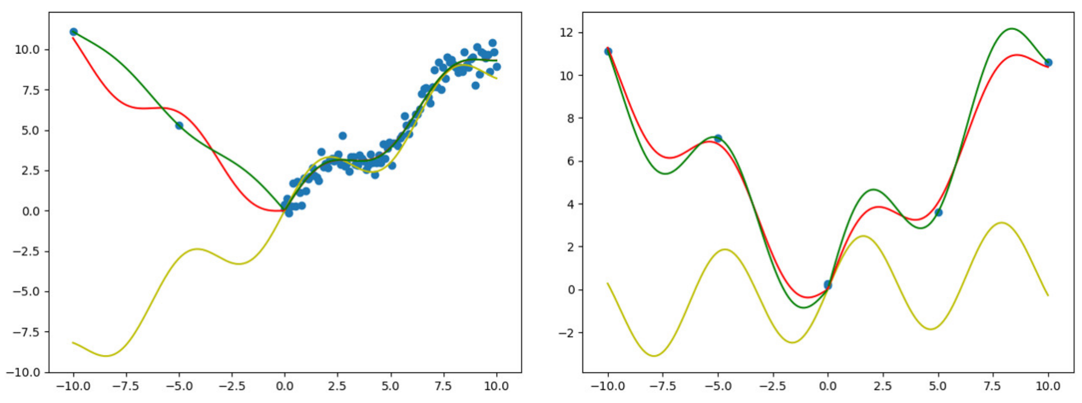


Figure 4.5: Left: Different training strategies on toy example. Yellow: Learn on all data points (dots in blue) globally (Strategy 1), Green: Learn on each domain individually (Strategy 2), Red: Share precisely one feature (Strategy 3). Right: The same experiment with balanced but very few data points.

$$\text{Strategy 1: } 0.9x + 1.6\sin(x) \tag{4.4}$$

$$\text{Strategy 2: } \begin{cases} -1.1x - 0.4\sin(x), & \text{if } x < 0 \\ x + 1.1sin(x), & \text{otherwise.} \end{cases} \tag{4.5}$$

$$\text{Strategy 3: } |x| + 1.6\sin(x) \tag{4.6}$$

We can observe that the capacity of the hypothesis space is too low in the case of the global training (Strategy 1). The learned function only fits the right domain since this accounts for most of the loss. While training strategy 2 almost succeeds in learning the true function on the right domain, it overfits to the data points on the left domain. Because of the lack of data samples, the model becomes overconfident on these points. The multi-domain learning approach successfully captures the sine-dynamic across both domains, while fitting the x-component correctly on the individual domains.

We would like to note that this problem can remain in balanced settings. For example, Figure 4.5 (right) shows the learned functions in the case of balanced, but very few data points. The global function still cannot fit this function due to a small hypothesis space, while strategy 2 overfit on the individual domains. Again, the multi-domain learning approach can leverage all data points for its sin-feature, resulting in a globally more exact solution.

4.3.2 Multi-Head Architecture

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. 4.1). Empirically, this is backed up by Wang *et al.* (2019a), 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.* (2019a). 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 5.3.

For illustrative purposes, see Figure 4.6. 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 chosen. 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 5.3.

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 5.3.

4.3.3 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.

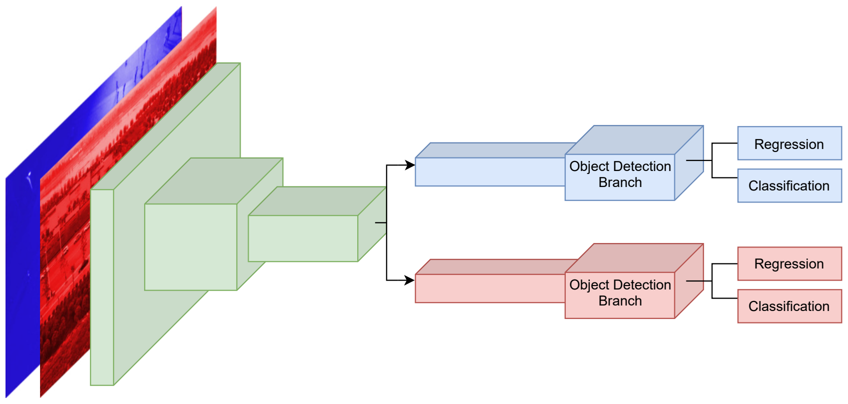


Figure 4.6: Illustration of a Time@3 model with day and night experts. The time is split into two domains, day (red) versus night (blue), where green outputs represent the shared stages (first, second, third). Every image is passed through the shared green stages. Then it is checked whether it is a day or night image and consequently passed through the red or blue stages, respectively.

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 5.3.

4.4 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

	L	M	H	mAP_{50}	mAP	mAP_{50}^{avg}	T
DE-FPN (Zhu *et al.* (2018a))	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.* (2018a))	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.* (2018a))	48.5	52.0		48.6	26.1	50.2	–
Time@4	**49.0**	**52.6**		**49.0**	**26.6**	**50.8**	7

Table 4.2: Several domain expert results for various freezing strategies on VisDrone. Altitude@x means that all stages until the xth. stage are shared.

finer domain splitting is possible in the case of the data set POG.

4.4.1 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 4.2.2, 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 T in percent (rounded to integers) to train a model longer than its baseline, i.e. $T = 10$ 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 (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

	$\downarrow + \rightarrow$	L	M	H	mAP_{50}	mAP	mAP_{50}^{avg}	T
DE-FPN	B	84.6	42.5	35.6	48.6	26.1	50.5	–
	A	49.1	50.0	41.2				
Alt.-ang.@4	B	**87.4**	**44.8**	**39.6**	**49.0**	**26.3**	**52.3**	10
	A	**49.7**	**50.1**	**42.2**				
DE-FPN	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	–				
Alt.-ang.-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 4.3: Results on specific domains for multi-dimension experts on VisDrone. 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).

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.* (2018a)), 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 4.2, 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 more than up until the second stage is 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. Lastly, the additional training time 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 4.3 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

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

Table 4.4: 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 4.5: EfficientDet-D0 Angle experts on VisDrone validation set

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 4.2).

This contrast is, furthermore, seen in underrepresented domains being improved the most, suggesting the diminished 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 4.2), from 51.6 mAP_{50} to 56.5 mAP_{50} and 35.6 mAP_{50} to 39.6 mAP_{50}, respectively.

Similar observations can be made from Table 4.4, 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 (PapersWithCode (2022a)), i.e. EfficientDet (Tan *et al.* (2020a)), 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.* (2020a)) domain-specific. As shown in Table 4.5, 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 21.8fps, suitable for live on-board processing. With all pre- and post-processing steps, we obtain a wall-clock time of 18.1fps.

	L	M	H	AP_{70}	AP_{70}^{avg}	T
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	–
Time@2	**53.4**	**54.1**		**50.1**	**53.8**	10

Table 4.6: Domain experts on the UAVDT testing set.

	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 4.7: Results for ResNet-101 backbone on UAVDT

4.4.2 UAVDT

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

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

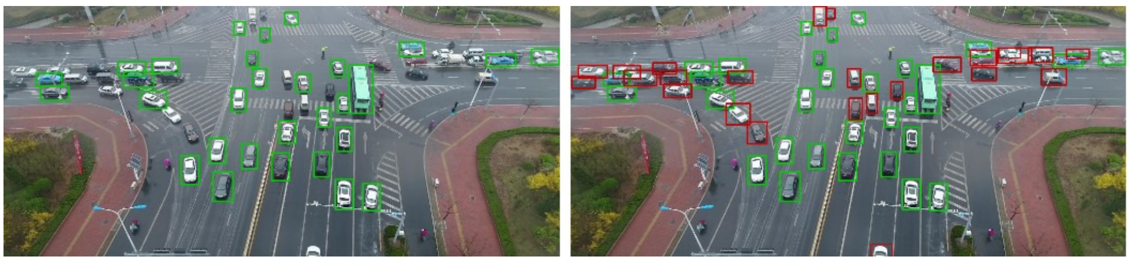


Figure 4.7: Example prediction of the altitude-expert model on a newly captured images.

	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 4.8: Altitude experts results on UAVDT test set

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 Table 4.2), 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 3.5fps faster. Note that they tested on different embedded hardware.

4.4.3 POG: Baseline and Expert Results

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 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, re-

	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 4.9: (Finer) Altitude experts results on POG test set

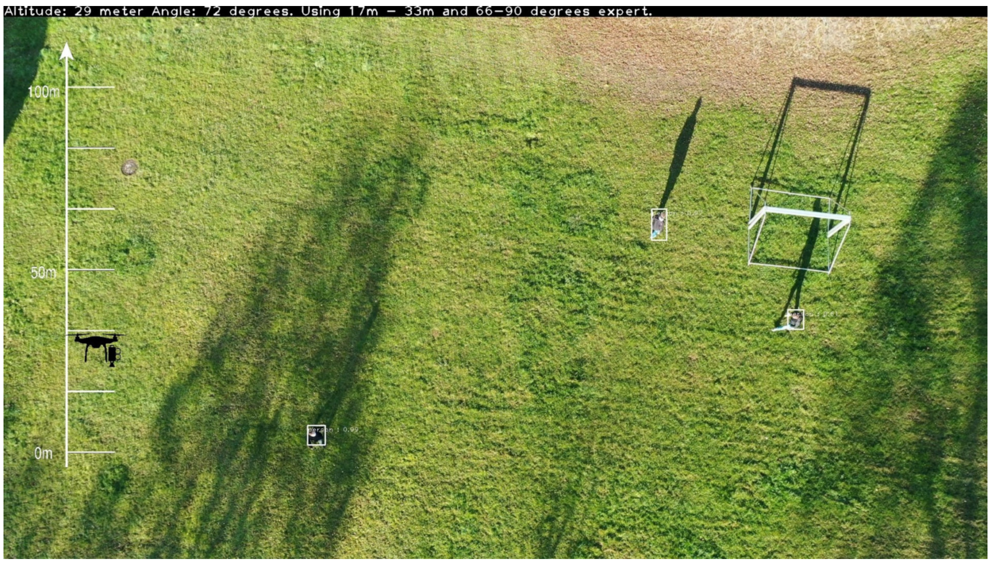

Altitude: 29 meter Angle: 72 degrees. Using 17m – 33m and 66–90 degrees expert.
100m
50m
0m

Figure 4.8: Example prediction of the altitude-expert model on a newly captured images.

spectively. 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). See Figure 4.8 for a visual demonstration of the expert model.

4.4.4 SeaDronesSee: Expert Results

For completeness, we also show that the expert models can improve the performance on SeaDronesSee. 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 4.10 shows, metadata can enhance the accuracy 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

Model	L	LM	M	MH	H	AP_{50}^{avg}
Faster R-CNN ResNet-50-FPN	**32.8**	29.8	23.5	40.5	**48.9**	35.1
5×Altitude@3	**32.8**	**29.9**	**26.2**	**41.5**	**48.9**	**35.9**

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

Table 4.10: SeaDronesSee results on different altitude- and angle-domains.

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 (compare to dataset statistics in Chapter 2).

4.5 Conclusion and Limitations

We successfully apply a multi-domain learning method to object detection from UAVs. We propose and analyze 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. In future work, it would be worth investigating how domains interact with each other on a deeper level. In this matter, incorporating softer boundaries between domains could be a promising direction. Furthermore, different sampling strategies, such as oversampling small domains, could be investigated.

Lastly, one key observation is that altitude variances have a great influence on the performances of an object detector. Therefore, in the next chapter, we will discuss the specific case of obtaining scale invariance.

Chapter 5

Gaining Scale Invariance

While the previous chapter discussed the case of reducing the domain bias, it was formulated to apply for any general domain. However, we saw that *altitude* plays a critical role in the performance of an object detector. Therefore, in this chapter, we focus on the specific case of obtaining scale invariance to alleviate the performance degradation due to altitude changes.

UAVs with mounted cameras are used in versatile application areas, 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.* (2018a)), while in search and rescue tasks, the span may be as large as 5 to 260 meters (Varga *et al.* (2021)). 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 vital to include this 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 different 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

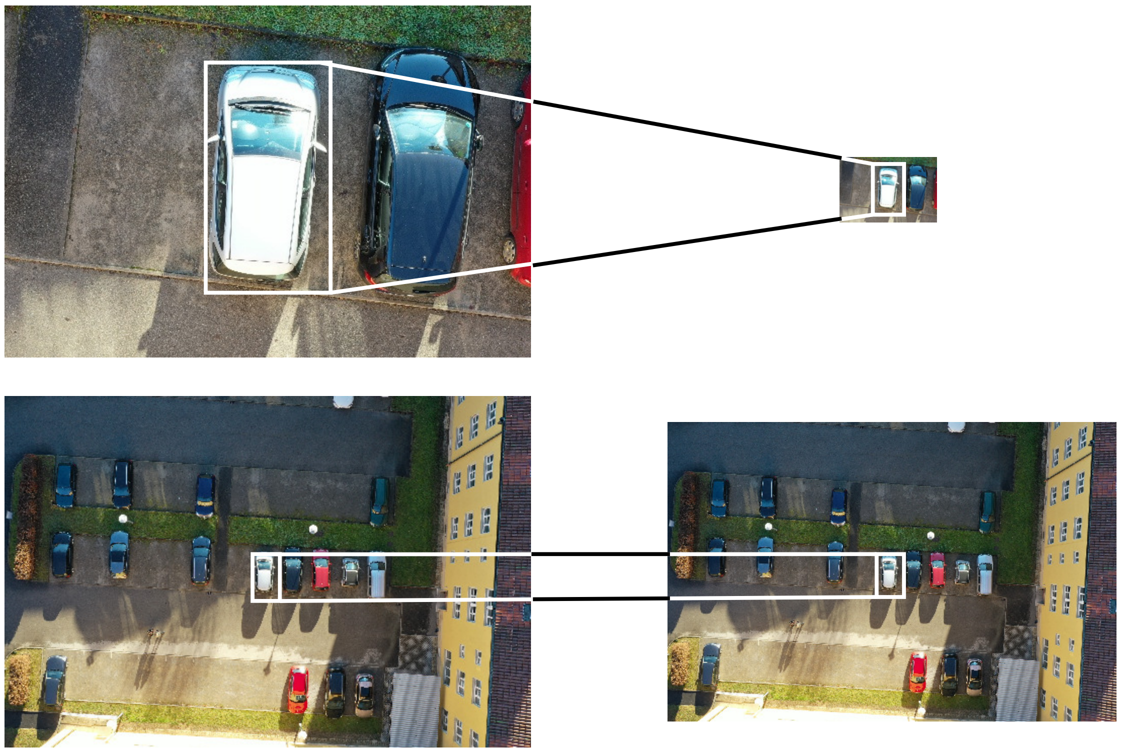


Figure 5.1: Example of the resizing process. Top images are captured at $10m$ flight altitude, the bottom ones at $60m$. On the right, we resized both images according to their respective height. Note how the bounding box of the silver car at the center of each respective picture is equal in size after resizing.

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, our key contributions 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.

5.1 Related work

Object detectors can broadly be divided into two categories; one-stage and two-stage detectors. Two-stage detectors (Ren (2016); Girshick (2015b)) 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.* (2019b); Tian et. al. (2019); Lin et. al. (2017b)), 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 viable 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); Redmon and Farhadi (2018)) or on mobile platforms (Ringwald et. al. (2019)) design a whole network architecture. On the other hand, we introduces a method applicable to most modern object detectors, improving their speed and detection performance.
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.* (2021b)). 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.*

(2021b)) aims at generic object detection while we go for the special case of BEV object detection.

5.2 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.

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 5.1. 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) but at different scales (Lin et. al. (2017a)). Therefore, it learns different representations for different scales of the same object. That means some of the capacity of the detector is wasted on 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 (making every object of one class of same size over full data set) 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 assume a fixed camera setup on the UAV. We can readily deduce the following formula from fundamental properties of the camera geometry (see Figure 5.2):

$$\text{GSD} := \frac{g}{s} = \frac{h}{f}, \tag{5.1}$$

where s refers to the image side length in pixels, h denotes the height of the UAV, f refers to the camera's focal length in pixels, and g denotes the ground length in meters

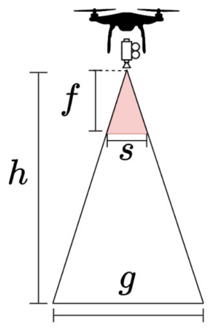


Figure 5.2: Illustration of the ground sample distance.

that is covered. With a fixed camera setup, the only varying factors in Equation (5.1) are the distance above ground h and the image size s. Therefore, if we ensure that the ratio $h/s =: C$ 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.

Furthermore, 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. 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 $4m \times 2m$ while our reference area is 32 px × 32 px, we obtain the desired GSD as follows:

1. First, we compute the reference area with the same aspect ratio as the average car. Here, this is roughly 45.25 px × 22.63 px. Then the desired GSD is

$$\frac{4}{45.25}{}^{m}/_{\text{px}}.$$

2. Inserting this into Equation (5.1), we obtain the image size to resize to by solving for s.

If we do not know the size of the average car in our data set a priori, we may estimate it by averaging the areas 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 $\tilde{s}$ 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

$$s = \frac{r}{m} \cdot \tilde{s}. \tag{5.2}$$

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 5.1. This method works together with any modern deep learning-based object detector since our approach is a preprocessing step.

If there is a heavy lens distortion, we first undistort the image using the camera intrinsics. We refer to Ojanen (1999) for an elaborate description. In our experiments, we disregard effects of lens distortion and perspective projection as these are minor compared to the general relation of altitude to object size.

5.2.1 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–$D0$ (Tan *et al.* (2020a)) 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 onto EfficientNet-backbones (Tan *et al.* (2020b)) and are therefore scalable in parameters, ranging up to EfficientDet–$D7$. Here, a higher number means that the model is larger and more accurate, while a lower number means that 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 (PapersWithCode (2022a)).

EfficientDet–$D0$ employs a Feature Pyramid Network (FPN) (Lin et. al. (2017a)), as is standard for modern object detectors. 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 (5.4). 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 $(32, 64, 128, 256, 512)$.

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 5.4. 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

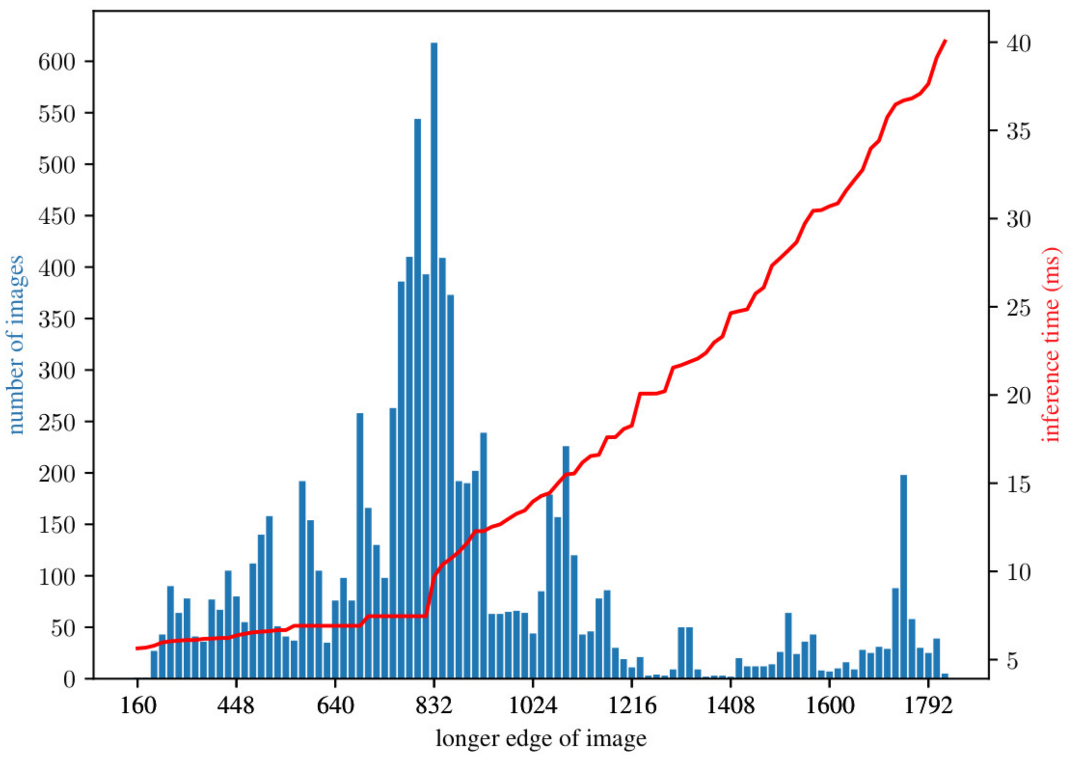


Figure 5.3: Distribution of image sizes after applying Adaptive Resizer on the UAVDT data set and the resulting inference time. The *x*-axis denotes the longer respective edge of the image, aspect ratios are kept during this process. The *y*-axis denotes the quantity in blue and the inference time in red.

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 5.3.

5.3 Experiments

We employ Faster R-CNN (Ren (2016)), CenterNet (Zhou *et al.* (2019b)), and EfficientDet–*D*0 (Tan *et al.* (2020b)) 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-

or ResNeXt-backbones (He *et al.* (2016); Xie *et al.* (2017)). 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.

5.3.1 Results on Bird's Eye View Portions

We conduct our experiments on two well-known UAV data sets, VisDrone (Zhu *et al.* (2018a)) and UAVDT (Yu et al. (2020)), and on POG. 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. Of these datasets, only POG features precise altitude annotations. We estimate the altitudes of the other two datasets as discussed below.

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 (3 levels of altitude, see Chapter 4), we generate finer height data artificially for UAVDT and VisDrone. We do so using the second method from Section 5.2. More precisely, we generate the image sizes by Equation (5.2). The data set contains images in between $10m$ and $110m$.

For the experiments on one-stage detectors, we employ EfficientDet–$D0$ and CenterNet as described in their respective original papers (Tan *et al.* (2020b); Zhou *et al.* (2019b)). 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 5.1 and 5.3. For both models, we observe that employing Adaptive Resizer improves inference speed by a factor of two to three, see also Section 5.3.4. In the case of EfficientDet (Table 5.1), 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 5.3), the 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 to the baseline.

We also include results for the Adaptive Resizer on two-stage detectors to see the potential of adaptive resizing on larger 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.* (2018a)). 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). Table 5.2 shows 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 (compare to Chapter 4). We observed that capture-altitudes are on average considerably 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 suffer greatly, see Section 5.3.5. Additionally, employing adaptive resizing speeds up inference by a factor of two to three.

	VisDrone	UAVDT	POG	FPS
$D0$ @ 2048	13.1	34.1	80.3	12
$D0$ @ 1792	17.7	30.0	74.3	15
$D0+$Adaptive	**20.6**	**58.8**	**83.0**	**32**

Table 5.1: AP_{50} results on the bev portions of the data sets. EfficientDet–$D0@x$ 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 RTX 2080 Ti GPU.

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

Table 5.2: 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)).

5.3.2 Effects of Cutting FPN

In this section, we compare the detector from Section 5.2.1, meant for fast inference and deployment to an embedded GPU, to a full-fledged EfficientDet–$D0$ model with Adaptive Resizer. The model from Section 5.2.1 has no feature pyramid network and

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

therefore only uses one feature map. Table 5.4 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 5.2.1 and Figure 5.4) 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 upper bound 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. Table 5.4 shows that after applying non-maximum suppression, on average less than two percent of all object per image are not detected by the first feature map.

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

Table 5.4: 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 FPN and Adaptive Resizer.

5.3.3 Results on Complete UAVDT Data Set

To also introduce a model that works on a full UAV data set, we use a multi-domain approach following (Kiefer *et al.* (2021)). More explicitly, we use the meta-data supplied

	UAVDT	VisDrone	POG	FPS
$D0$–noFPN	49.3	**23.6**	79.2	**56**
$D0$–FPN	**58.8**	20.6	**83.0**	32

Table 5.5: AP$_{50}$ results on UAVDT, VisDrone, and POG. The compared models are EfficientDet–$D0$ with Adaptive Resizer. $D0$–noFPN is a model without FPN like described in Section 5.2.1, $D0$–FPN is the standard model with Adaptive Resizer, including fpn.

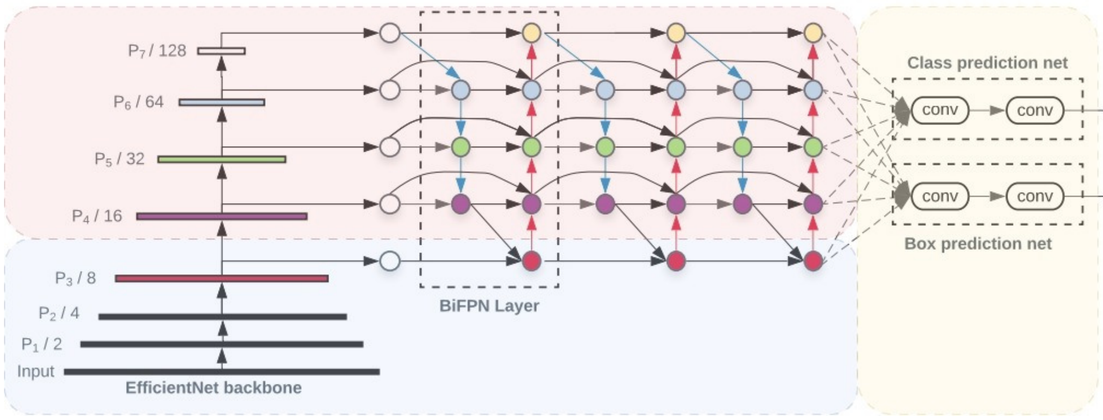


Figure 5.4: Adapted original illustration of an EfficientDet–$D0$ (Tan *et al.* (2020a)). Starting from the image, the backbone-network extracts feature maps P_1, P_2, P_3 (left part of blue box). Then these are input to the feature pyramid network $P4 - P7$ (red) and afterwards handed to the heads (yellow). These perform classification and regression. The object detector without FPN from Section 5.2.1 is encircled with the dashed line. The bulk from the red box is for dealing with scale variance in general settings. We can remove this using adaptive resizer.

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 on all other images. Both are loaded before inference and available in GPU memory, so there is a slight overhead added and no drop in inference time for each of the models. To achieve the results reported in Table 5.6 on UAVDT, we use the models from Table 5.2 for the two-stage detector experiments. For the experiments with EfficientDet–$D0$ we use the model without FPN from Section 5.2.1 and the baseline from Table 5.1.

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 5.6: 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).

5.3.4 Timing Benchmarks

Tables 5.1 and 5.3 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 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, being the least prone to statistical outliers during the benchmark test. Table 5.1 and 5.5 show, where the speed improvement of the EfficientDet–*D*0+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.* (2020b)). Because convolutional networks are fully-connected in the channel-dimension, cutting these brings the largest speed improvement.

Figure 5.3 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–*D*0 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 5.1. Therefore, on embedded hardware, Adaptive Resizer improves inference speed by a factor of two to three.

5.3.5 Height Transfer

Without Adaptive Resizing, a model learns different representations for object instances with varying scales, as discussed in Section 5.2. Therefore the model learns separate representations for objects belonging to the same class but appearing on images from different altitudes. In simple terms: the objects on which the model without Adaptive Resizer did not train can not be recognized during testing.

On the other hand, 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 in the discussed fashion. Essentially, the Adaptive Resizer allows for transferring knowledge in between altitudes. An example: our network can learn from images taken between $0m$ and $50m$ above ground, and then perform well on images captured between $50m$ and $100m$.

To prove these claims, 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 5.7 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 5.1 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 basically en par with its performance on the whole BEV split of UAVDT, e.g. 47.9 versus 49.3 AP^{50} (see Table 5.5).

5.4 Conclusion and Outlook

In this work, 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

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

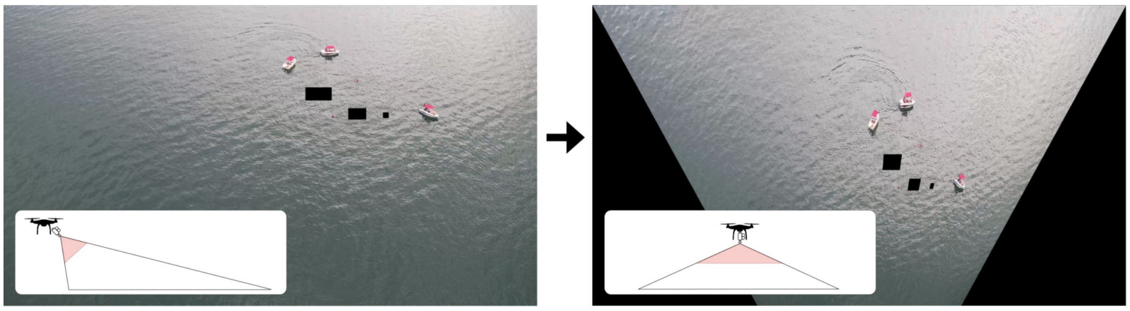


Figure 5.5: Possible extension of adaptive resizing to apply on non-BEV imagery.

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 seen before. Furthermore, we used a multi-task fashioned approach to capitalize on our method on generic UAV imagery.

An immediate follow-up question would be whether we can loosen the BEV restriction and apply adaptive resizing to acute viewing angles. E.g., we could apply the bird's eye view transformation on any image with acute viewing angle as illustrated in Figure 5.5. However, several apparent problems arise: Objects become skewed, so that objects that are far away make up large parts of the resulting image. The corresponding bounding box hence is much larger, breaking the assumption that objects of the same class should have roughly the same size. It remains to be shown by future works whether this can be alleviated. We discuss more on the underlying 3D geometry in Chapter 8.

The BEV assumption also revealed that there is a great lack of large-scaled UAV object detection datasets. Hence, in the next section, we discuss how we can leverage synthetic data and, again, show how metadata helps to construct higher quality datasets.

Chapter 6

Obtaining Efficient Synthetic Data

Although object detection on natural images taken from hand-held or car-mounted cameras has been studied intensively, object detection from UAVs trails behind in performance (Zhu *et al.* (2018a)). This is partly due to the limited amount of annotated publicly available data sets. In turn, this is partly caused by the highly complex data generating missions, which are subject to permissions, UAV flying restrictions and environmental factors (Varga *et al.* (2022)). Furthermore, there are more degrees of freedom in the UAV domain (camera angles, position), which account for objects from unnatural perspectives, e.g. small objects from above. These difficulties are on top of other common obstacles, such as high and enduring labeling costs.

With more publicly available data sets, object detection on UAVs could be improved. However, data collection and labeling are expensive and time-consuming. Furthermore, data set collection raises serious privacy (e.g. GDPR (GDPR (2022), ExposingAI (2022)) and security concerns because specific locations demand a long and complicated approval procedure.

Moreover, currently published data sets suffer from large class and domain imbalances (Du *et al.* (2018); Zhu *et al.* (2018a); Messmer *et al.* (2021)). Both problems are inherently caused by a problematic capturing procedure, where many variables cannot be controlled. For example, in UAVDT, the classes ”truck” and ”bus” make up only approx. 5% of all objects, and only 8% of all images are captured from high altitudes (Du *et al.* (2018)). Other data sets also suffer from these imbalances (Zhu *et al.* (2018a); Varga *et al.* (2022)).

On the other hand, synthetically generated data for computer vision problems can help train data-demanding visual perception systems because it is comparably fast and inexpensive to acquire. It also allows access to ground truth annotation data. Furthermore, this data can easily be tailored to specific requirements. Several works address synthetic data generation in computer vision. However, most of them focus on driving, simulating constrained traffic situations (Kar *et al.* (2019); Srivastava *et al.* (2019); Hurl *et al.* (2019)). Few works consider the generation of synthetic data captured from UAVs. However, these only focus on the capturing process of the sensors and are lacking in world, object and physics details, or do not feature them at all (Kar *et al.* (2019); Fonder and Droogenbroeck (2019)).

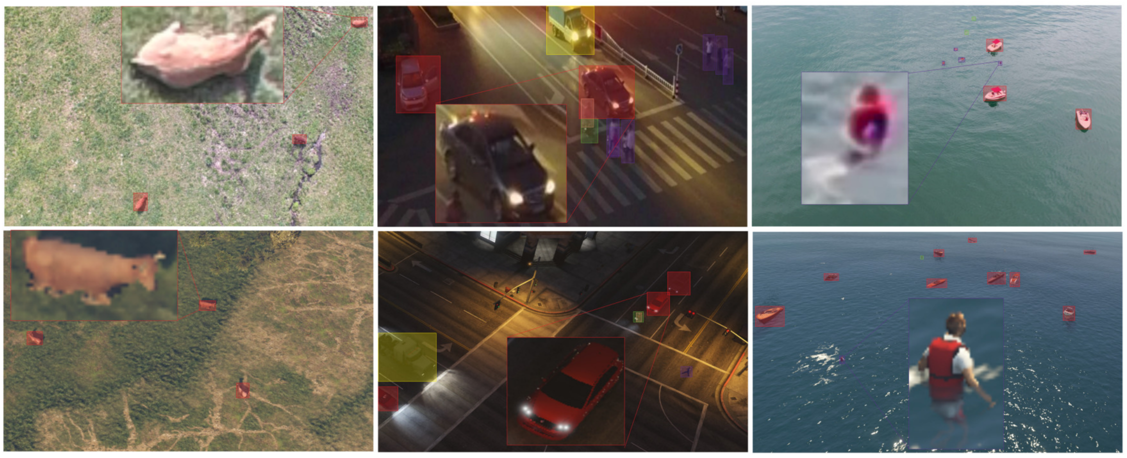


Figure 6.1: Real (top) and synthetic (bottom) image samples in different applications scenarios with ground truth annotations. Representative objects are magnified.

While the use of synthetic data in the context of autonomous driving has been investigated thoroughly, it is not clear whether these findings can be applied in the UAV setting. As mentioned in the beginning, the problems for general UAV object detection also apply to synthetic data generation. Most simulation engines focus on autonomous driving, and therefore rendering is aimed at looking realistic for these scenarios. Even if these simulation engines could technically be adapted to the UAV setting, light conditions, shadows, visibility range, rendered resolution, and more may significantly affect the quality of the rendered footage. In turn, the generated data may be less valuable for transfer to the real world.

In this chapter, we consider the video game Grand Theft Auto V (GTAV) (Games (2022)) as a simulation platform. It offers numerous detailed object models that interact in a large world with realistic graphics and physics simulations. Building on previous works (Johnson-Roberson *et al.* (2016); Angus *et al.* (2018); Yue *et al.* (2018); Hurl *et al.* (2019)), we extend the DeepGTAV data extraction tool to work for airborne scenarios by extending its functionalities regarding in-game agent and camera position and rotation, environment manipulation, object spawning and metadata extraction.

Using this simulation engine, we create three large-scale high-resolution (4K) synthetic object detection data sets in different application scenarios. Using these, we evaluate different training strategies and their transfer performances on three corresponding real-world data sets. We provide analytical insights and actionable advice on which settings to choose by doing extensive experiments and ablations.

In this chapter, we discuss the following contributions:

- We modify and extend the DeepGTAV tool, in particular, to allow the production of airborne data. We discuss the specific improvements in Section 6.4.

- We provide three large-scale high-resolution metadata annotated data sets in different UAV application scenarios and make them publicly available.
- We evaluate the applicability of these data sets to improve real-world object detection and analyze the from-scratch performance.
- We analyze the influence of different parameters of the data generation, e.g. the graphics quality and the alignment of metadata.

6.1 Related Synthetic Data Generation Engines

Many data generation engines focus on autonomous driving. AirSim (Shah *et al.* (2018)) and Carla (Dosovitskiy *et al.* (2017)) leverage the Unreal Engine (Qiu and Yuille (2016)) to create a simulation engine suitable for autonomous driving scenarios. While AirSim also provides support for UAV scenarios, it lacks a world to simulate objects and its physics implementation, which must be created and modeled first. Instead of laboriously creating simulation engines, some researchers use the computer game GTAV (Games (2022)) to generate data. With DeepGTAV (Johnson-Roberson *et al.* (2016)), researchers leverage GTAV to gather data for autonomous driving. PreSIL (Hurl *et al.* (2019)) builds upon this approach to refine the data acquisition process with their tool DeepGTAV-PreSIL. However, both systems lack the feature to modify the in-game agent and camera position and rotation, essentially limiting it to a single autonomous driving scenario. Further manipulations, such as spawning objects, manipulating the environment and extracting the metadata, are not possible either. The work at hand builds upon these two works, improving and extending them to open it for UAV research.

There are many more synthetic, simulated environments. Please see Nikolenko *et al.* (2019) for an overview.

6.2 Related Data Sets Taken on UAVs

The first large-scale real-world object detection data sets taken on UAVs were VisDrone (Fan *et al.* (2020b)) and UAVDT (Du *et al.* (2018)). Other data sets for object detection and tracking emerged with many different foci (Pei *et al.* (2019); Mueller *et al.* (2016); Hsieh *et al.* (2017); Mundhenk *et al.* (2016); Li and Yeung (2017); Krajewski *et al.* (2018); van Gemert *et al.* (2014); Ofli *et al.* (2016); Varga *et al.* (2022); Bozcan and Kayacan (2020)).

The need for data sets caused many researchers to focus on synthetic data. Mid-Air (Fonder and Droogenbroeck (2019)) presents a synthetic data set for unstructured environments captured with the Unreal Engine (Qiu and Yuille (2016)) in combination with AirSim (Shah *et al.* (2018)). They laboriously built a world with landscape and streets for drone navigation. However, their world is lifeless and does not feature any

Data Set	Domain	Type	#Images	Widths	Alt.	Ang.	O.
VisDrone	traffic	real	10,209	2,000	×	×	×
AU-AIR	traffic	real	32,823	1,920	✓	✓	✓
PreSIL	traffic	synth.	40,000	1,920	–	–	×
DGTA-VisDrone	traffic	synth.	50,000	3840	✓	✓	✓
Airbus Ship	marine	real	40,000	768	–	–	×
SeaDronesSee	marine	real	5,630	3,840	✓	✓	✓
DGTA-SeaDronesSee	marine	synth.	100,000	3,840	✓	✓	✓
Cattle	agriculture	real	670	4,000	×	×	×
DGTA-Cattle	agriculture	synth.	50,000	3,840	✓	✓	✓

Table 6.1: Comparison with the most prominent annotated aerial object detection data sets in three domains.

objects of interest. The Synthinel-1 (Kong *et al.* (2020)) data set features synthetic data showing building footprints with segmentation masks.

An overview comparing the most important data sets is shown in Table 6.1.

6.3 Methods Narrowing the Sim-to-Real Domain Gap

On the model level, many approaches apply synthetic-to-real domain adaptation (Hoffman *et al.* (2018); French *et al.* (2017); Li *et al.* (2018a); Zou *et al.* (2018); Chen *et al.* (2018); Prakash *et al.* (2019); Tobin *et al.* (2017); Tsai *et al.* (2018)). By disentangling the features, these techniques aim to learn domain invariant features that lead to better transfer capabilities. A subset of these methods performs image stylization to make the synthetic images look more similar to their real-world counterparts (Richter *et al.* (2021); Zhu *et al.* (2017a)), referred to as narrowing the appearance gap. Another domain gap arises from differences in content, called content gap (Kar *et al.* (2019)). It depicts the layout and types of objects featured in the synthetic world instead of the real world.

Orthogonal to these gaps, we focus on another gap called meta gap. Meta gap depicts the imaging conditions at the time of capture, such as altitude, viewing angle and time. It can be seen as a useful abstraction of the content gap. These metadata are freely available in the synthetic engine and on an actual UAV. We leverage these metadata to align the distributions of the synthetic and real-world data set, leading to better performance and data efficiency.

6.4 DeepGTA-UAV: Tool Description and Improvements

The DeepGTAV framework as used in this work builds upon the DeepGTAV-PreSIL framework (Hurl *et al.* (2019); Hurl (2022)), built upon the DeepGTAV framework (Ruano (2022a)).

Originally, DeepGTAV was built as a reinforcement learning environment for self-driving cars, providing functionality to interact with GTAV through a TCP-server and building upon the functionality of SkriptHookV (Blade (2022)).

The Python library VPilot (Ruano (2022b)) interacts with DeepGTAV, which runs in GTAV.

DeepGTA-PreSIL integrates DeepGTAV and GTAVisionExport (Barto (2022)),

the technique presented in (Johnson-Roberson *et al.* (2016)), to extract depth and stencil buffers from the rendering pipeline. With those and the world coordinates of objects extracted in GTAV, pixel-wise object segmentation data can be extracted. From those, object bounding boxes can be calculated. The DeepGTAV framework also allows extracting voxel-wise LiDAR segmentation data, which was not used in this work. The most notable improvements made in this work include:

1. Increasing the ease of use and the multitude of scenarios of which synthetic data can be generated by improving the VPilot interface of DeepGTAV. In particular, it is now possible to freely modify the in-game position, camera position and rotation, manipulate the environment (time of day, weather), spawn objects (e.g. pedestrians, cars) and specify their animations.

2. Improving the speed and reliability of the DeepGTAV framework (to obtain almost no overhead, compared to running GTAV natively).

3. Allowing the extraction of metadata of the generated data (like time of day, height, camera angle and others).

4. Providing multiple easily comprehensible and modifiable data generation scripts for several airborne scenarios and different strategies of metadata distribution.

5. Modifications to capture 4k image data (although we did not analyze the influence of high resolution data).

In this work, those adaptations were used to capture object detection data from a UAV perspective (in comparison to an autonomous car scenario in previous works (Johnson-Roberson *et al.* (2016); Hurl *et al.* (2019))).

For optimal use as a simulation environment, some modifications were made to the GTAV game files by installing the modifications *Simple Increase Traffic (and Pedestrian)* (Dorahamu (2022)), *No chromatic aberration lens distortion* (VenJam1n (2022)) and *Heap Limit Adjuster* (Division (2022)). Additionally, the automatic spawning of objects was modified to match the class distribution in VisDrone.

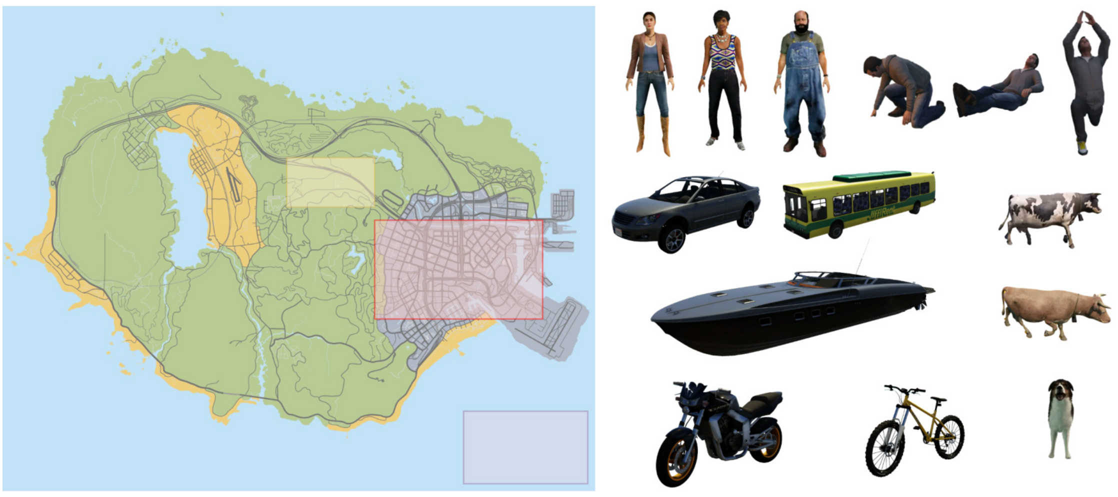


Figure 6.2: Left: Map used in GTAV and areas where datasets were taken (Red: VisDrone, Yellow: Cattle, Purple: SeaDronesSee). Right: Example in-game assets; 3 out of approx. 1000 people; 3 out of over 150k animations; 5 out of approx. 800 vehicles; 3 out of 32 animals. Each object has various manifestations (different colors, clothing,...).

Furthermore, the bounding box quality was improved by setting the game resolution to 7680x4320DSR in NVIDIA GeForce Experience (Nvidia (2022a)). This results in an upscaling of the rendering buffers to 4k, which is needed to obtain pixel-perfect object segmentation data for 4k images.

6.5 Data Set Generation

To assess the usefulness of synthetic data as training data for real-world scenarios, particularly to assess the usefulness of DeepGTAV and its adaptability in this context, we examined three different object detection scenarios.

In choosing those three scenarios, we are bound to existing real-world data sets to assess the real-world performance. We find that the data sets VisDrone (Zhu *et al.* (2018b)), SeaDronesSee (Varga *et al.* (2022)) and Cattle (Shao *et al.* (2020)) are very popular in their respective application domain and therefore choose these. VisDrone aims for traffic surveillance in Asian cities with very crowded scenes. SeaDronesSee's application domain is search and rescue in open water featuring swimmers and boats, with the main challenges being reflective regions, shadows and waves or seafoam. Finally, Cattle aims to bring autonomous vision systems to agriculture (cattle detection).

Using the DeepGTAV framework, we generate synthetic training data for these scenarios by specifying VPilot data generation scripts for each scenario. In the following, we briefly describe the different capturing procedures employed in the VPilot scripts used to generate the synthetic data sets.

	DGTA-Cattle	DGTA-SeaDronesSee	DGTA-VisDrone
GPS $\times 10^3$	$[0,17]\times[13,22]$	$[-28,-18]\times[-25,-13]$	$[-12,14]\times[-22,13]$
Spawn	4×cow@2s	4×ppl@2s, 4×boat@2	3×bike@8, 1×motor@8
Altitude	10-80m	0-80m	0-40m
Cam Pitch	20-90°	20-90°	20-90°
Spawn y	50-250m	50-250m	50-150m
Spawn x	-160-160m	-160-160m	50-150m

Table 6.2: Data set generation settings. "Spawn" refers to spawning in addition to the in-game spawning.

For the generation of all synthetic data sets, the game world of GTAV is systematically traversed. New images are captured with one frame per second to obtain mainly distinct images. We export the 4k images but discard the segmentation and depth maps to focus on pure object detection. Along with every frame, we export the corresponding ground truth bounding boxes and the meta labels, i.e. altitude, principal axes (yaw, pitch, roll of camera rotation), time of the day and weather state.

For the random traversals of the game world, the camera height and angles were varied. Additional in-game objects were spawned (e.g. vehicles, pedestrians). See Table 6.2 for details. For example, in Cattle, every two seconds, four cows are spawned 50-250m in front of the camera with a left-right offset of -160-160m. The objects were spawned in the in-game traffic and pathfinding and were despawned after 200 seconds. Additionally, a new random travel location in this area was chosen every 60 seconds to prevent the in-game navigation from getting stuck. Finally, see Figure 6.2 for an overview of the map and example in-game assets that were used for the creation of these datasets.

Concise descriptions of the data sets are given in Tables 6.1 and 6.4.

6.6 Models and Training setup

As object detection models, we take two one-stage real-time detectors, EfficientDet-$D0$ (E.-$D0$) (Tan *et al.* (2020a)) and Yolov5 (Redmon *et al.* (2016); Redmon and Farhadi (2017, 2018); Bochkovskiy *et al.* (2020); Jocher *et al.* (2020)), both being on the forefront of real-time object detectors as measured by their performances on the COCO test set (Lin *et al.* (2014); PapersWithCode (2022b)). In particular, EfficientDet-$D0$ is the state-of-the-art model for real-time detectors on the large-scale UAVDT traffic surveillance data set (Du *et al.* (2018)). For that, we use the implementation from GmbH (2020) with an image size of 2176px width and anchor scales of (0.3 0.5 0.7).

Yolov5 (Jocher *et al.* (2020)) is a state of the art implementation of the Yolo object detection model implemented with multiple improvements to the Yolo framework that

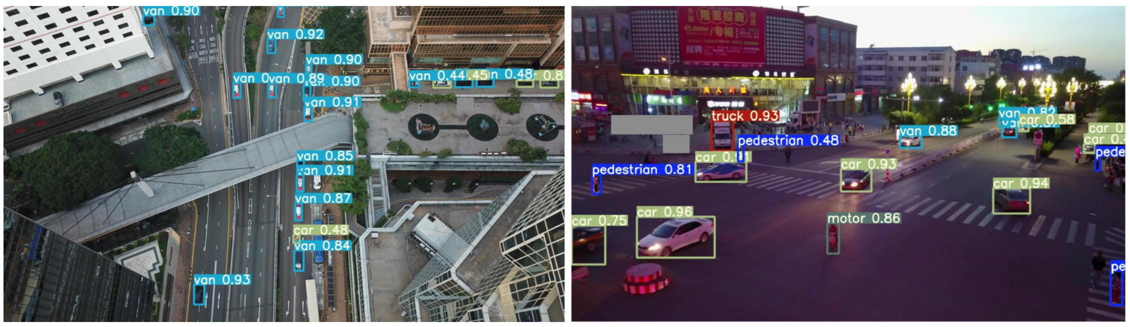


Figure 6.3: Example predictions on VisDrone of a YOLO model purely trained on synthetic DGTA-VisDrone (pretrained on COCO).

have been found in recent years. In this work, we used the unmodified YOLOv5m6 implementation of Yolov5 in release v5.0 (Jocher *et al.* (2020)) with an image size of 1280x1280px and a batchsize of 48. Unless otherwise specified, we used the provided weights pre-trained on COCO (Lin *et al.* (2014)).

Furthermore, as a two-stage detector we take the best performing single-model (no ensemble) on VisDrone from the workshop report (Zhu *et al.* (2018a)) (DE-FPN), i.e. a Faster R-CNN (F.R.) with a ResNeXt-101 64-4d (Xie *et al.* (2017)) backbone (removing P6), which is trained using color jitter and random image cropping. The anchor sizes and strides are decreased to (16, 32, 64, 128, 256) and (4, 8, 16, 32, 64).

We measure the models' performances on the popular mean average precision metric with overlap 0.5, i.e. mAP@0.5 (Lin *et al.* (2014)). As we are interested in real-world performance, we test on the test set of the real-world data set unless indicated otherwise.

6.7 Experimental Evaluation

First, we conducted different experiments to show that synthetic training data could yield good real-world performance or improve the performance of a real-world object detector. Then we conducted further ablation studies to examine different factors that could influence or modulate the positive effect of synthetic training data.

On a general level, we wanted to obtain actionable advice for an engineer using synthetic data to train an object detector, which factors should be examined with emphasis and which factors could be ignored. Such factors could be the graphics quality of the simulation environment or the alignment of synthetic and real height distributions.

In the following, those experimental conditions and their results will be described. For better clarity, we discuss the design of each ablation study and its result individually.

	Data set	Synthetic	Real	SyntheticToReal
E.-*D0*	Cattle	29.2	78.4	**85.8**
	SeaDronesSee	10.3	36.3	**38.8**
	VisDrone	1.2	24.6	**27.2**
F.R.	Cattle	38.8	90.5	**91.5**
	SeaDronesSee	14.6	54.7	**59.0**
	VisDrone	2.4	48.6	**51.2**
YOLO	Cattle	64.2	**88.8**	86.9
	SeaDronesSee	10.5	55.8	**60.3**
	VisDrone	10.2	43.9	**45.0**

Table 6.3: Performance of object detectors for different training strategies given by mAP@50.

6.7.1 General Benefit of Synthetic Data in UAV Object Detection

The main goal of this work is to examine the usefulness of synthetic training data to train object detectors from scratch or improve the performance of object detectors trained with real-world data.

From this goal, there naturally arise three conditions which we want to compare. First, as a baseline, we observe the performance of the object detector on the real-world data set. Second, we observe the performance of the object detector trained only on an entirely synthetically generated data set. Finally, we observe the performance of an object detector which is first pre-trained on a synthetic data set and then transfer-trained on the real-world data set. From preliminary experiments, we found these strategies to be superior to alternative strategies, such as combined training, similar to previous literature (Varol *et al.* (2017)).

For all three application scenarios, the corresponding real-world training set was split into a training, validation and test set. For fully synthetic training and synthetic pre-training the data set was split into a training and a validation set, as no testing is conducted on the synthetic data. See the sizes of the different complete sets in Table 6.4.

Table 6.3 shows that purely training on synthetic data can already provide minimal working solutions. While all object detectors perform well on the simple data set Cattle (29.2-64.2 mAP@50), the accuracies on VisDrone and SeaDronesSee are far lower. This is partly due to missing classes in the corresponding synthetic data sets ((awning-)tricycle in VisDrone and life jacket in SeaDronesSee). However, another apparent factor is the difference in the appearance of certain classes from the synthetic to the real data set. For example, see Figure 8.1 to compare the same classes in the synthetic and real data set. The default appearances of classes vary in some cases significantly. This appearance gap may be narrowed by manually editing the appearance of classes to resemble real-world objects. Despite these challenges, synthetic pre-training with subsequent transfer training

	Number of Images			
Data Set	Train	Val	Test	Classes
VisDrone	6471	548	1610	people, bike, car, truck, van motor, tricycle, awning-tricycle, bus
SeaDronesSee	2975	859	1796	swimmer, floater, boat swimmer†, floater†, LJ
Cattle	402	134	134	Cow
DGTA-VisDrone	40000	10000	–	people, bike, truck car, motor, bus, van
DGTA-Sea-DronesSee	90000	10000	–	swimmer,floater,boat swimmer†,floater†
DGTA-Cattle	40000	10000	–	Cow

Table 6.4: Number of images and classes. Note that (awning-)tricycle is abbreviated as (a.-)tri., life jacket as LF and people as ppl.

boosts performance on VisDrone and SeaDronesSee significantly across all models. On the SeaDronesSee evaluation benchmark (Kiefer *et al.* (2022)), we can even achieve state-of-the-art performance by surpassing the best model by +5.6 mAP@50. While the performance improvement is not as apparent on Cattle, pre-training on synthetic data helps for EfficientDet-*D*0 and Faster R-CNN.

These experiments show that although synthetic data sets cannot replace corresponding real-world data sets, they enhance the detection performance. Even the models from Varga *et al.* (2022) can be beaten just by synthetic pre-training. See Figure 6.3 for example predictions on VisDrone.

6.7.2 Effect of Data Set Sizes

We wanted to test the influence that the data set sizes had on the performance in this context. The intuitive hypothesis was that we would observe a curve of diminishing returns for larger data sets, which is typical in machine learning. We hypothesized that we would observe such diminishing returns for the size of the real-world data set, as well as for the size of the synthetic (pre-training) data set. Furthermore, we hypothesized that the effect of synthetic pre-training would be more emphasized when using a smaller real-world data set.

To conduct this ablation, we varied the size of the real-world data set and of the synthetic data set for VisDrone and SeaDronesSee training.

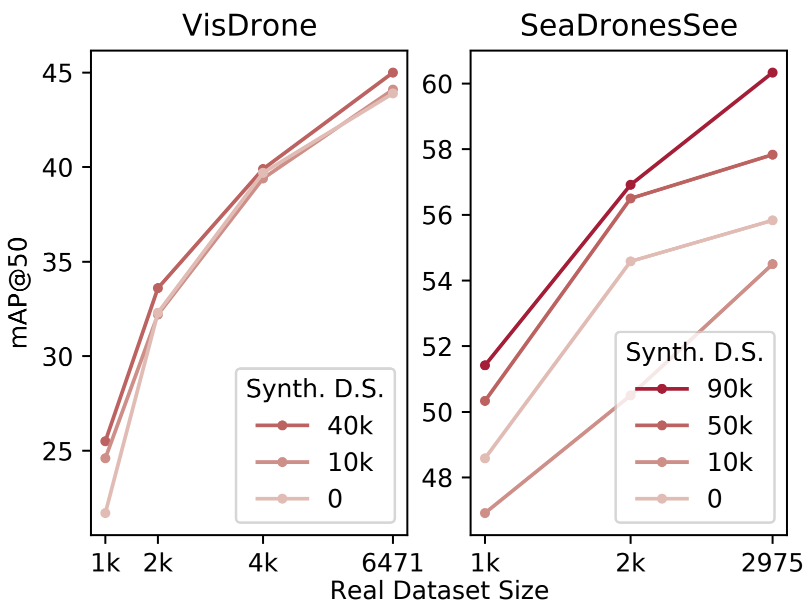


Figure 6.4: The effect of the real and synthetic data set size on the model performance for VisDrone and SeaDronesSee. The color shading specifies the size of the synthetic data set that was used for pre-training.

Figure 6.4 shows the effect of different data set sizes in synthetic pre-training. As hypothesized, we observe an improved mAP@50 with an increase of the real-world data set size, and an increase of the synthetic pre-training data set size with a diminishing return for larger data sets. Furthermore, we observe the hypothesized interactive effect, such that the improvement from using synthetic pre-training data is disproportionally larger for smaller real-world data sets.

Our found effects are consistent with the numbers of those reported in Hurl *et al.* (2019), that is, improvements of about +1.0 to +5.0 mAP@50 by using additional synthetic training data. We note that in Hurl *et al.* (2019), a 3D LiDAR object detection task was examined instead of a 2D object detection task. Compared to Hurl *et al.* (2019), in our work, those improvements are not only on one class but over the whole 10 and 6 classes of VisDrone and SeaDronesSee, respectively.

6.7.3 Effect of using pre-trained weights

The use of initial weights pre-trained on large scale image data sets like ImageNet or COCO has become a standard for many vision machine learning tasks in recent years. We hypothesized that there could be an interaction between using such pre-trained weights and synthetic (pre-)training. For example, such pre-trained weights could encode invariance to image noise, which is prominent in real world images, but may be missing

	Random initial weights	Pre-trained on COCO
Real	43.1	43.9
Synthetic	7.5	10.2
SyntheticToReal	44.7	45.0

Table 6.5: Effect of COCO pre-trained weights on performance given by mAP@50 evaluated on VisDrone.

in synthetically generated images. The use of pre-trained initial weights would thereby allow a purely synthetically trained model to obtain the information it could not obtain from the synthetic data, thereby introducing an interactive effect. If this hypothesis was true, using pre-trained weights would improve the performance of a purely synthetically trained model disproportionally more than it improves the performance of a model trained on real world data.

Table 6.5 shows the effect of using COCO pre-trained weights as initial weights for the training. Due to the small effect sizes, we would argue that our results are inconclusive at this time. We observe a more significant absolute improvement of the mAP@50 when using COCO pre-trained weights on pure DGTA-VisDrone training than on pure VisDrone training, which would speak for the hypothesized interactive effect. However, when observing the absolute obtained mAP@50 of those different conditions, this observed difference in improvements would also be consistent with an effect of diminishing returns of improvements for a larger mAP@50. This effect would be that starting from a weaker performance baseline, the same improvement (using COCO pre-trained weights) would yield a larger improvement than when starting from a more robust baseline.

So we conclude that there is at least no strong effect where using pre-trained weights trained on real-world imagery would disproportionally improve the performance of synthetic training. However, note that there might be other factors to consider, such as training time and stability.

6.7.4 Effect of Good/Bad graphics settings

One of the parameters that one would intuitively look at when observing synthetic training data or when trying to improve the use of synthetic training data is the realism of this training data. This could be framed as closing the Sim-To-Real gap as discussed above.

A similar but not fully congruent perspective on this coming from game development is trying to improve the realism of games by improving their graphics quality. In many cases, those improvements of graphics quality come with high computational demands (higher polygon models, higher resolution textures, demanding physics simulations like particle effects and lighting) and high amounts of labor, e.g. from graphics artists.

Therefore, from an engineering perspective, this raises the question, how well spent

	Real	Synthetic	SyntheticToReal
Real data Baseline	43.9	-	-
Low Quality	-	8.4	45.1
High Quality	-	10.2	45.0

Table 6.6: Performance between low quality and high quality in-game settings (given by mAP@50). We compare the performance for only Real training on VisDrone against the performance with synthetic data as DGTA-VisDrone.

this effort is to obtain the final goal of increasing the synthetically trained model's performance.

To answer this question, we conducted experiments comparing the effects of synthetic training data obtained from GTAV either set to the highest or lowest possible graphics settings.

Table 6.6 shows the obtained model performance using synthetic data obtained from GTAV on the lowest and highest graphics settings, respectively. We observe that for purely synthetic training, the model trained with higher graphics setting images outperforms the one trained on lower graphics quality images. This effect, however, is small.

There is no observable effect in the synthetic pre-training condition with transfer training on the real-world data set.

6.7.5 Aligning Domain Distributions

Modifying the DeepGTAV tools to extract metadata allows us to automate parts of the data generation process by aligning the metadata distributions of the real and synthetic data sets. Instead of laboriously setting the correct metadata settings in the synthetic data generation process, one could fall back to the corresponding real data set to adapt the parameters automatically. Aligning the distributions may result in higher synthetic data quality or efficiency.

For instance, in SeaDronesSee, every image is annotated with the capture time stamp. By bootstrap sampling from the time distribution, we sample a new data set of 100k synthetic images with the correct time distribution. As before, we train a Yolov5 model and test it on the SeaDronesSee test set (Synthetic Only), and we transfer-train it on SeaDronesSee (Synthetic-To-Real).

Table 6.7 shows that aligning the time helps in synthetic only training and synthetic pre-training by increasing the performance over the unaligned baseline by +4.1 and +0.2 mAP@50, respectively. The performance increase is mainly due to SeaDronesSee only featuring day-time images, such that sampling synthetic night images deteriorates the performance.

Similarly, we can also automatically adjust the camera angle. The images in the Cattle data set have been taken from a downward-facing camera where a gimbal corrects for

Table 6.7: Effect of time and angle alignment on performance (given as mAP@50).

SeaDronesSee	Synthetic	SyntheticToReal
Only real training	-	55.8
Unaligned Synthetic Pre-training	10.5	60.3
Time Aligned	**14.6**	**60.5**
Cattle		
Only real training	-	78.4
Unaligned Synthetic Pre-training	29.2	**85.8**
Angle Aligned (Subset)	**36.6**	**85.8**

UAV angular movement. We sample from DGTA-Cattle only these images that fall into the range of these angles (with an error threshold of at most 20 degrees). This reduces the original 40k images to only approx. 10% of the images (3,954). We train an EfficientDet-*D*0 on this subset.

Interestingly, the performance of the synthetic only training improves over the more extensive unaligned DGTA-Cattle training. This is likely due to the limited capacity of an EfficientDet-*D*0 model, resulting in the model distributing its performance across many other angular viewpoints, which are unnecessary for the performance in this use-case. This experiment illustrates that less but more targeted data may be sufficient to reach the same performance while reducing the need to filter the data by only aligning the metadata distributions manually.

6.8 Limitations and Conclusions

We demonstrated that synthetic data can be leveraged for object detection on UAVs. We can improve the performance over only real training by synthetic pre-training on multiple application scenarios. Synthetic-only training yields satisfactory results, but performances are not yet competitive to real training.

From our ablations, we conclude that the use of more synthetic training data improves the performance. The use of weights pre-trained on large scale image data sets constantly improves the performance, although we find no interactive effect with synthetic training. The graphics quality of the simulation engine appears to be important for purely synthetic training but not for synthetic pre-training. In general, metadata alignment is vital for the usefulness and data efficiency of synthetic training data.

We hope that the adaptation of the DeepGTAV tools helps cast light on object detection on UAVs via synthetically generated footage. In future works, the capabilities of the DeepGTAV framework to produce object segmentation data, LiDAR data and video could be leveraged.

Chapter 7

Tackling Weakly Supervised Data

Autonomous vision aboard UAVs has grown to an important research area (Zhu *et al.* (2018a); Menouar *et al.* (2017); Lygouras *et al.* (2019); Mishra *et al.* (2020)). Next to traffic surveillance (Fan *et al.* (2020b); Du *et al.* (2018)) and agriculture (Tsouros *et al.* (2019)), also the field of search and rescue (SaR) has been tackled (Varga *et al.* (2021); Mishra *et al.* (2020)). However, while several works focus on path-planning and mission implementation (Bevacqua *et al.* (2015); Hayat *et al.* (2020); Mayer *et al.* (2019)), few works address the actual vision part, necessary for autonomously searching certain areas.

Finding interesting regions on the sea is a hard problem, since objects of interest are often not known a priori or have a vast variety of different appearances which is why supervised methods often fail in these scenarios. Even if object categories are known beforehand, current methods focus on object detection, which is not viable for large image resolutions and real-time (for rigor defined here to be $>$25FPS) performance on embedded hardware. Both constraints occur in reliable SaR missions.

Furthermore, labeled data sets in these environments are scarce as the data acquisition is complicated, requiring strict safety regulations for all subjects, and is expensive (Kiefer (2022)). Instead, it is considerably easier and cheaper to obtain raw data of sea surfaces.

What is more, often a low bandwidth, possibly due to large distances or suboptimal weather conditions, does not allow for the whole footage being transmitted to a ground station. This becomes especially severe in maritime scenarios, where the drone is far away from any ground station (Avalon (2022); Dortmund (2022)). While compression can be done on-board, it is often not sufficient and furthermore results in image quality loss across the whole image, i.e. also possibly quality loss in regions of the image that need to be analyzed more thoroughly to exclude false positives or negatives.

Recently, special purpose video codecs that allow few regions of an image to be coded with near constant picture quality have been proposed to tackle this problem (Steinert and Stabernack (2022)). The high quality regions that are transmitted can subsequently be combined with an actual classical object detection system on a ground station with much more hardware resources.

This methodology separates the problem into two stages: generating few high-recall regions of interest of a high dimensional image in real-time in a low resource environment and classifying these regions into known classes on a ground station with more

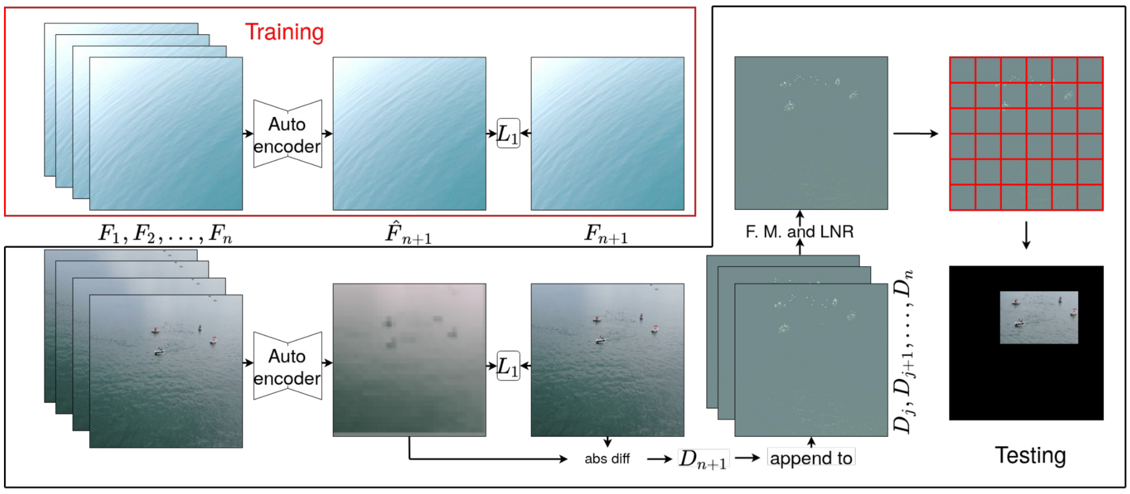


Figure 7.1: Future frame prediction autoencoder pipeline. The frames $F_1, \ldots, F_n$ are concatenated and input into the autoencoder, which learns to predict F_{n+1} via $\hat{F}_{n+1}$ and the L_1 loss. The error frame (absolute difference between the two), D_{n+1}, is concatenated to the last $D_j, \ldots, D_n$. Then ,frame momentum and local noise reducer are applied until a final grid is put on the resulting error frame to yield final regions of interests.

resources. Motivated by these observations, in this work, we formulate and formalize the former problem and propose an autoencoder-based future frame prediction model that generates meaningful regions of interest on sea surfaces which can run in real-time on an embedded GPU. Owing to the nature of maritime environments, we show that classical methods perform poorly due to dynamic backgrounds, wave movements, sun reflections and others while modern methods are too slow. As this method is a type of anomaly detection method, it does not require bounding box annotations. We introduce a metric that measures the recall at a given amount of footage being transmitted and show that this method outperforms classical methods on multiple benchmarks and metrics.

To train the proposed model, we capture over 60 minutes of 4K video footage with several cameras depicting the sea surface from different angles and altitudes at different days and waters. We also capture video footage of objects and manually label it.

In this chapter, we formulate a novel problem of obtaining high-recall regions of interest in a high-resolution and real-time scenario and propose a future frame prediction autoencoder to detect these regions in real-time on an embedded GPU. To this end, we describe a new benchmark, consisting of 60 minutes of video footage of the sea surface in various conditions as training set for our method. This so-called *Maritime Anomaly Detection Benchmark* is hosted on the web server. Finally, we analyze the proposed method and compare it to traditional and modern methods on two large-scale public data sets.

7.1 Related Work: Maritime Computer Vision

Airborne maritime data sets are scarce and mostly focus on synthetic aperture radar satellite imagery and ships (Airbus (2022); Chen *et al.* (2020); Wang *et al.* (2019b); Zhang *et al.* (2021b)). Marques *et al.* (2015); Varga *et al.* (2021); Lygouras *et al.* (2019) provide UAV-based maritime detection data sets. While the data set in Lygouras *et al.* (2019) features only stock photos scraped from the internet, the Seagull data set (Marques *et al.* (2015)) and SeaDronesSee (Varga *et al.* (2021)) provide video material with objects of interest. Of these data sets, only Seagull provides frames that do not contain objects. However, the videos suffer from heavy lens distortion and distortion caused by a rolling shutter. We collect 60 minutes of video footage ($>$100000 frames) depicting the sea surface in various altitudes at different angles and days with multiple cameras. We weakly annotate the footage such that no objects of interest are visible in any of the frames.

7.2 Related Work: UAV-based Detection

Rudimentary vision methods in SaR scenarios aboard a UAV are done in Scherer *et al.* (2015), using color, text or shape cues using OpenCV (Bradski (2000)) to detect objects of interest. Similarly, Rudol and Doherty (2008) use Haar features to detect objects of interest in SaR missions. Among the learning-based methods, Ferreira and Silveira (2020) consider the application of ship detection, classifying images into positives (containing ships) and negatives. While it is also an unsupervised method, they ignore the localization. Lygouras *et al.* (2019) describe a complete SaR system from path planning over detection to action. However, their detection system is a basic YOLO variant unsuitable for large resolutions and real-time. Furthermore, it is restricted to the objects it is trained on. Generally, all literature regarding supervised UAV object detection can be considered related (Fan *et al.* (2020b); Zhu *et al.* (2018a); Du *et al.* (2018); Varga *et al.* (2021); Xia *et al.* (2018); Kiefer *et al.* (2021); Price *et al.* (2018)), albeit not viable, since they do not work in real-time large-resolution scenarios on embedded hardware.

7.3 Related Work: Region proposal networks

Selective search (Uijlings *et al.* (2013); Van de Sande *et al.* (2011)) generates many thousand little informative boxes for use of an object detector at a later stage, which makes it inapplicable in embedded environments. Common region proposal networks (Zhong *et al.* (2019)) are used in two-stage object detectors, such as Faster R-CNN (Ren *et al.* (2015)), but require bounding box supervision to be learned. Methods for weakly supervised object detection often employ region proposal networks (Tang *et al.* (2018)), but require image-level annotations.

Background subtraction methods (Benezeth *et al.* (2010)) are used to separate the background from the foreground, which is defined by the scene captured by a static camera. Most of the methods are not suitable for dynamically changing scenes caused by camera and background movement. Furthermore, these methods do not focus on obtaining meaningful bounding box locations but only on the obtained segmentations.

(Video) Anomaly detection methods (Deecke *et al.* (2018); Sultani *et al.* (2018); Nguyen and Meunier (2019)) learn on normal samples and detect anything previously not seen as anomalies. In images, this is often done for industrial parts (Staar *et al.* (2019); Roth *et al.* (2021)), and in static videos for surveillance in traffic and crowded scenes (Saligrama and Chen (2012); Zhao *et al.* (2017); Zhou *et al.* (2019a); Liu *et al.* (2018)). Earlier methods only focus on classifying images or frames (Ferreira and Silveira (2020); Liu *et al.* (2018)), while newer methods also consider localizing anomalies (Li *et al.* (2021); Szymanowicz *et al.* (2022a)). However, these methods either ignore the temporal dimension or are not suitable for real-time use. Furthermore, video anomaly methods are designed for static scenes. Our work focuses on dynamic scenes and requires models running in real-time on embedded hardware. Furthermore, the focus is on generating high recall meaningful bounding box regions that potentially contain objects of interest.

7.4 Autoencoder for Video Anomaly Detection

We are given a high resolution (e.g. 4K) video stream depicting the sea surface. Furthermore, we have an embedded GPU (e.g. Nvidia Xavier AGX). The task is to select regions of interest in every frame which are to be transmitted down (e.g. via a streaming FPGA (Steinert and Stabernack (2022))). Each region is defined via four bounding box locations, describing the corners of the region in pixels (similar to classical object detection). Depending on the exact use case, the remaining regions are either also transmitted with lower quality or completely omitted.

We propose an autoencoder-based future frame prediction architecture to detect anomalies (See Fig. 7.1). We train a shallow autoencoder on sequences of normal images $F_1, \ldots, F_n$ depicting the sea surface such that the model learns to predict the next normal frame. Subsequently, the predicted frame $\hat{F}_{n+1}$ is subtracted from the original next frame F_{n+1}. The hypothesis is that the autoencoder fails to reconstruct objects that differ from the sea surface in their colors, shapes and textures. Furthermore, by incorporating the last few frames, the autoencoder learns temporal correlations of water movements.

Common future frame prediction networks employ large networks, such as UNet (Liu *et al.* (2018); Szymanowicz *et al.* (2022b)) or even larger models (Yu *et al.* (2020)). They operate on small video resolutions and are not suitable for employment on embedded hardware. Applying models in real-time on embedded hardware and for high resolutions requires us to fall back to shallow autoencoder architectures. We follow the basic principle of an encoder-decoder architecture, but only employ small channel dimensions for the filters as these make up for a large computational overhead. We refrain from using depth-wise separable convolutions (Howard *et al.* (2017)) or more advanced

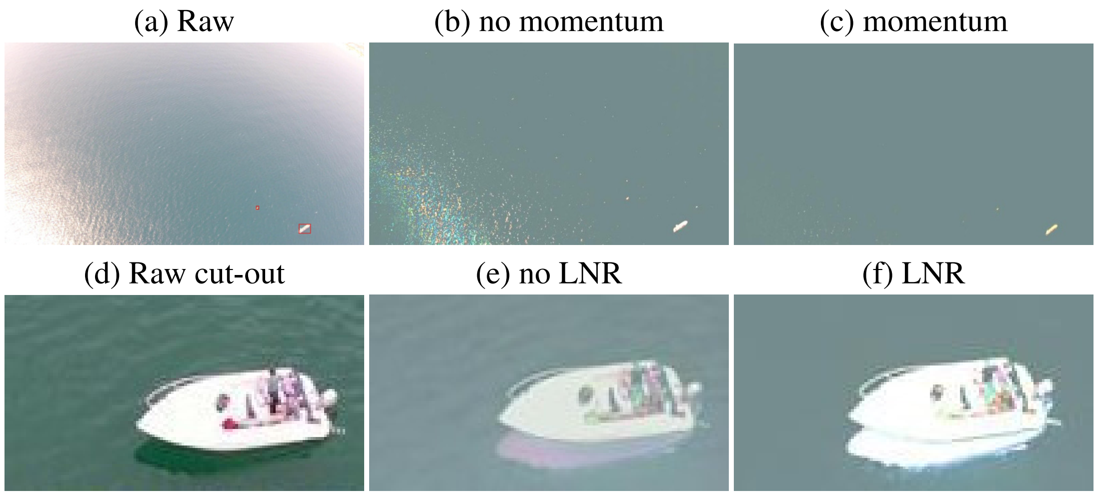


Figure 7.2: Qualitative errors on Seagull (top) and SeaDronesSee (bottom). Note that the random noise at the bottom left of (c) almost vanished.

methods (Lebedev *et al.* (2014)), since they are not optimized for embedded GPUs. For the first layer, we concatenate the past n frames along the channel dimension and apply a regular 2D convolution with filter dimension $n \times 4$, kernel size 3×3 and stride 2. We perform the same convolution six times, while halving the channel dimension each time due to performance. The decoder performs the symmetric operations via deconvolutions.

We hypothesize that reconstruction errors due to wave patterns and sun reflections are more temporally unstable than actual anomalies. Thus, we propose to include an error frame momentum term, which averages over the past n error frames $D_1, ..., D_n$. This assumes that the camera movement is not too quick as then, the actual anomalies also move quickly in the image plane, eliminating the error frame momentum effect. However, for frame rates of roughly 30, this is negligible. Figure 7.2 (c) and Section 7.6 show the advantage of using this component.

To counteract the local noise induced by an imperfect reconstruction coming from sun reflections and wave patterns, we introduce a local noise remover (LNR). Channel-wise, we multiply each pixel of the error frame by its immediate vertical and horizontal surrounding neighbour. We repeat this procedure three times. This ensures that only regions of larger error areas are detected as anomalies (as opposed to noisy areas) in subsequent steps. This can be seen as a morphological operation, however also different, since we do not use a structuring element (Zhuang and Haralick (1986)). See how the waves are eliminated in Fig. 7.2 (f), while the boat is amplified.

Importantly, this method is sensitive to regions above the horizon. Therefore, we leverage metadata from the UAV's on-board sensors to determine the horizon line in open water. This allows us to ignore this region in the autoencoder training and inference phase which, in turn, results in more robust anomaly detection performance and faster

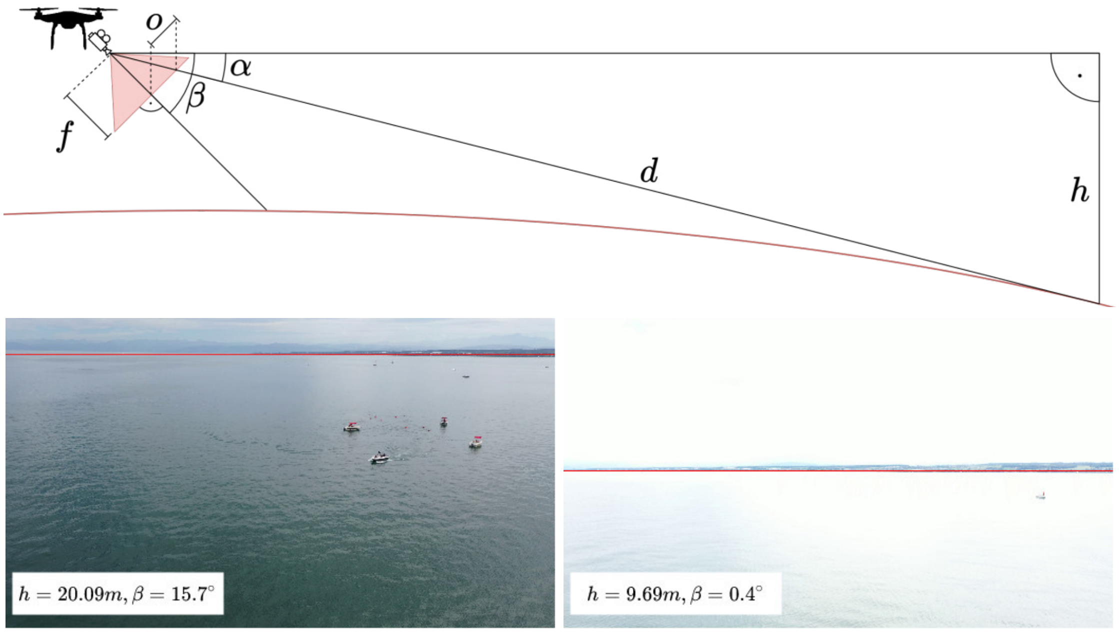

o
α
β
f
d
h
$h = 20.09m, \beta = 15.7^\circ$
$h = 9.69m, \beta = 0.4^\circ$

Figure 7.3: Illustration of the horizon cutter (top) and predictions (bottom). The curvature is just for visualization purpose to show the tangentiality (ignored for computation).

inference times. Notably, this computation has virtually no overhead. The horizon line can be computed using the UAV's height, camera gimbal pitch and roll angle, and the camera intrinsics. Ignoring the effect of atmospheric refraction, we can estimate the distance to the horizon as a function of the height of the observer as $d = 3.57h^{1/2}$. This approximation is fairly accurate for heights that are typical for SaR-UAVs (far below 1000m) (Bohren and Fraser (1986)). We furthermore ignore the curvature of the earth, which is also negligible for these heights. We compute the angle α to the horizon via $\alpha = \arcsin(h/d)$. Using the focal length f (in pixels) and the camera gimbal pitch β, we can then compute the camera perspective projection on the image plane, which yields the height offset o in pixels to the horizontal center line of the image plane as $o = \tan(|\alpha - \beta|) \cdot f \cdot \text{sgn}(\alpha - \beta)$. Naturally, we truncate o to be within the range of the number of horizontal pixels. To account for the roll angle γ of the UAV (or camera gimbal), we can simply add a roll angle induced offset at the left and subtract at the right of the image given as $o_r = \tan(\gamma) \cdot p_w/2$, where p_w is the pixel width of the video. While the horizontal pixel location o is an approximation, it is quite robust to errors in the altitude h. Since an exact error analysis is not in the scope of this work, we just report values for altitudes that are common in the data set SeaDronesSee. For $\beta = 0 - 20^\circ, h < 300m$ it holds that $10m$ in altitude error results in approx. $1px$ offset change in a 4K image. However, o is very sensitive to errors in the gimbal angle β . For example, for $h = 130m, \beta = 16^\circ$, 1° in angle error results in approx. $40px$ offset change. Therefore, it is essential to have a well-calibrated gimbal and UAV IMU. The latter can be accurate up to 0.1° when configured

Errors of	γ	o	$o/2160$
horizon visible (5%)	0.8°	71px	3.3%
horizon not visible (95%)	–	2px	0.1%
total	–	5.45px	0.3%

Table 7.1: Error to the ground truth horizon as measured by roll angle γ and pixel offset o in a 4K image.

properly (Suzuki *et al.* (2016)). See Figure 7.3 for an illustration. Empirically, we show that the horizon cutter performs well despite the occurrence of nearby land. We manually annotate the horizon line for a subset of the SeaDronesSee-Tracking validation set and compute the pixel offset error and the roll angle γ error. Table 7.1 shows that despite some land mass blocking the horizon (also see Fig. 7.3), the error is negligible.

Lastly, we apply a grid of size $m_1 \times m_2$ on the error frame and for every grid window we average over the error frame pixels contained in it. We select the maximum number of boxes outputted given a certain bandwidth, which we simply break down in $p\%$ of the area of the whole frame.

7.5 Data Set Generation and Webserver

To test our approach, we gathered 60 minutes of 4K video footage on open water at three different days with three different cameras. We made sure to include altitudes and viewing angles from $5-120m$ and $0-90°$. We manually filtered out the sequences that contained objects considered anomalous, such as humans, boats, life jackets and buoys. Each frame is annotated with its corresponding metadata information, such as altitude, all angles of the UAV principal axes, camera gimbal pitch angle, time, GPS and others.

This data, called **OpenWater**, comes along with over 20 minutes of bounding box annotated footage in open water where we annotated the same classes as in SeaDronesSee, serving as anomalies. Everything but the test annotations will be uploaded to avoid researchers from overfitting. Researchers may upload their predictions to the web server, which will be evaluated and published on the server side for fair comparisons.

For our experiments, we choose a grid size of 48×27, predict the fifth frame from the past four, and use an error frame momentum of two. Influences of the components are discussed in further sections.

As we operate on high resolution videos and in real-time scenarios, we compare to three methods commonly used for background subtraction and anomaly detection: Mean filter (MF) (Zhang and Ding (2012)), frame differencing (FD) (Mohamed *et al.* (2010)) and Gaussian mixture model (GMM) (Zivkovic (2004)). For GMM we use three Gaussians. We extend every method with the grid component and employ the horizon cutter.

We evaluate on the following datasets (see also Fig. 7.4):

- We use our **Open Water** data set as the training set for SeaDronesSee as the latter does not consist of frames without objects. It consists of > 100000 frames of open water without any objects,captured on multiple days with three different 4K video cameras. Analogously to SeaDronesSee, it comes with precise meta annotations for all frames.
- We test on SeaDronesSee-MOT. It depicts humans, boats and other objects in open water (incl. bboxes) serving as our anomalies.
- The **Seagull** data set features video data showing boats, ships, life rafts and other objects from a fixed wing UAV. It also features video clips containing no objects. The latter serve as our training set for the Seagull test set. The videos are of Full HD resolution and have a heavy lens distortion and distortion caused by a rolling shutter. See Figure 7.4 for examples.

We measure the recall given a certain bandwidth (percentage of the video frame transmitted averaged over all frames). Therefore, we consider as evaluation metric $R^{p\%}$, which is the recall over all frames, given that at most $p\%$ of the image may be transmitted. Each region to be transmitted must be encoded by a rectangular bounding box. We consider an object to be correctly detected if there is an overlap with the predicted region of at least 50%, which is common for aerial object detection (Zhu *et al.* (2018b); Varga *et al.* (2021); Du *et al.* (2018)). Furthermore, we report the average recall (AR) averaged over 10 equidistant percentages p from $p = 0.05$ to $p = 0.95$, denoted AR.

7.5.1 Anomaly Detection Performance

7.6 Experiments

Fig. 7.5 shows AR and the recall values for all the methods for the 10 transmission percentages p, while we interpolate in between. The autoencoder consistently outperforms the baselines for all values of p and for AR. However, the difference is especially visible for low values of p, which is the primary use case in this application scenario (Steinert and Stabernack (2022)). For example, for $p = 5\%$ the autoencoder achieves 70.1% and 40.0% recall for SeaDronesSee and Seagull, respectively, which is over 15 resp. 32 percent points more than the best baseline. Subsequently, we focus on the case of low p.

We report the average reconstruction errors err_b (average L_1 reconstruction error within ground truth boxes), err_r (average L_1 reconstruction error rest) and their differences Δ_r in Table 7.2. The autoencoder yields higher Δ_r, which shows its ability to discriminate better between normal and anomalous regions. Notably, the values for SeaDronesSee are generally much higher than for Seagull due to Seagull's lower image quality and higher blurriness (see Figure 7.4).

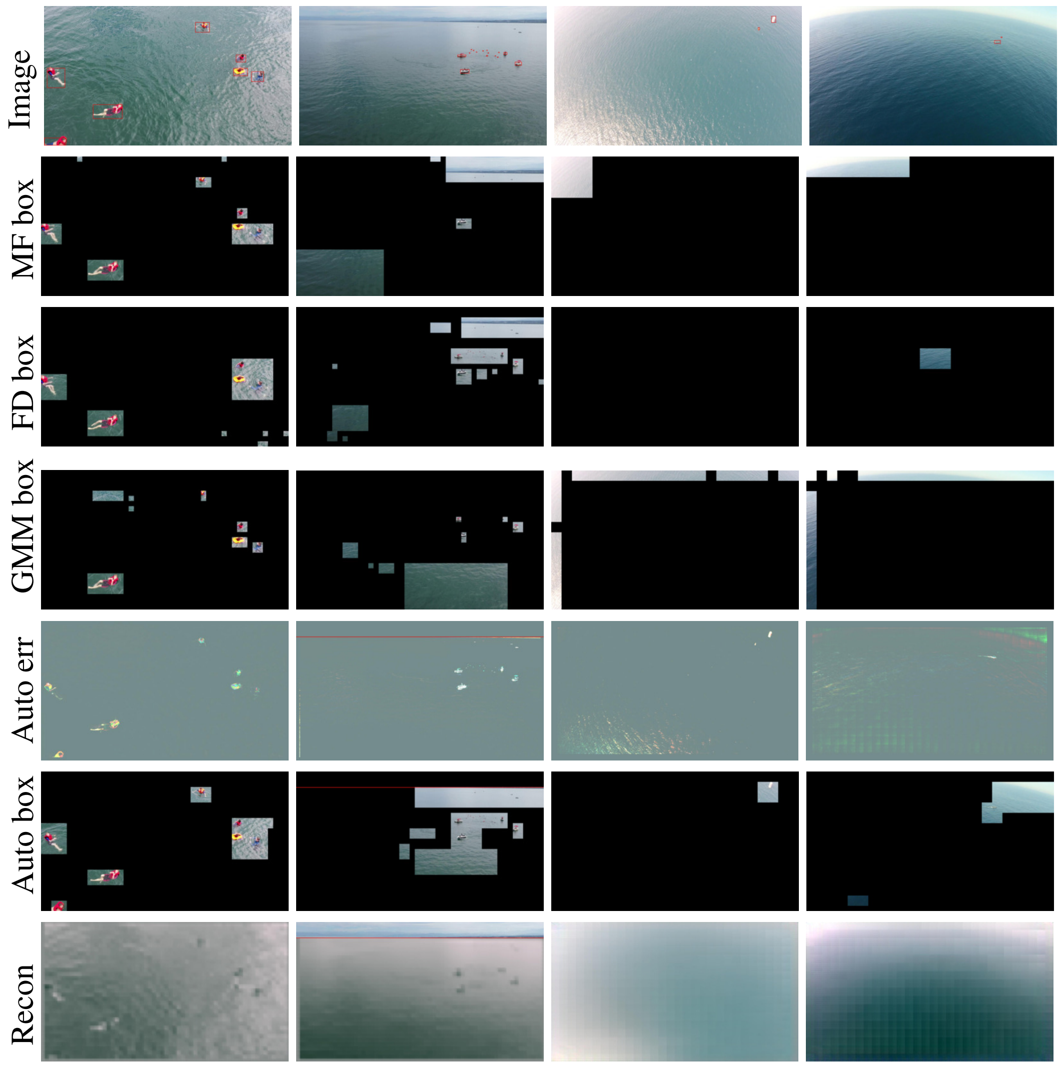


Figure 7.4: Qualitative results for mean filter (MF), frame differencing (FD), Gaussian mixture model (GMM) and the autoencoder (Auto) on SeaDronesSee (left two columns) and Seagull (right two columns). For Auto, we plot the error heat map and the reconstructed image (Recon).

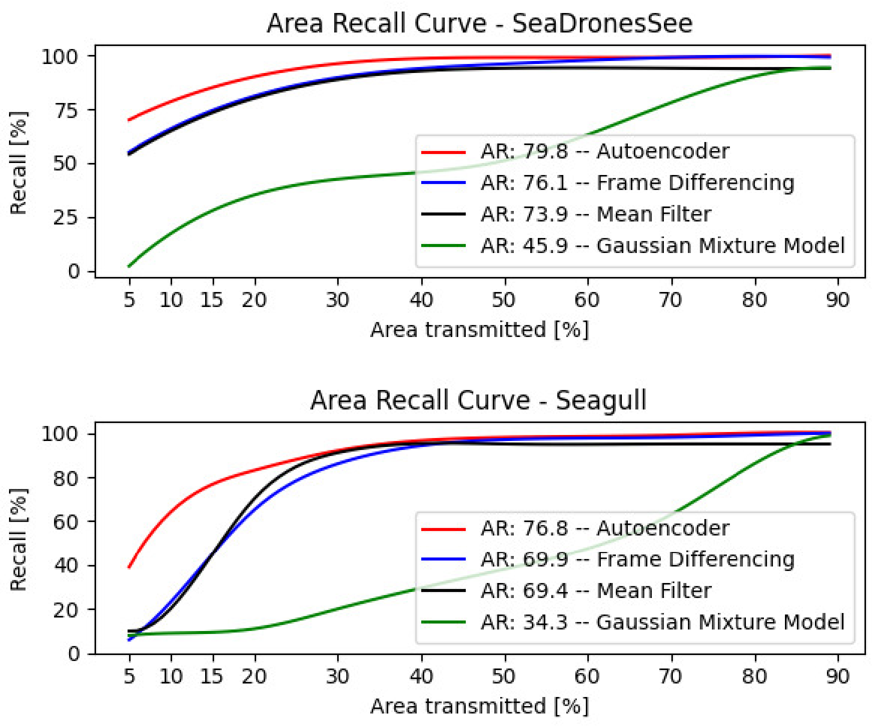


Figure 7.5: Area recall curves for SeaDronesSee and Seagull.

	SeaDronesSee			Seagull		
	err_b	err_r	Δ_r	err_b	err_r	Δ_r
MF	78.3	0.3	78.0	0.37	0.3	0.07
FD	34.7	2.5	32.2	1.7	0.3	1.4
GMM	4.3	0.2	4.1	0.3	0.2	0.1
Auto	79.5	0.2	**79.3**	2.2	0.2	**2.0**

Table 7.2: Average L_1 recon. error within boxes and outside for $p = 5\%$.

Future Frames	–	✓	✓	✓
Local Noise Remover	–	–	✓	✓
Frame Momentum	–	–	–	✓
$R^{p=5\%}$	60.3	66.2	68.6	**70.1**

Table 7.3: Autoencoder ablation experiment on SeaDronesSee.

Table 7.3 analyzes the influence of different components. When using future frames, we take the past four frames to predict the fifth. For frame momentum, we use the past two frames. It shows that using future frames yields the greatest benefits. All components improve the performance.

We analyze the influence of the horizon cutter on the performance of the autoencoder. As only SeaDronesSee incorporates metadata, we perform experiments on this data set. Only 5% of all frames actually show the horizon. Therefore, we restrict the influence of the horizon cutter to only that portion, as it does not have any on the other part. Remarkably, the autoencoder with horizon cutter achieves 86.3% recall whereas it only achieves 47.0% without. As the autoencoder only is trained on frames of open water, the different image statistics of the sky skew the image reconstruction error on these parts which expectedly results in a loss in performance. Both experiments used $p = 5\%$.

So far, we considered the case where we only have access to normal frames as training data. However, often we are given some labeled training data. Thus, we propose to use an adversarial training objective where we maximize the prediction penalty of the autoencoder within ground truth boxes and minimize it everywhere else. That way, the model is punished for learning to reconstruct actual anomalies. We evaluate this strategy by comparing it to its naive counterpart, i.e. not backpropagating the loss within boxes.

We compare these two approaches by training on the SeaDronesSee tracking train set and testing on the SeaDronesSee tracking test set. For $p = 5\%$,the ignoring yields a recall of 65.7% in contrast to 67.2% with adversarial loss.

7.6.1 Obtaining Fewer Bounding Boxes

Aside from the restriction of choosing at most $p\%$ of the frames, which may be imposed due to a potentially low bandwidth, another restriction may come from common video codecs' inability to process a large number of regions of interest. Therefore, another type of restriction on a region proposer may be the number of regions it yields.

Thus, we propose to merge regions of interest touching each other at corners using Suzuki's border following method (Suzuki *et al.* (1985)). As this may yield a larger than allowed area to be transmitted, each resulting box is ranked based on its reconstruction error. This leads to fewer and larger bounding boxes at the expense of a lower recall.

Table 7.4 shows the number of boxes and the recall for the standard and the merging method for $p = 5\%$. Note that without merging we have the same number of bounding boxes for all the methods since we allow 5% of the area of the image to be transmitted. We can substantially decrease the number of bounding boxes at the cost of a slightly lower recall. We note that this also highly depends on the anomaly distribution since for clustered anomalies it is easier to merge bounding boxes (see Fig. 7.4).

	Not Merging		Merging	
Method	$\#B$	$R^{\text{p}=5\%}$	$\#B$	$R^{\text{p}=5\%}$
MF Zhang and Ding (2012)	**65**	54.8	7	49.6
FD Mohamed *et al.* (2010)	**65**	55.0	8	53.2
GMM Zivkovic (2004)	**65**	0.2	**5**	0.1
Auto	**65**	**70.1**	6	**64.3**

Table 7.4: Fewer bounding boxes via reduced recall on SeaDronesSee.

	MF	FD	GMM	U-NET	CFLOW	Auto
1K	**64**	**70**	**35**	8	12	**48**
4K	**50**	**62**	17	1	3	**27**

Table 7.5: Running times in FPS. Bold values depict real-time methods.

7.6.2 Running Times

Finally, we consider the running times of the individual methods on embedded hardware. We deploy them on an NVIDIA Xavier (Nvidia (2022b)) mounted on a DJI Matrice 100. We transform all methods into optimized engines using TensorRT (Vanholder (2016)) and set the Xavier to MAX-N mode and report the running times averaged over 1000 frames. Table 7.5 shows the speed comparison between traditional and modern (U-NET (Liu *et al.* (2018)), CFLOW (Gudovskiy *et al.* (2022))) methods. The much simpler baselines run in real-time, while the modern methods are slow.

For completeness, we replaced our architecture with the popular UNet architecture and trained it on SeaDronesSee using halved resolution and filter dimensions (more did not fit into a 3090Ti w/ 24GB). Interestingly, the performance trailed the performance of our method (78.1 AR). This led us to the conjecture that the high resolution is crucial in this application, which makes sense if we consider that many objects are of $\approx$ 20px size.

7.7 Conclusion and Outlook

We formulated the novel problem in maritime SaR of finding relevant regions of interest in a low-resource real-time and high-resolution scenario. We show that an autoencoder-based future frame prediction model is a promising direction even in a resource constrained setting. We make the benchmark publicly available and hope that the field of maritime SaR will be advanced by means of fast neural networks in the future.

Chapter 8

Building Memory Maps

When we make predictions about the presence and location of objects, we have an internal understanding of our surrounding world: implicitly, we know where we are in relation to the object and we know about the topology of a given scene. This internal understanding of the surrounding geometry allows us to reason robustly about the existence and location of objects. Furthermore, while we make detection errors when shown ambiguous static scenes, over time, we are able to strengthen our belief about our predictions. This is due to slight changes in appearance caused by different view points, slight movement of objects or just by integrating our predictions over a certain period of time.

We argue that this awareness of our surrounding is particularly important in aerial scenarios, where we need to reason about our environment in the presence of many uncertainties, caused by the smallness of objects. Motivated and inspired by this human-based analogy, in this work, we aim to improve several computer vision tasks for object detection and tracking on UAVs.

Conventional techniques for object detection and tracking from UAV perspectives ignore the intrinsic geometry and topology present in UAV-generated imagery, frequently relying on off-the-shelf methods designed for COCO-like scenarios or with slight modifications, see e.g. Mittal *et al.* (2020). This makes it difficult for these methods to aggregate uncertain predictions over time, since movements in image space cannot correctly be tracked and, hence, features are hard to be associated temporally.

We argue that this shortcoming can easily be mitigated by leveraging freely available sensors onboard the UAV. GPS alongside compass and IMU measurements allow us to reason about our and the objects' locations in 3D coordinates. In turn, this allows us to create a *temporal memory map* of previous predictions, so that we can aggregate information over time in a geometrically sensible way, resulting in more robust predictions.

In this chapter, we show how correctly considering the 3D geometry allows us to propose a temporal memory map that results in more robust predictions. In particular, we derive mathematical formulas detailing the 3D geometry around a UAV. Then, we propose a memory map to robustify several temporal computer vision tasks. Lastly, We show in multiple experiments on diverse benchmarks the utility of our method.

8.1 Related Work

Although video object detection (VOD) on UAVs has become more relevant, with many applications emerging and many network architectures becoming fast enough to deploy on embedded devices, there is still no consensus on which VOD approach seems to be the most promising. Broadly speaking, there are optical flow-based networks, memory networks and tracking-based networks (Wu *et al.* (2021)). While memory networks, such as STDnet-ST (Bosquet *et al.* (2021)), achieve high accuracies, their heavy architectures prohibit their deployment on embedded devices. Optical flow-based networks are faster, but their benefit over single-frame object detectors is limited (Corsel *et al.* (2023)). Most tracking-based networks first do single-frame object detection and use the association for the detection reciprocally (Luo *et al.* (2019)). In fact, the best two VOD models of the last VisDrone-VID challenge were single-frame methods, entirely ignoring the temporal domain (Zhu *et al.* (2019)).

Furthermore, all common video object detectors on UAVs ignore the underlying 3D geometry of the scene and only operate in image space (Wu *et al.* (2021)). While there is much research focusing on geolocation, it is only focused on obtaining world coordinates of objects for downstream tasks, such as following a target (Zhao *et al.* (2019a)). Similarly, occupancy networks and other mapping approaches aim at obtaining a map for mapping or scene understanding (Wei *et al.* (2022)).

As opposed to these works, we aim to leverage metadata to build a memory map in GPS space that *improves the video object detection performance*. Since we only rely on freely available metadata onboard the UAV, this approach is viable for small UAVs with a standard RGB camera in low-cost settings (compare to active geolocation via laser (Yang *et al.* (2019b))).

8.2 Deriving Formulas for 3D Geometry

Figure 8.1 illustrates that we would like to know the GPS position of the swimmer as indicated by the orange cross. How do we obtain that? First, we discuss how to obtain relative coordinates to the UAV, then how to use these to obtain actual world coordinates via passive geolocation.

8.2.1 Relative Coordinates

We consider a mathematical perspective projection camera model since this resembles the common use-case for cameras on UAVs. We consider our camera to have a focal length f given in pixels. For simplicity, we assume our image to be of 4K (3840x2160) resolution. Other resolutions follow an analogous derivation. Our camera is looking down at an angle of β, which is the variable gimbal angle. The gimbal also balances a potential UAV roll angle, so that we assume there to be a zero camera roll angle.

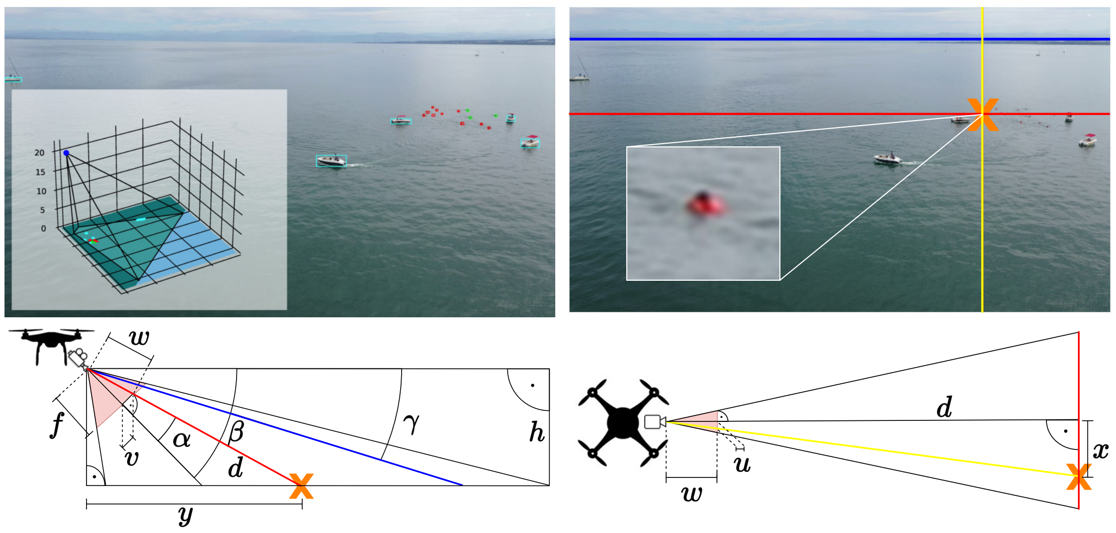


Figure 8.1: **Top left**: Illustration of predictions and the corresponding 3D geometry/-topology. We aggregate information over time, taking the correct 3D geometry into account. **Top right**: Here, we have that $h = 20m$ and $\beta = 15°$ resulting in $y = 110m$ and $x = 25m$. This means, that the swimmer is $110m$ in front and $25m$ to the right of the UAV as measured on the ground. Note that we also estimate the horizon line as indicated by the blue line. **Bottom**: Illustration of variables and formulas to estimate the vertical distance y (red lines) and the horizontal distance x (yellow lines, illustrated on bottom left) from the UAV to the swimmer (orange cross).

First, we estimate the on-ground distance from the UAV to the red line, which we denote as y (see Figure 8.1). We assume the ground to be flat, i.e. we ignore elevation change and the curvature of the earth. The latter is a reasonable assumption for heights that are typical for UAV missions (Bohren and Fraser (1986)), while the first may make a difference in very dynamic terrains.

Using the focal length f and the y-axis pixel position offset v from the horizontal center pixel line of the image, we compute y by computing the angle α via

$$\alpha = \arctan\left(\frac{v}{f}\right). \tag{8.1}$$

This allows us to compute y via

$$y = \tan(90° - (\beta - \alpha))h. \tag{8.2}$$

Having the vertical ground-distance to the object of interest, we compute the horizontal ground-distance x by looking from the top as indicated on the bottom left of Figure 8.1. For that, we need the variables d and w, which both are easily computed via Pythagoras' theorem, which then yield x:

$$d = \sqrt{h^2 + y^2},\ w = \sqrt{f^2 + v^2},\ x = \frac{u}{w} d. \tag{8.3}$$

For simplicity, we ignored the sign of x, which indicated whether we are on the left or on the right of the vertical center line. Naturally, in the actual implementation, we incorporate that. Likewise for the special cases when an object of interest is behind the UAV, which happens for gimbal angles close to 90°.

Furthermore, we compute the horizon line in image space as outlined in the previous chapter.

8.2.2 Absolute Coordinates

Using the relative coordinates of an object (x- and y- ground-distances to UAV), we compute its GPS coordinates based on the UAV's GPS coordinates as follows. Given the camera heading angle θ, we compute the rotation matrix and rotate the relative coordinates of an object to obtain

$$\begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{8.4}$$

Finally, we map the relative coordinates to GPS coordinates via

$$la^{object} = la + \frac{y_r}{r}\frac{180}{\pi}, \tag{8.5}$$

$$lo^{object} = lo + \frac{x_r}{r}\frac{180}{\pi}\frac{1}{\cos(lat\ \pi/180)}. \tag{8.6}$$

We note that we compute the earth radius based on the latitude to account for the slight ellipsoid nature of the earth. Lastly, note that we can reverse the computation to map from 3D to 2D, i.e. the image space.

8.3 Method: Temporal Memory

Classical grid maps in robotics, such as occupancy grid maps, divide the environment into single grid cells and estimate the occupancy probability for each cell (Nuss *et al.* (2018)). We also aim to build temporary memory maps but in the context of modeling dynamic objects of interest. While there are several works extending occupancy grid maps to account for dynamic objects, e.g. Chung and Huang (2010), our focus is on leveraging maps to improve the downstream performance of computer vision detection and tracking algorithms over time. Hence, we propose a simple class of what we denote *temporal memory maps*.

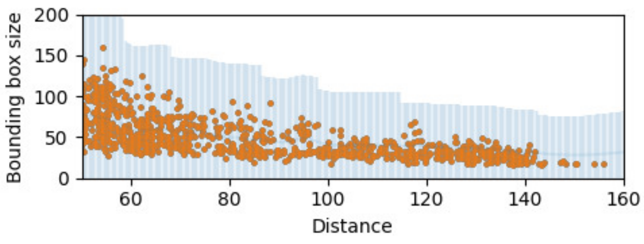


Figure 8.2: Distance (x-axis) to size (y-axis) relation for all swimmers in SeaDronesSee object detection train set (orange dots). The blue area denotes the accepted area of sizes.

8.3.1 Map Representation

We consider a UAV-centric map with a fixed size context window, i.e., given the latitude la_t and longitude lo_t of the UAV at discrete time step t, the memory map is defined to be

$$M_t = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \middle| \begin{matrix} x \in \{la_t - \frac{c}{2} + \frac{j}{n} | j \in 0, \ldots, n(la_t + \frac{c}{2})\} \\ y \in \{lo_t - \frac{c}{2} + \frac{j}{n} | j \in 0, \ldots, n(lo_t + \frac{c}{2})\} \end{matrix} \right\}, \tag{8.7}$$

which denotes a quadratic, north-oriented map around la_t and lo_t of edge size c with n^2 equidistantly spaced elements. Note that c and n should be problem-dependent reasonable quantities. This treatment makes sure that the surrounding environment of the UAV can be considered at all times. Note that c should be large enough to account for the relevant field of view of the UAV and n should be large enough to have sufficient resolution. Furthermore, the map will forget information that is outside of the context window, which needs to be considered in downstream applications.

From here, the exact procedure of how to update and leverage the map differs between the computer vision tasks. Broadly speaking, all methods define a memory map on M_t, indicating likelihoods of object locations.

8.3.2 Video Object Detection

After analyzing single-image object detectors, we found that most trained object detectors can detect almost all objects before the non-maximum suppression (NMS) stage, which in our case is doing both, removing duplicate boxes and low-confidence ones. Perhaps unsurprisingly, this is due to object detectors distributing many anchors across the image, from which potential objects are regressed. However, detections tend to cluster around actual objects, thereby achieving theoretical recall values of close to one.

An immediate observation yields that we can filter out predictions based on their sizes. Common object detectors employ multiple stages, responsible for multiple scales. Often, the detectors output predictions, whose sizes are far off from any reasonable quantities. By looking at the distance d and the class indicator of any predicted object, we establish

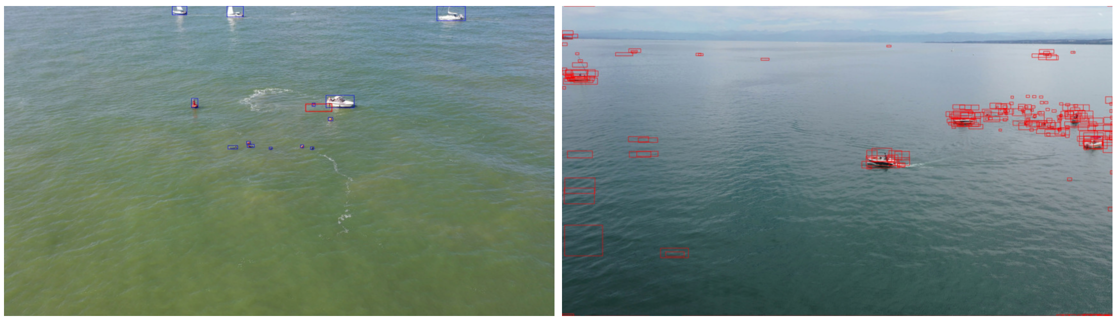


Figure 8.3: **Left**: Blue boxes are true positive detections, the red box is a false positive swimmer detection that was successfully filtered out by analyzing the distance-size relation. **Right**: Before NMS, there are many low-confidence detections, but they cluster around actual objects. Ignoring the confidence, these detections would yield a recall value of one. For visualization, only one-colored boxes.

Model name	#B	Rem.FP	Rem.TP	%
Maritime-VSA (Kiefer *et al.* (2023))	674K	8,087	24	1.2
DetectoRS (Kiefer *et al.* (2023))	838K	6,705	67	0.8
YOLOv7-Sea (Kiefer *et al.* (2023))	378K	4,926	54	1.3

Table 8.1: Numbers of removed (Rem.) boxes (B) for best three models trained and tested on SeaDronesSee Object Detection v2.

a distance-to-size (size measured by diameter of the box) relation and analyze it on the SeaDronesSee object detection train set (see Figure 8.3). For each class separately, we perform a Gaussian process regression on these data points. This yields a mean function $m(x)$ and a covariance function $cov(x)$. We scale the covariance function by a factor dependent on the distance from the maximal to the minimal bounding box size in a small interval around x. Finally, we check whether a new prediction is inside the resulting accepted area, and if not, we discard that prediction (see Figure 8.2). Applying this procedure on the three best models of the SeaDronesSee Object Detection v2 challenge, Table 8.1 shows that we can already remove a small amount of FPs.

Nevertheless, there needs to be a suppression stage because the high recall still comes at the cost of many false positive bounding boxes. We argue that robust detection can be achieved by temporally and geometrically sensible aggregation of many of the low-confidence detections that cluster around actual objects. Since we do this aggregation in actual GPS space, we can cumulate likelihoods from different viewpoints, hence this treatment is theoretically invariant to camera movements, which often cause traditional methods, tracking features across frames, to fail (e.g. compare to Chapter 3).

To provide evidence for the initial claim about the high recall before NMS, we train a

Faster R-CNN ResNet-18 and a YOLOv7 (configs from Kiefer *et al.* (2023)) on SeaDronesSee v2 (Kiefer *et al.* (2023)). For evaluation, we remove the NMS stage and obtain recall values of 97.4 % and 98.8%, resp. Compare that to the recall after NMS stage: 86.4% resp. 87.6%. See also Figure 8.3 for a sample image showing pre-NMS detections.

This leads us to hypothesize that a large part of missing detections can be obtained (and also false detections can be suppressed) if the confidence scores are modified such that low-confidence detections in *GPS areas*, where there have been detections for several consecutive time steps, are boosted, and, simultaneously, (semi-)confident false positive detections are suppressed if they only appear in certain frames (randomly).

Memory Map Construction:

Therefore, we propose to map all detections before NMS to GPS space as follows. For every box detection $b_i = [x_1^i, x_2^i, y_1^i, y_2^i]$, $i = 1, \dots, m$, we take its bounding box center $[(x_2^i + x_1^i)/2, (y_2^i + y_1^i)/2]$ and compute its GPS coordinates $g^{b_i} = [la^{b_i}, lo^{b_i}]$ via the formulas in Section 8.2. We compute the closest grid window to g^{b_i} via

$$w^{b_i} = \arg\min_{m \in M_t} ||g^{b_i} - m||. \tag{8.8}$$

Ideally, this is the actual GPS position of that prediction. However, to account for imprecision in the sensor data and deviations from taking the center of the bounding box (which may not be the GPS center of the object), we assign weight to the neighbouring space as well, albeit less, since w^{b_i} is our best single-point prediction[1]. Hence, we construct a memory map likelihood over M_t as follows: We add a truncated Gaussian density with radius r at w^{b_i}, i.e.,

$$\tilde{p}_t(x) \leftarrow \tilde{p}_t(x) + se^{-x^2} \text{ for } x \in [w^{b_i} - r, w^{b_i} + r] \subset M_t, \tag{8.9}$$

where we slightly abuse the notation by mixing scalars and vectors (computation to be understood component-wise). Note that we hid the other parameters (constants and variance) in the scaling factor s, which we will *additionally* make dependent on the confidences c_i belonging to b_i.

After every time step, we truncate $\tilde{p}$, such that the memory map values are ≤ 1 at all times:

$$p_t(x) = \min(\tilde{p}_t(x), 1). \tag{8.10}$$

Furthermore, we introduce a forgetting factor ϕ, by which the memory map p_t is rescaled after each update step so that the memory map will not be overloaded over time.:

$$p_t(x) \leftarrow \phi p_t(x). \tag{8.11}$$

[1]Naively applying this method on all boxes before filtering of incorrectly sized boxes leads to too large or small bounding boxes becoming boosted.

Note that we explicitely did not model p_t as a probability distribution over M_t as this would introduce a competition of weights among the objects leading to deteriorated results when the number of objects and their prediction certainty vary (similar to a discrete Bayes filter (Fox *et al.* (2003)), although here, we do not have a control update).

Adjusting the Detector Confidences According to p_t:

At time t, each bounding box b_i has a confidence $c_i \in [0,1]$, which we will update based on p_{t-1} via

$$\hat{c}_i = c_i + p_{t-1}(w^{b_i}). \tag{8.12}$$

Note that we do not update the map based on the updated confidence $\hat{c}_i$ but on the original c_i. Lastly, note that we perform this procedure *class-wise*, so we keep a map for each class separately. After the procedure, we proceed with standard NMS.

8.3.3 Extension to Object Tracking and Reidentification

Similarly as for VOD, a *temporal memory map* helps in tracking scenarios. We apply the memory map to object tracking via a simple extension.

For frame t, we use the GPS location of each tracked object in frame $t-1$. We take the boosted predictions of frame t closest to the old GPS location with a confidence above a threshold as the new location (in image and GPS space) of the object to be tracked. We will see that this simple formulation of a tracker helps to track objects over time in presence of camera movements, as Chapter 3 showed. Even if a fast camera movement yielding blurry frames results in the object detector missing the object temporally - if an object will be redetected in subsequent frames - the object can be assigned the correct ID again.

While there is benefit in using memory maps for *short-term* tracking tasks, they are especially beneficial in *long-term* tracking, where ground truth objects may leave the frame entirely. The task of reidentification in classical scenarios is solved by feature-based comparison of objects across time. In UAV-based domains, this often fails for small objects as features look similar across different objects. However, the treatment in GPS space allows us to remember where an object was, allowing us to reidentify it in subsequent frames over a pre-defined time horizon.

8.3.4 Extension to Video Anomaly Detection

The goal of video anomaly detection is to output regions of the frames that are considered anomalous, either spatially (from appearance) or temporally (unseen movements). As discussed previously, may assume anomalies to be temporally stable, i.e. they exist over a considerable period of time. This makes it ideal to aggregate information at certain GPS positions over a sequence of frames. Since it is hard to explicitly define regions that are anomalous, most methods output an anomaly heatmap, indicating areas of likely anomalies. Therefore, we propose to map the entire image plane to GPS space and aggregate anomalous regions over time in a geometrically reasonable way. We note that this requires an efficient vectorized implementation to still achieve real-time inference. Section 8.4.3 discusses this shortly.

For the correct mapping of pixels to GPS coordinates, we need to make sure that the projection is well-defined. Therefore, we apply the previously introduced horizon cutter, which first computes the horizon and only maps the pixels below it to GPS space. We ignore pixels too far in the horizon as this results in too much noise.

We replace the 2D difference heat map from previous chapter with our memory map, which will be added to the previous memory map. After each step, we also apply a forgetting factor ϕ. Afterwards, we apply the same post-processing steps as indicated there.

8.4 Results and Analysis

We test on SeaDronesSee-MOT and PeopleOnGrass-Video. Note that we focus on the detection of people in both benchmarks only, since this is the hardest class to detect, as Chapter 3 revealed. For that, we fuse the classes *swimmers*, *swimmers with life jacket* and *life jacket* into a single people class in SeaDronesSee-V (SeaDronesSee-MOT without instance IDs).

For short-term tracking, we also employ SeaDronesSee-MOT. It currently only supports short-term tracking, i.e. objects that left the scene and reappear do not have the same id. Therefore, we relabel a video clip of SeaDronesSee-MOT, such that it also supports long-term tracking, which we need to assess the performance of long-term tracking (reidentification).

For Video Anomaly Detection, we fall back to the OpenWater data set of the Maritime Anomaly Detection Benchmark as dicussed in previous chapter, since it is the only one featuring metadata.

Finally, we perform additional experiments to evaluate an extension to multiple UAVs in a collaborative scenario.

8.4.1 Video Object Detection

We apply the proposed memory map on the output of a YOLOv7-Tiny (Wang *et al.* (2022)) trained and tested on SeaDronesSee-V and POG-V, respectively. We choose a grid with 0.5m resolution and a size of 300m. We truncate the Gaussian updates with a radius of 6m.

As baselines, we take the well-known Video Object Detectors Deep Feature Flow (DFF) (Zhu *et al.* (2017b)), Flow-guided Feature Aggregation (FGFA) (Zhu *et al.* (2017c)) and the recent Temporal RoI Align (T.R.A.) (Gong *et al.* (2021)) as implemented in Chen *et al.* (2019b) with their default configuration.

	SeaDronesSee-VOD		POG-V		
Model name	AP_{50}	AR	AP_{50}	AR	FPS
DFF (Zhu *et al.* (2017b))	53.7	34.2	30.3	18.8	3.4
FGFA (Zhu *et al.* (2017c))	53.9	34.1	34.7	24.8	1.6
T.R.A. (Gong *et al.* (2021))	60.6	42.9	29.9	19.3	2.4
YOLOv7-Tiny (Wang *et al.* (2022))	68.2	41.1	81.3	36.2	**25.1**
YOLOv7-Tiny + Mem. Map	**72.8**	**44.0**	**84.4**	**38.8**	**25.1**

Table 8.2: Video Object Detection accuracy for Swimmers and People on SeaDronesSee-Video and POG-Video, respectively. The last column denotes running times (wall-clock time in frames per second) benchmarked on an Nvidia Xavier AGX.

Table 8.2 shows that incorporating our memory maps leads to increased AP_{50} and AR (average recall with averaged IoU levels 0.5:0.05:0.95 and at most 100 detections) values by successfully boosting the confidence scores. In particular, we achieve a +4.6 and +3.1 AP_{50} increase over the single-image YOLOv7-Tiny object detector on SeaDronesSee-V and POG-V, respectively. Interestingly, popular video object detectors lack behind in performance because they are not targeted to aerial VOD. Moreover, the increased AR values shows that our method detects more objects, which is the main challenge in UAV-based detection.

Figure 8.4 visualizes the temporal memory map, projected to the image space. It shows the heatmap values of likely object locations, which were aggregated from the previous frames. The red areas cluster around actual objects. This results in pre-NMS boxes becoming boosted, as Figure 8.4 shows. While the baseline YOLOv7-Tiny originally did not detect a single swimmer, now we detect eight.

Furthermore, the standard video object detectors are not suitable for deployment on embedded devices. Our method only adds a negligible time to a standard YOLOv7-Tiny, which can run in real-time on an Nvidia Xavier AGX.

8.4.2 Object Tracking

Applying our method on SeaDronesSee-MOT yields an increase in HOTA and MOTA and a decrease of ID switches and fragmentations over the DeepSORT short-term tracker (see Table 8.3).

To test the reidentification capability, we run our method on top of a DeepSORT with reidentification module on the relabeled video as described earlier. We can successfully reassign the same id to all the objects in the video whereas the baseline fails to do so (4 people are reassigned new IDs.)

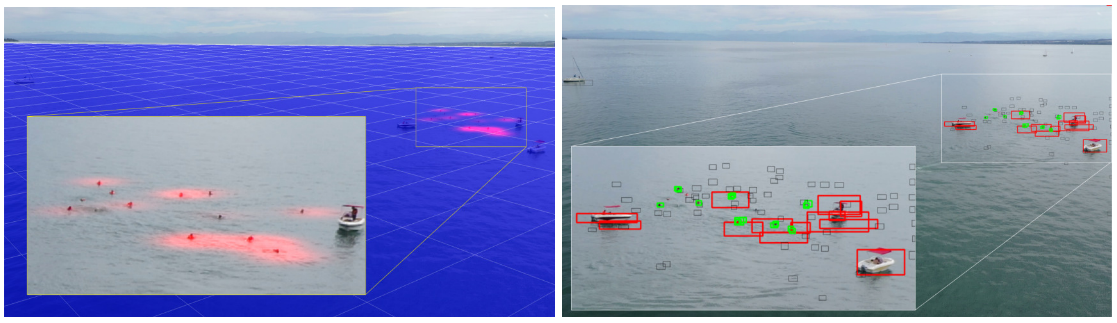


Figure 8.4: **Left**: Heatmap illustration of memory map on a SeaDronesSee-V test frame. We project actual GPS latitude and longitude lines to image space. We also project the memory map from GPS space to image space and visualize it as a heatmap. The right image part shows that the memory map successfully puts weight around actual swimmers' locations. **Right**: Before NMS, many swimmers are predicted (black boxes), several of which are automatically removed due to incorrect sizes (red boxes). Only the confidences of the green boxes are boosted. Applying a confidence threshold of 0.5 yields that 8 swimmers are detected as opposed to none for the baseline (YOLOv7-Tiny).

Model name	HOTA↑	MOTA↑	IDs↓	Frag↓
ByteTracker (Kiefer *et al.* (2023))	65.0	76.9	68	841
DeepSORT (Kiefer *et al.* (2023))	66.6	80.0	44	805
DeepSORT +Memory Map	**67.2**	**80.8**	**35**	**721**

Table 8.3: MOT accuracy on SeaDronesSee-MOT. We used the output of DeepSORT and built on top a mem. map resulting in fewer id switches and fragmentations.

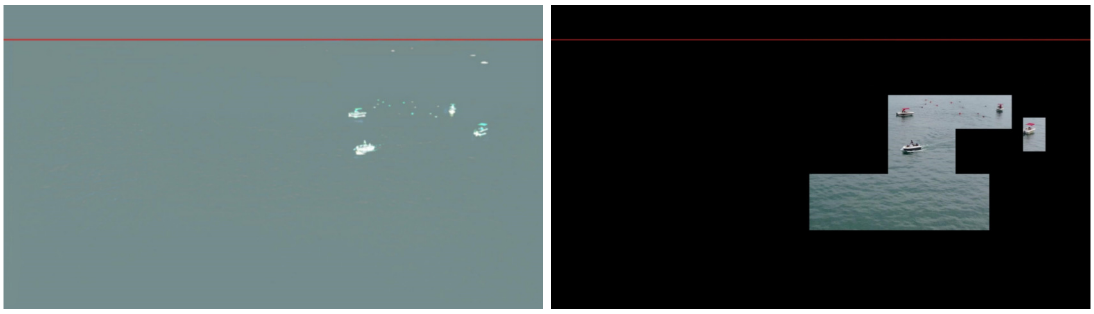


Figure 8.5: Difference anomaly map aggregated in 3D space over several frames (left) and resulting regions of interested (right).

Model name	AR↑	$R^{p=5\%}$ ↑	FPS
Gaussian Mixture Model (Kiefer *et al.* (2023))	45.9	2.6	17
Mean Filter (Kiefer *et al.* (2023))	73.9	54.3	50
Frame Differencing (Kiefer *et al.* (2023))	76.1	54.8	62
Autoencoder (Kiefer *et al.* (2023))	79.8	71.0	27
Autoencoder + Memory Map	**82.8**	**74.8**	27

Table 8.4: Video Anomaly Detection accuracy on SeaDronesSee-MOT. We built our memory maps on top of the Autoencoder from previous chapter.

8.4.3 Video Anomaly Detection

Table 8.4 shows that adding the module to the anomaly detector from previous chapter yields an increase of +3 AR to 82.8 (average recall over multiple levels of broadcasting rate; measured differently than average recall from object detection, please see previous chapter). This performance increase is also reflected in the recall at broadcasting rate of 5%, i.e. there is a +3.8 $R^{p=5\%}$ improvement over the Autoencoder.

Figure 8.5 shows the resulting difference anomaly map aggregated over the last frames and the resulting regions of interest that are returned.

The speed does not increase considerably over the Autoencoder, still achieving real-time inference benchmarked on an Nvidia Xavier AGX as Table 8.4 shows.

8.4.4 Cooperative Detection via Multiple UAVs

The GPS memory map allows for a joint representation of objects' locations. In this section, we demonstrate that it can be leveraged in a collaborative setting with multiple UAVs. This allows for cross-UAV knowledge transfer and leads to more robust predictions. Furthermore, it allows for detections of (partially) occluded objects. For example, Figure 8.6 shows the same scenery captured from different locations at the same time

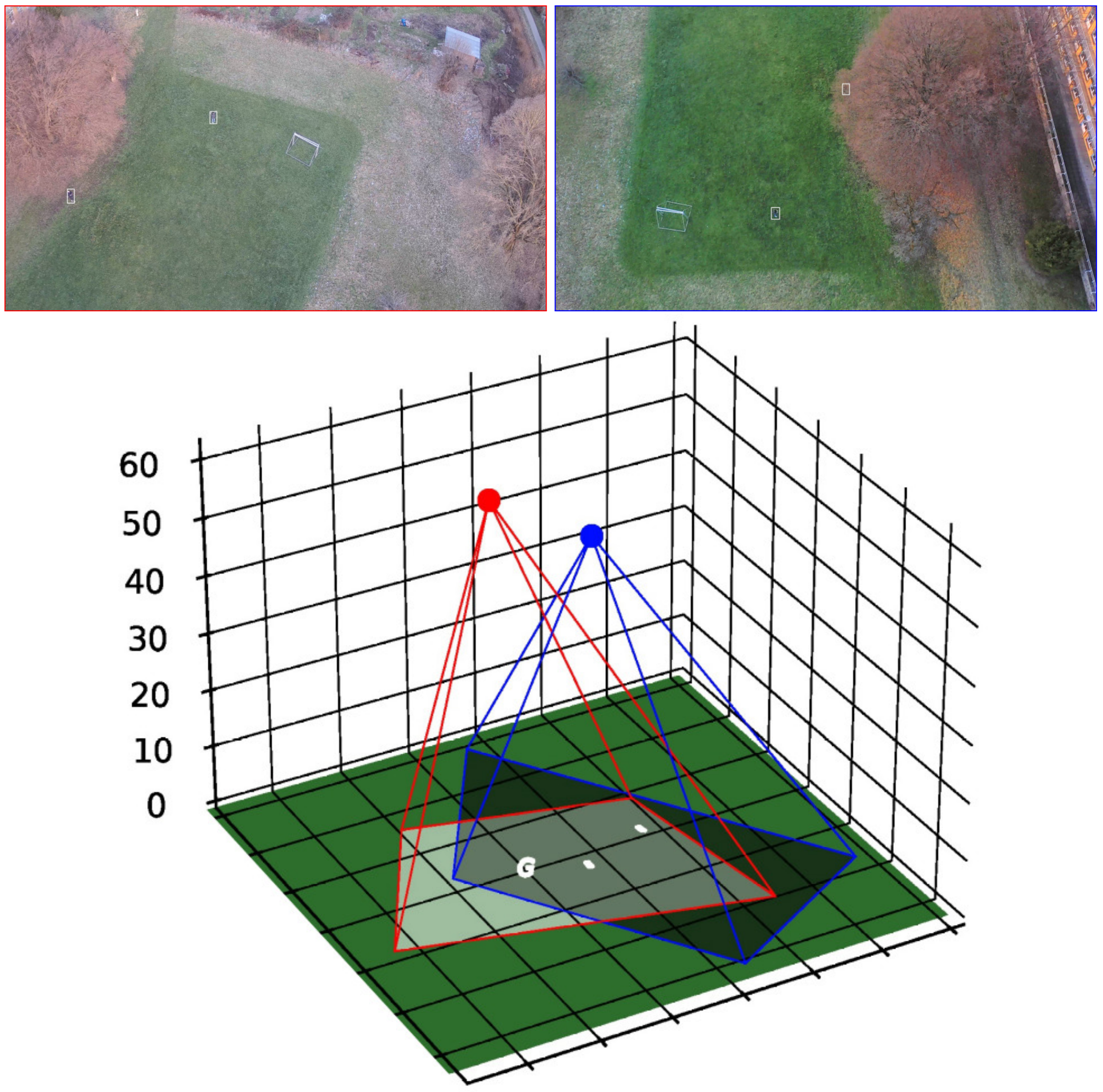


Figure 8.6: Two UAVs inspect the same scenery from different locations at the same time. The red camera (left) has a good view on both objects and the goal (G), while the blue camera (right) is underexposed (brightened for visualization) and one person is partially occluded by a tree. We aggregate both memory maps together to obtain more robust detections. Without considering the underlying geometry, this is not possible with standard video object detectors.

with two people are walking on the grass.

Having a joint location likelihood representation allows the UAVs to share information. We apply the same memory map from before on both UAVs' object detectors' output, but this time, we average their memory maps in the overlapping region. For that, we take an EfficientDet-D0 trained on POG as in Chapter 4 and test it on the following data. We capture four minutes of footage in a similar environment as POG from the viewpoint of two UAVs, a DJI Matrice 210 and a DJI Mavic 2 Pro, denoted *2AVs*. We varied variables, such as altitude, pitch and heading viewing angle and GPS location. We annotated the people visible in both video streams and compare the performance of the single-frame object detector with the memory map from before. We obtain an AP_{50} of 61.3 for the single-frame object detector compared to an AP_{50} of **68.9** for the memory map.

To shot the utility of a joint memory map for tracking, we compare a fast single-object tracker, PrDiMP18 (Danelljan *et al.* (2020)), to a simple tracker based on the memory map. For that, we apply an EfficientDet-D0 on a consecutive subset of 2AVs that contains partial occlusions of a person in one video stream (blue camera in Fig. 8.6). While the baseline tracker cannot handle the occlusion and fails to track the person behind the tree entirely (4.3 AP_{50}) for the blue camera, our method leverages the joint memory map that transfers knowledge from the red UAV to the blue (86.3 AP_{50}).

The joint memory map also allows for reidentification in long-term tracking tasks *while the object is moving*. To test this, we take a standard DeepSORT trained on POG with the default Reid model within mmtracking (Chen *et al.* (2019b)) as baseline. We take a subset of 2AVs where one camera leaves the scenery entirely, while the other is tracking the object throughout. When reappearing, it immediately uses the track id information from the other camera, while DeepSORT failed to reidentify the object, which we believe to be attributed to the object size.

8.5 Conclusion and Discussion

While not using meta-data remains the standard in computer vision on UAVs, using metadata to boost the performance shows promising results without inducing a large computational overhead. We showed that we can improve standard methods by considering the underlying 3D geometry. Reasoning about detections, tracklets and anomalies in an interpretable and robust way allows for more trustworthy methods.

Although in our experiments on POD-Video we did not encounter any problems with terrain elevation change, future work remains to show how strict the assumption of even-level terrain is on other benchmarks. For that, it is inevitable to collect larger and more diverse benchmarks.

In future works, it will be interesting to see how a symbiosis of metadata and other computer vision tasks looks like. Lastly, it seems relevant to dive into the topic of uncertainty quantification methods to account for the errors in metadata values.

Chapter 9

Conclusions & Outlook

In this work, we discussed computer vision on UAVs from a holistic standpoint. Instead of ignoring the environment around the UAV and thereby treating the computer vision problems domain-agnostic, we explicitly take it into account. We analyzed several problems in the context of computer vision on UAVs and provided techniques and methods to overcome or alleviate these. Furthermore, we presented multiple computer vision methods that are capable of running in real-time on a consumer UAV. We also looked at data acquisition and the actual deployment on embedded hardware, hoping to progress the field from theoretical-only benefits to actual practical utility.

In particular, in Chapter 2, we outlined the full pipeline for data acquisition and labeling, which is major shortcoming in current research for computer vision on UAV. We released a major benchmark for UAV-based maritime computer vision supporting several tasks, such as object detection, single-object- and multi-object-tracking. We discussed the intricacies when planning missions and also how to efficiently capture corresponding metadata. The latter is not featured in current benchmarks and our works were the first to release a large-scaled benchmark to include this data. The release of these benchmarks opened the door for research on multi-modal holistic computer vision on UAVs.

Using these captured benchmarks, we analyzed the state-of-the art in Chapter 3. We compared various object detectors and trackers with respect to their robustness and speed. In particular, we organized the *1st Workshop on Maritime Computer Vision* to progress the research in this field and to obtain a better understanding of current limitations.

These baselines models helped us in the subsequent chapters to measure the advancements we can make using metadata. Chapter 4 discussed a major challenge in object detection on UAVs: domain imbalance and the resulting domain bias. We analyzed this unexplored phenomenon that occurs in most UAV-based benchmarks and showed how we can mitigate it by means of using metadata. By introducing expert heads, we show that we can design models much more robust without increasing the inference time.

In Chapter 5, we go one step further and concentrate on the prominently occurring scale variance problem. Scale variance is challenge that is very particular to UAV-based data sets, where objects' sizes may vary immensely. Models that do not consider this issue may overfit to certain altitudes or sizes which often is the case in imbalanced data sets. Therefore, we introduced a pre-processing technique applicable to bird's eye view imagery that essentially yields scale invariant object detectors. As models only ever see same-sized objects, the generalization performance is increased drastically. Furthermore, it results in faster detectors and altitude knowledge transfer.

However, the previous chapter once more illustrated that the lack of data in the UAV domain makes it hard to develop and compare methods. For this reason, in Chapter 6, we introduced a new synthetic data generation tool based on the video game GTAV. We demonstrated how we can leverage it to obtain high-resolution UAV-based imagery of high fidelity, with many realistically looking objects in a diverse range of environments. Importantly, we showcase how we obtain pixel-perfect ground truth annotation of all dynamic objects in the world. These image-level annotations come together with meta-data-level annotations, such as altitude, viewing angle, IMU information and more. We demonstrated that this synthetic data may be used to train models that can be deployed in the real-world. FOr optimal performance, we showed that we can obtain best model performance by consecutively training on synthetic and real data. Last but not least, we showed how metadata allows us to obtain more efficient synthetic data by constraining the image data on relevant footage.

Chapter 7 took the first step in the direction of weakly supervised and temporal data. For the particular task of video anomaly detection in the maritime domain, we show that classical methods fail to detect anomalies reliably. We designed an efficient end-to-end video anomaly detection architecture and described how we use metadata to train and run it efficiently. Considering the metadata allowed us to compute the horizon line efficiently so that the autoencoder architecture could effectively learn the appearance and dynamics of water. This allowed it to detect unseen objects and movement downstream.

Lastly, Chapter 8 went the final step towards considering the surrounding 3D geometry. Taking into account the metadata across time allowed us to create memory maps of our surrounding. These memory maps helped in understanding our surrounding environment which we leverage to obtain more robust video object detection models. We showed that a environmentally-aware detector is much more robust towards uncertain appearances by aggregating predictions over time in 3D space as opposed to image space. Furthermore, we showed that a correct treatment of the surrounding lead to more robust video anomaly detectors and in a collaborative scenarios.

From the current state of research, many avenues can be explored. Hopefully, with more emerging benchmarks, which do include metadata, we may analyze which computer vision methods can be improved by means of this data. It seems in many ways that the robotics and the computer vision communities are research separately. However, we argue that a joint development is inevitable for robust and intelligent agents. In particular, mapping methods, which are primarily a robotics topic, and detection of dynamic objects, seem to be treated separately even though there is a great overlap. We hope that our methods helped to bridge some of the gap between domain-agnostic and holistic methods.

Abbreviations

A	Acute
AP	Average Precision
AR	Average Recall
AUto	Autoencoder
FD	Frame Differencing
FPS	Frames per second
GMM	Gaussian Mixture Model
H	High
L	Low
mAP	Mean Average Precision
M	Medium
MAV	Micro Aerial Vehicle
MF	Mean Filter
MOT	Multi-Object Tracking
OD	Object Detection
POG	PeopleOnGrass
R	Right
SaR	Search and Rescue
SOT	Single-Object Tracking
UAV	Unmanned Aerial Vehicle
VOD	Video Object Detection
WACV	Winter Conference on Applications of Computer Vision

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 (2022). Airbus Ship Detection Challenge. `https://www.kaggle.com/c/airbus-ship-detection`. Accessed: 2022-07-05.

Aiskyeye (2022). VisDrone. Vision Meets Drones: A Challenge. `http://aiskyeye.com/`. Accessed: 2022-07-05.

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*.

Angus, M., ElBalkini, M., Khan, S., Harakeh, A., Andrienko, O., Reading, C., Waslander, S., and Czarnecki, K. (2018). Unlimited road-scene synthetic annotation (ursa) dataset. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 985–992. IEEE.

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.

App-Airdata (2022). Airdata for UAV. `https://app.airdata.com/`. Accessed: 2021-03-01.

Avalon (2022). Avalon Project. `https://seadronessee.cs.uni-tuebingen.de/avalon`. Accessed: 2022-07-05.

Balestrieri, E., Daponte, P., De Vito, L., Picariello, F., and Tudosa, I. (2021). Sensors and measurements for uav safety: An overview. *Sensors*, **21**(24), 8253.

Barto, C. (2022). Gta vision export. `https://github.com/umautobots/GTAVisionExport`. Accessed: 2022-08-01.

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.

Belmonte, L. M., Morales, R., and Fernández-Caballero, A. (2019). Computer vision in autonomous unmanned aerial vehicles—a systematic mapping study. *Applied Sciences*, **9**(15), 3196.

Benezeth, Y., Jodoin, P.-M., Emile, B., Laurent, H., and Rosenberger, C. (2010). Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, **19**(3), 033003.

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

Bevacqua, G., Cacace, J., Finzi, A., and Lippiello, V. (2015). Mixed-initiative planning and execution for multiple drones in search and rescue missions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 25, pages 315–323.

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.

Blade, A. (2022). Script Hook V. `http://www.dev-c.com/gtav/scripthookv/`. Accessed: 2022-08-01.

Blikharskyy, Y., Kopiika, N., Khmil, R., Selejdak, J., and Blikharskyy, Z. (2022). Review of development and application of digital image correlation method for study of stress–strain state of rc structures. *Applied Sciences*, **12**(19), 10157.

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

Bohren, C. F. and Fraser, A. B. (1986). At what altitude does the horizon cease to be visible? *American Journal of Physics*, **54**(3), 222–227.

Bolya, D., Foley, S., Hays, J., and Hoffman, J. (2020). Tide: A general toolbox for identifying object detection errors. In *European Conference on Computer Vision*, pages 558–573. Springer.

Bosquet, B., Mucientes, M., and Brea, V. M. (2021). Stdnet-st: Spatio-temporal convnet for small object detection. *Pattern Recognition*, **116**, 107929.

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: Software Tools for the Professional Programmer*, **25**(11), 120–123.

Cai, Z. and Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162.

Cao, Y., He, Z., Wang, L., Wang, W., Yuan, Y., Zhang, D., Zhang, J., Zhu, P., Van Gool, L., Han, J., *et al.* (2021). Visdrone-det2021: The vision meets drone object detection challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2847–2854.

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.

Cazzato, D., Cimarelli, C., Sanchez-Lopez, J. L., Voos, H., and Leo, M. (2020). A survey of computer vision methods for 2d object detection from unmanned aerial vehicles. *Journal of Imaging*, **6**(8), 78.

Chen, G., Wang, W., He, Z., Wang, L., Yuan, Y., Zhang, D., Zhang, J., Zhu, P., Van Gool, L., Han, J., *et al.* (2021a). Visdrone-mot2021: The vision meets drone multiple object tracking challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2839–2846.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., *et al.* (2019a). Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. (2019b). MMDetection: Open MMLab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Chen, K., Wu, M., Liu, J., and Zhang, C. (2020). Fgsd: A dataset for fine-grained ship detection in high resolution satellite images. *arXiv preprint arXiv:2003.06832*.

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

Chen, Y., Li, W., Sakaridis, C., Dai, D., and Van Gool, L. (2018). Domain adaptive faster r-cnn for object detection in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3339–3348.

Chung, S.-Y. and Huang, H.-P. (2010). Slammot-sp: simultaneous slammot and scene prediction. *Advanced Robotics*, **24**(7), 979–1002.

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.

Corsel, C. W., van Lier, M., Kampmeijer, L., Boehrer, N., and Bakker, E. M. (2023). Exploiting temporal context for tiny object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 79–89.

Crisp, D. J. (2004). The state-of-the-art in ship detection in synthetic aperture radar imagery. Technical report, Defence Science And Technology Organisation Salisbury (Australia) Info ....

Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., and Zhang, L. (2021). Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7373–7382.

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.

Darkpgmr (2022). DarkLabel Annotation Tool, Github. `https://github.com/darkpgmr/DarkLabel`. Accessed: 2022-07-05.

Deecke, L., Vandermeulen, R., Ruff, L., Mandt, S., and Kloft, M. (2018). Image anomaly detection with generative adversarial networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 3–17. Springer.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., and Leal-Taixé, L. (2020). Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*.

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*.

Division, F. S. (2022). Gtav mod - heap limit adjuster. https://de.gta5-mods.com/tools/heap-limit-adjuster-600-mb-of-heap. Accessed: 2022-08-01.

Dorahamu (2022). Gtav mod - simple increase traffic and pedestrians. https://de.gta5-mods.com/misc/simple-increase-traffic-and-pedestrian. Accessed: 2022-08-01.

Dortmund, T. (2022). Larus Project. `http://larus.kn.e-technik.tu-dortmund.de/`. Accessed: 2022-07-05.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

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, Y., Wan, J., Zhao, Y., Zhang, B., Tong, Z., and Dong, J. (2021). Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2809–2819.

Du et. al., D. (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.

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.

ExposingAI (2022). Ms-celeb-1m privacy infringment. https://exposing.ai/msceleb/. Accessed: 2022-08-01.

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.

Ferreira, N. and Silveira, M. (2020). Ship detection in sar images using convolutional variational autoencoders. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 2503–2506. IEEE.

Fonder, M. and Droogenbroeck, M. V. (2019). Mid-air: A multi-modal dataset for extremely low altitude drone flights. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*.

Fox, V., Hightower, J., Liao, L., Schulz, D., and Borriello, G. (2003). Bayesian filtering for location estimation. *IEEE pervasive computing*, **2**(3), 24–33.

French, G., Mackiewicz, M., and Fisher, M. (2017). Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208*.

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.

Games, R. (2022). Grand theft auto v. `https://www.rockstargames.com/de/games/V`. Accessed: 2022-08-01.

GDPR (2022). General data protection regulation. https://gdpr.eu/. Accessed: 2022-08-01.

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

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

GmbH, S. (2020). A pytorch implementation of efficientdet object detection. *`https://github.com/signatrix/efficientdet`*.

Gong, T., Chen, K., Wang, X., Chu, Q., Zhu, F., Lin, D., Yu, N., and Feng, H. (2021). Temporal roi align for video object recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1442–1450.

Gudovskiy, D., Ishizaka, S., and Kozuka, K. (2022). Cflow-ad: Real-time unsupervised anomaly detection with localization via conditional normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 98–107.

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.

Hayat, S., Yanmaz, E., Bettstetter, C., and Brown, T. X. (2020). Multi-objective drone path planning for search and rescue with quality-of-service requirements. *Autonomous Robots*, **44**(7), 1183–1198.

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.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR.

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.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

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.

Hurl, B. (2022). DeepGTAV-PreSIL Code. `https://github.com/bradenhurl/DeepGTAV-PreSIL`. Accessed: 2022-08-01.

Hurl, B., Czarnecki, K., and Waslander, S. (2019). Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529. IEEE.

Ikeuchi, K. (2021). *Computer vision: A reference guide*. Springer.

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.

Jocher, G., Changyu, L., Hogan, A., Yu, L., changyu98, Rai, P., and Sullivan, T. (2020). ultralytics/yolov5: Initial Release. *Github Pages*.

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., and Vasudevan, R. (2016). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*.

Kanellakis, C. and Nikolakopoulos, G. (2017). Survey on computer vision for uavs: Current developments and trends. *Journal of Intelligent & Robotic Systems*, **87**, 141–168.

Kar, A., Prakash, A., Liu, M.-Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A., and Fidler, S. (2019). Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4551–4560.

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. (2022). SeaDronesSee Benchmark data acquisition. `https://seadronessee.cs.uni-tuebingen.de/dataacquisition`. Accessed: 2022-07-05.

Kiefer, B., Ott, D., and Zell, A. (2021). Leveraging synthetic data in object detection on unmanned aerial vehicles. *arXiv preprint arXiv:2112.12252*.

Kiefer, B., Messmer, M., and Varga, L. (2022). SeaDronesSee Benchmark. `https://seadronessee.cs.uni-tuebingen.de/`. Accessed: 2022-07-05.

Kiefer, B., Kristan, M., Perš, J., Žust, L., Poiesi, F., Andrade, F., Bernardino, A., Dawkins, M., Raitoharju, J., Quan, Y., *et al.* (2023). 1st workshop on maritime computer vision (macvi) 2023: Challenge results. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 265–302.

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.

Kong, F., Huang, B., Bradbury, K., and Malof, J. (2020). The synthinel-1 dataset: a collection of high resolution synthetic overhead imagery for building segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1814–1823.

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.

Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernández, G., Nebehay, G., Porikli, F., and Čehovin, L. (2016). A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **38**(11), 2137–2155.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. (2014). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.

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.

Li, C.-L., Sohn, K., Yoon, J., and Pfister, T. (2021). Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9664–9674.

Li, P., Liang, X., Jia, D., and Xing, E. P. (2018a). Semantic-aware grad-gan for virtual-to-real urban scene adaption. *arXiv preprint arXiv:1801.01726*.

Li, Q., Mou, L., Liu, Q., Wang, Y., and Zhu, X. X. (2018b). 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 *European conference on computer vision*.

Lin et. al., T.-Y. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

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

Lin et. al., T.-Y. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

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

Liu, W., Luo, W., Lian, D., and Gao, S. (2018). Future frame prediction for anomaly detection–a new baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6536–6545.

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*.

Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., and Leibe, B. (2021). Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, **129**(2), 548–578.

Luo, H., Xie, W., Wang, X., and Zeng, W. (2019). Detect or track: Towards cost-effective video object detection/tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8803–8810.

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.

Marques, M. M., Dias, P., Santos, N. P., Lobo, V., Batista, R., Salgueiro, D., Aguiar, A., Costa, M., da Silva, J. E., Ferreira, A. S., *et al.* (2015). Unmanned aircraft systems in maritime operations: Challenges addressed in the scope of the seagull project. In *OCEANS 2015-Genova*, pages 1–6. IEEE.

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

Menouar, H., Guvenc, I., Akkaya, K., Uluagac, A. S., Kadri, A., and Tuncer, A. (2017). Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, **55**(3), 22–28.

Messmer, M., Kiefer, B., and Zell, A. (2021). Gaining scale invariance in uav bird's eye view object detection by adaptive resizing. *arXiv preprint arXiv:2101.12694*.

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

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016b). MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*. 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.

Mittal, P., Singh, R., and Sharma, A. (2020). Deep learning-based object detection in low-altitude uav datasets: A survey. *Image and Vision computing*, **104**, 104046.

Mohamed, S. S., Tahir, N. M., and Adnan, R. (2010). Background modelling and background subtraction performance for object detection. In *2010 6th International Colloquium on Signal Processing & its Applications*, pages 1–6. IEEE.

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.

Nguyen, T.-N. and Meunier, J. (2019). Anomaly detection in video sequence with appearance-motion correspondence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1273–1283.

Nikolenko, S. I. *et al.* (2019). Synthetic data for deep learning. *arXiv preprint arXiv:1909.11512*, **3**.

Nuss, D., Reuter, S., Thom, M., Yuan, T., Krehl, G., Maile, M., Gern, A., and Dietmayer, K. (2018). A random finite set approach for dynamic occupancy grid maps with real-time application. *The International Journal of Robotics Research*, **37**(8), 841–866.

Nvidia (2022a). Nvidia geforce experience. https://www.nvidia.com/de-de/geforce/geforce-experience/. Accessed: 2022-08-01.

Nvidia (2022b). Nvidia Xavier. `https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-agx-xavier/`. Accessed: 2022-07-05.

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.

Ojanen, H. (1999). Automatic correction of lens distortion by using digital image processing. *Rutgers University, Dept. of Mathematics technical report*.

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*.

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., and Walsh, J. (2020). Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*, pages 128–144. Springer.

PapersWithCode (2022a). Object Detection on COCO test-dev. `https://paperswithcode.com/sota/object-detection-on-coco`. Accessed: 2021-03-01.

PapersWithCode (2022b). Object detection on coco test-dev. `https://paperswithcode.com/sota/`. Accessed: 2022-08-01.

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. *Proceedings - 2020 17th Conference on Computer and Robot Vision, CRV 2020*, pages 230–237.

Prakash, A., Boochoon, S., Brophy, M., Acuna, D., Cameracci, E., State, G., Shapira, O., and Birchfield, S. (2019). Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7249–7255. IEEE.

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.

Price, E., Lawless, G., Ludwig, R., Martinovic, I., Bülthoff, H. H., Black, M. J., and Ahmad, A. (2018). Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles. *IEEE Robotics and Automation Letters*, **3**(4), 3193–3200.

Qiu, W. and Yuille, A. (2016). Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer.

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*.

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, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (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.

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. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, **28**.

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.

Ren, S. e. a. (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.

Richter, S. R., AlHaija, H. A., and Koltun, V. (2021). Enhancing photorealism enhancement. *arXiv preprint arXiv:2105.04619*.

Ringwald, T., Sommer, L., Schumann, A., Beyerer, J., and Stiefelhagen, R. (2019). UAV-net: A fast aerial vehicle detector for mobile platforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

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.

Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., and Gehler, P. (2021). Towards total recall in industrial anomaly detection. *arXiv preprint arXiv:2106.08265*.

Ruano, A. (2022a). DeepGTAV Code. `https://github.com/aitorzip/DeepGTAV`. Accessed: 2022-08-01.

Ruano, A. (2022b). Deepgtav-vpilot code. https://github.com/aitorzip/VPilot. Accessed: 2022-08-01.

Rudol, P. and Doherty, P. (2008). Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. In *2008 IEEE aerospace conference*, pages 1–8. Ieee.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., *et al.* (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, **115**(3), 211–252.

Saligrama, V. and Chen, Z. (2012). Video anomaly detection based on local statistical aggregates. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2112–2119. IEEE.

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.

Scherer, J., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., Vukadinovic, V., Bettstetter, C., Hellwagner, H., and Rinner, B. (2015). An autonomous multi-uav system for search and rescue. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, pages 33–38.

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.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer.

Shao, W., Kawakami, R., Yoshihashi, R., You, S., Kawase, H., and Naemura, T. (2020). Cattle detection and counting in uav images based on convolutional neural networks. *International Journal of Remote Sensing*, **41**(1), 31–52.

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.

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.

Srivastava, K., Singh, A. K., and Hegde, G. M. (2019). Multi modal semantic segmentation using synthetic data. *arXiv preprint arXiv:1910.13676*.

Staar, B., Lütjen, M., and Freitag, M. (2019). Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, **79**, 484–489.

Steinert, F. and Stabernack, B. (2022). Architecture of a low latency h. 264/avc video codec for robust ml based image classification. *Journal of Signal Processing Systems*, pages 1–16.

Sultani, W., Chen, C., and Shah, M. (2018). Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488.

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.

Suzuki, T., Takahashi, Y., and Amano, Y. (2016). Precise uav position and attitude estimation by multiple gnss receivers for 3d mapping. In *Proceedings of the 29th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2016)*, pages 1455–1464.

Szymanowicz, S., Charles, J., and Cipolla, R. (2022a). Discrete neural representations for explainable anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 148–156.

Szymanowicz, S., Charles, J., and Cipolla, R. (2022b). Discrete neural representations for explainable anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 148–156.

Tan, M., Pang, R., and Le, Q. V. (2020a). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790.

Tan, M., Pang, R., and Le, Q. V. (2020b). EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10778–10787.

Tang, P., Wang, X., Wang, A., Yan, Y., Liu, W., Huang, J., and Yuille, A. (2018). Weakly supervised region proposal network and object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 352–368.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636.

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.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE.

Tsai, Y.-H., Hung, W.-C., Schulter, S., Sohn, K., Yang, M.-H., and Chandraker, M. (2018). Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7472–7481.

Tsouros, D. C., Bibi, S., and Sarigiannidis, P. G. (2019). A review on uav-based applications for precision agriculture. *Information*, **10**(11), 349.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, **104**(2), 154–171.

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 de Sande, K. E., Uijlings, J. R., Gevers, T., and Smeulders, A. W. (2011). Segmentation as selective search for object recognition. In *2011 international conference on computer vision*, pages 1879–1886. IEEE.

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, page 2.

Varga, L. A. and Zell, A. (2021). Tackling the background bias in sparse object detection via cropped windows. *arXiv preprint arXiv:2106.02288*.

Varga, L. A., Kiefer, B., Messmer, M., and Zell, A. (2021). Seadronessee: A maritime benchmark for detecting humans in open water. *arXiv preprint arXiv:2105.01922*.

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 (WACV)*, pages 2260–2270.

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 109–117.

VenJam1n (2022). Gtav mod - no chromatric aberration and lens distortion. https://de.gta5-mods.com/misc/no-chromatic-aberration-lens-distortion-1-41. Accessed: 2022-08-01.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., *et al.* (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, **2018**.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

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

Wang, Y., Wang, C., Zhang, H., Dong, Y., and Wei, S. (2019b). A sar dataset of ship detection for deep learning under complex backgrounds. *remote sensing*, **11**(7), 765.

Wei, Z., Yao, R., Kang, J., Chen, X., and Wu, H. (2022). Three-dimensional spectrum occupancy measurement using uav: Performance analysis and algorithm design. *IEEE Sensors Journal*, **22**(9), 9146–9157.

Wu, X., Li, W., Hong, D., Tao, R., and Du, Q. (2021). Deep learning for unmanned aerial vehicle-based object detection and tracking: A survey. *IEEE Geoscience and Remote Sensing Magazine*, **10**(1), 91–124.

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.

Wu et. al., Z. (2019). Delving into robust object detection from unmanned aerial vehicles: A deep nuisance disentanglement approach. In *Proceedings of the IEEE 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. (2019a). Clustered object detection in aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8311–8320.

Yang, X., Lin, D., Zhang, F., Song, T., and Jiang, T. (2019b). High accuracy active standoff target geolocation using uav platform. In *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, pages 1–4. IEEE.

Yang et. al., F. (2019). Clustered Object Detection in Aerial Images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8310–8319. Institute of Electrical and Electronics Engineers Inc.

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, W., Lu, Y., Easterbrook, S., and Fidler, S. (2020). Efficient and information-preserving future frame prediction and beyond. *arXiv preprint arXiv:1704.04899*.

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

Yue, X., Wu, B., Seshia, S. A., Keutzer, K., and Sangiovanni-Vincentelli, A. L. (2018). A lidar point cloud generator: from a virtual world to autonomous driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 458–464.

Zhang, H., Wang, Y., Dayoub, F., and Sunderhauf, N. (2021a). Varifocalnet: An iou-aware dense object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8514–8523.

Zhang, R. and Ding, J. (2012). Object tracking and detecting based on adaptive background subtraction. *Procedia Engineering*, **29**, 1351–1355.

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, T., Zhang, X., Li, J., Xu, X., Wang, B., Zhan, X., Xu, Y., Ke, X., Zeng, T., Su, H., *et al.* (2021b). Sar ship detection dataset (ssdd): Official release and comprehensive data analysis. *Remote Sensing*, **13**(18), 3690.

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.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022a). ByteTrack Github Repository. `https://github.com/ifzhang/ByteTrack`. Accessed: 2022-11-09.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022b). Bytetrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, pages 1–21. Springer.

Zhao, H., Zhang, H., and Zhao, Y. (2023). Yolov7-sea: Object detection of maritime uav images based on improved yolov7. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*.

Zhao, X., Pu, F., Wang, Z., Chen, H., and Xu, Z. (2019a). Detection, tracking, and geolocation of moving vehicle from uav using monocular camera. *IEEE Access*, **7**, 101160–101170.

Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H., and Hua, X.-S. (2017). Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1933–1941.

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

Zhong, Z., Sun, L., and Huo, Q. (2019). An anchor-free region proposal network for faster r-cnn-based text detection approaches. *International Journal on Document Analysis and Recognition (IJDAR)*, **22**(3), 315–327.

Zhou, H., Yuan, Y., and Shi, C. (2009). Object tracking using sift features and mean shift. *Computer vision and image understanding*, **113**(3), 345–352.

Zhou, J. T., Du, J., Zhu, H., Peng, X., Liu, Y., and Goh, R. S. M. (2019a). Anomalynet: An anomaly detection network for video surveillance. *IEEE Transactions on Information Forensics and Security*, **14**(10), 2537–2550.

Zhou, X., Wang, D., and Krähenbühl, P. (2019b). Objects as points. *arXiv preprint arXiv:1904.07850*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017a). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Zhu, P., Wen, L., Du, D., Bian, X., Ling, H., Hu, Q., Nie, Q., Cheng, H., Liu, C., Liu, X., *et al.* (2018a). 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., Bian, X., Ling, H., and Hu, Q. (2018b). Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437*.

Zhu, P., Du, D., Wen, L., Bian, X., Ling, H., Hu, Q., Peng, T., Zheng, J., Wang, X., Zhang, Y., *et al.* (2019). Visdrone-vid2019: The vision meets drone object detection in video challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision 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., Xiong, Y., Dai, J., Yuan, L., and Wei, Y. (2017b). Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2349–2358.

Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. (2017c). Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417.

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*.

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.

Zhuang, X. and Haralick, R. M. (1986). Morphological structuring element decomposition. *Computer vision, graphics, and image processing*, **35**(3), 370–382.

Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31. IEEE.

Zou, Y., Yu, Z., Kumar, B., and Wang, J. (2018). Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305.