POZNAN UNIVERSITY OF TECHNOLOGY

A Multi-Module Object Detection System Based on Deep Neural Networks

by

Aleksandra Kos

in the
Institute of Robotics and Machine Intelligence
Faculty of Control, Robotics, and Electrical Engineering

Supervisor: Dominik Belter, Ph.D., D.Sc. (Eng.)
Auxiliary Supervisor: Karol Majek, Ph.D. (Eng.)

2025

Abstract

This dissertation focuses on the problem of tiny object detection in high-resolution images, with applications in real-time systems such as autonomous robots and UAVs. Existing methods often require significant computational resources, are too slow or achieve poor detection quality, when objects are very small. To address these limitations, a modular framework (SegTrackDetect) is developed. It combines a lightweight local detector, that operates on selected full resolution areas of the input image, with three supporting modules: ROI Estimation based on semantic segmentation, ROI Prediction using object tracking, and Global Filtering with two novel algorithms to reduce redundant partial detections. Together, these components allow selective high-resolution inference on relevant image regions, improving speed, detection quality, and computational efficiency.

The first component, the ROI Estimation Module, uses semantic segmentation to guide the placement of detection windows. By aligning window selection with meaningful image content, this approach reduces redundant computation while retaining important contextual information, offering a more efficient alternative to uniform sliding-window techniques. The second component, the ROI Prediction Module, integrates object tracking to recover regions that may be missed by the Estimator because of its low input resolution. By exploiting temporal continuity, it ensures that small and difficult-to-detect objects remain visible to the system, while at the same time lowering the computational cost of the segmentation branch. The third component, the Global Filtering Module, addresses one of the core limitations of window-based methods: partial detections in overlapping regions of the detection windows. It introduces two novel algorithms – Overlapping Box Suppression (OBS), which favors complete detections, and Overlapping Box Merging (OBM), which combines multiple partial detections into a full one. Together, they provide a strong alternative to standard Non-Maximum Suppression and significantly improve quality by lowering both false positive and false negative detections.

Through the integration of these three modules, SegTrackDetect enables selective full-resolution inference on relevant subregions of high-resolution images. This modular design allows balancing detection quality, computational efficiency, outperforming traditional sliding-window and several state-of-the-art approaches while remaining lightweight enough for real-time deployment. The dissertation demonstrates that segmentation, tracking, and advanced filtering strategies can be combined into customizable framework for efficient tiny object detection and provides an open-source implementation for real-world applications.

Streszczenie

Niniejsza rozprawa dotyczy problemu detekcji bardzo małych obiektów na obrazach wysokiej rozdzielczości, ze szczególnym uwzględnieniem zastosowań wymagających działania w czasie rzeczywistym na robotach autonomicznych czy dronach. Istniejące metody często wymagają znacznych zasobów obliczeniowych, działają zbyt wolno lub charakteryzują się niską jakością detekcji, zwłaszcza gdy obiekty są bardzo małe. Aby przezwyciężyć te ograniczenia opracowano wielomodułowy system detekcji SegTrackDetect. System ten łączy lekki lokalny detektor, działający na wybranych obszarach obrazu w pełnej rozdzielczości, z trzema modułami wspierającymi: Estymacją ROI opartą na segmentacji semantycznej, Predykcją ROI wykorzystującą śledzenie obiektów oraz Filtrowaniem Globalnym, które obejmuje dwa nowe algorytmy redukujące problem częściowych detekcji. Połączenie tych modułów umożliwia selektywną analizę istotnych obszarów obrazu w wysokiej rozdzielczości, poprawiając czas działania, jakość detekcji i efektywność obliczeniową.

Pierwszy moduł, Estymacja ROI, wykorzystuje segmentację semantyczną do wyznaczania położenia okien detekcji. Dopasowując obszary do istotnych fragmentów obrazu, zmniejsza on nadmiarowe obliczenia, jednocześnie zachowując kontekst, co stanowi bardziej efektywną alternatywę dla prostych metod wykorzystujących okno przesuwne. Drugi moduł, Predykcja ROI, integruje śledzenie obiektów w celu przywrócenia regionów pominiętych przez estymator z powodu jego niskiej rozdzielczości wejściowej. Przez wykorzystanie ciągłości danych wejściowych, zapewnia on widoczność bardzo małych i trudnych do wykrycia obiektów, przy jednoczesnym obniżeniu kosztów obliczeniowych segmentacji. Trzeci moduł, Filtrowanie Globalne, rozwiązuje jedno z głównych ograniczeń metod opartych na oknach: częściowe detekcje we wspólnych obszarach okien detekcji. Wprowadza on dwa nowe algorytmy: Overlapping Box Suppression (OBS), który zachowuje pełne detekcje, oraz Overlapping Box Merging (OBM), który łączy fragmentaryczne wyniki w jedną spójną detekcję. Razem stanowią one alternatywę dla klasycznego Non-Maximum Suppression, poprawiając jakość poprzez ograniczenie zarówno fałszywie pozytywnych częściowych detekcji, jak i pominiętych obiektów.

Integracja tych trzech modułów sprawia, że SegTrackDetect umożliwia selektywną analizę w pełnej rozdzielczości tylko na istotnych podobszarach obrazu. Modularna konstrukcja pozwala zachować równowagę między jakością detekcji a efektywnością obliczeniową, przewyższając tradycyjne metody sliding-window oraz szereg najnowocześniejszych metod z literatury, a jednocześnie pozostając wystarczająco lekką do zastosowań w czasie rzeczywistym. Wykazano, że poprzez połączenie segmentacji, śledzenia obiektów oraz zaawansowanych strategii filtrowania uzyskano elastyczną i wydajną metodę detekcji bardzo małych obiektów, która została publicznie udostępniona umożliwiając zastosowanie w rzeczywistych problemach zarówno naukowych jak i wdrożeniowych.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dominik Belter, for his constant scientific guidance and support over the past four years. From shaping the research topic, helping me navigate academic challenges, to co-authoring publications and providing constant advice, his support has been essential for completing this dissertation. I am equally grateful to my co-supervisor, Karol Majek, for encouraging me to begin this doctoral journey and for his unwavering support at every stage, regardless of time or circumstance. His help went beyond academic guidance, allowing me to combine research with professional work, and his mentorship has been a continuous source of inspiration and motivation.

Finally, I would like to thank my family. To my mother, for nurturing curiosity and a love of learning from an early age, and to Jakub, for his patience, encouragement, and support during the most intense periods of this work. Their care and belief in me have been invaluable throughout this process.

I would also like to acknowledge the financial support of the Ministry of Education and Science, which made this Industrial PhD possible. This research was funded under the "Doktorat Wdrożeniowy" program (grant number DWD/5/0203/2021), and I am grateful for the support that enabled the successful completion of this work.

Abbreviations

AP	Average Precision
AR	Average Recall
BDOT10k	Baza Danych Obiektów Topograficznych
CIoU	Complete Intersection over Union
CLI	Command Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DETR	Detection Transformer
DIoU	Distance Intersection over Union
DotD	Dot Distance
FAF	False Alarms per Frame
FNA	False Negative Associations
FP	False Positive
FPA	False Positive Associations
FPS	Frames per Second
FPN	Feature Pyramid Network
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GIoU	Generalized Intersection over Union
GT	Ground Truth
HBB	Horizontal Bounding Box
HD	High Definition
HOTA	Higher Order Tracking Accuracy
IBS	Incomplete Box Suppression
IDFP	Identity False Positive
IDFN	Identity False Negative
IDP	Identification Precision
IDR	Identification Recall
IDSW	Identity Switches
IDTP	Identity True Positive
IoU	Intersection over Union
JSON	JavaScript Object Notation
MAV	Micro Aerial Vehicle

ML	Mostly Lost
MOT	Multiple Object Tracking
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
MT	Mostly Tracked
MTSD	Mapillary Traffic Sign Dataset
MS COCO	Microsoft Common Objects in Context Dataset
NMS	Non-Maximum Suppression
NWD	Normalized Wasserstein Distance
OBS	Overlapping Box Suppression
OBB	Oriented Bounding Box
OBM	Overlapping Box Merging
Pascal VOC	Pascal Visual Object Classes Dataset
P	Precision
PFD	Personal Flotation Device
PT	Partially Tracked
R	Recall
R-CNN	Region-based Convolutional Neural Network
ResNet	Residual Network
RFD	Receptive Field Distance
RNN	Recurrent Neural Network
ROI	Region of Interest
SAHI	Slicing Aided Hyper Inference
SAM	Segment Anything Model
SD	Single Detection
SOT	Single Object Tracking
SORT	Simple Online and Realtime Tracking
SSD	Single Shot MultiBox Detector
STT-IoU	Spatio-Temporal Tube Intersection over Union
SW	Sliding-Window
TP	True Positive
TPA	True Positive Associations
UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once

Contents

Chapter 1

Introduction

1.1 Motivation

Object detection is a fundamental computer vision task with wide-ranging applications, including autonomous driving, industrial inspection, and robotic perception. Over the past decade, deep learning has significantly advanced the field, enabling models to detect and classify objects with high accuracy using features learned from large-scale datasets. These advances have been largely driven by powerful convolutional architectures and benchmark challenges such as the MS COCO dataset [81], which evaluates performance across objects of varying sizes.

In robotics, object detection is essential for scene understanding, navigation, and human-robot interaction. Mobile and aerial robots, in particular, often rely on onboard perception systems to make real-time decisions in dynamic environments. Unlike offline image processing, robotic systems impose strict constraints on latency, memory usage, and computational resources – especially when deployed on embedded platforms or battery-powered UAVs. As a result, detection models in robotics must strike a balance between accuracy and efficiency. The work presented in this dissertation is directly motivated by an industrial problem: real-time detection of wind turbine defects. This challenge is being addressed in an ongoing applied research project within the Cufix company, where the system proposed in this work is being developed and deployed. Although the work is driven by the wind turbine inspection problem, public benchmarks are used for the evaluation of the proposed method to compare the system against established baselines.

Standard object detection models [12, 62, 136] are typically evaluated using the COCO benchmark [81], which reports performance in terms of Average Precision (AP) and Average Recall (AR) across three object scales: small, medium, and large. Even for the latest state-of-the-art methods, a significant gap remains between the detection performance on small and medium objects. This discrepancy stems mainly from the limited number of pixels that represent small objects. In COCO, small objects are defined as those with an area below 32^2 pixels. However, this definition does not adequately reflect the scale of objects in more challenging settings such as UAV or vehicle-mounted detection systems, where objects of interest may occupy just a few dozen pixels in images that exceed full HD resolution. Tiny object detection becomes especially

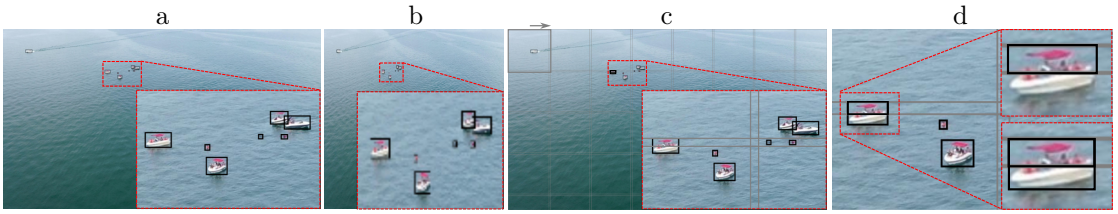


FIGURE 1.1: Challenges posed by tiny object detection in high-resolution images: (a) inference at the original resolution preserves object features but is resource-intensive; (b) downsampling enables fast and lightweight inference but greatly reduces spatial information; (c) naive sliding-window preserves features but is time-consuming and causes object fragmentation; (d) standard Non-Maximum Suppression struggles in window-based settings – depending on the confidence and IoU thresholds, it may retain fragmented detections if their score is higher than the full detection (top row) or keep both detections if their IoU is low (bottom row).

critical in such robotic scenarios. For example, drones may need to detect distant vehicles or people, and ground robots may need to identify small tools. These small objects often have a significant impact on the task at hand. However, detecting tiny objects remains challenging because their visual features are easily lost during image downsampling. Even advanced detectors struggle with low-resolution or heavily occluded instances. In addition, tiny object detection is sensitive to factors such as receptive field size, anchor box design, and spatial resolution of backbone feature maps.

In these higher-resolution domains, often starting at 1280×720 and frequently reaching or exceeding 1920×1080, standard object detectors struggle. Inference running on full-resolution images becomes computationally expensive and slow, which is not suitable for mobile robotics and real-time systems (Fig. 1.1a). Although input downsampling can reduce processing time (Fig. 1.1b), it often causes tiny objects to vanish entirely from the feature maps, leading to missed detections. A simple workaround is a naive sliding-window approach, which applies a fixed-size detector across overlapping patches of the original image. While this improves recall for tiny objects, it greatly increases runtime and causes the fragmentation of larger objects (Fig. 1.1c). To overcome this trade-off, a class of window-based object detection methods emerged, designed to identify promising regions for high-resolution inference. These typically rely on initial region selection mechanisms, such as clustering [157], density maps [34, 75], or preliminary detections [71, 169], to estimate the likely object-rich areas. However, most of these methods are optimized for dense scenes and are not well suited for detecting sparsely distributed objects.

Many window-based detection strategies attempt to maintain performance on large objects by combining full-image and tile-based predictions [75, 96, 142, 155, 157, 169]. However, training with mixed-resolution inputs can lead to inconsistencies in object scale, which may reduce overall detection quality. To mitigate such issues, some systems dynamically adapt image resolution based on object scale estimates, for example, by upsampling the entire image [178] or selectively enhancing relevant tiles [31]. Others focus on the post-processing stage, introducing methods like Incomplete Box Suppression (IBS) [71] to address detection fragmentation across tiles.

While numerous approaches exist for object detection in high-resolution imagery, most suffer from one of two key limitations: high computational cost due to dense sliding-window operations, or diminished performance due to aggressive downsampling. Moreover, few existing

solutions are designed to support both dense and sparse scenes or to operate effectively in real-time video applications. This thesis addresses these gaps by proposing an efficient modular system for real-time tiny object detection in high-resolution video data. The system combines segmentation-based and tracking-based ROI selection with novel post-processing algorithms to improve detection performance while reducing computational cost. This modular design allows it to adapt to both dense and sparse object layouts and makes it particularly well-suited for robotics applications.

1.2 Problem Statement

Tiny object detection in high-resolution images presents a unique set of challenges that are not sufficiently addressed by conventional object detection pipelines. Standard methods typically operate on downsampled images, which leads to the loss of critical fine-grained features necessary for detecting small objects. This problem is particularly evident in high-resolution domains such as UAV-based monitoring, surveillance, and mobile robotics, where objects of interest often occupy only a few pixels and appear at multiple scales. Sliding-window approaches offer a brute-force alternative by exhaustively scanning high-resolution inputs with overlapping tiles. While effective in improving recall for small objects, this strategy is computationally expensive and impractical for real-time applications. To reduce computational cost, a class of window-based detection systems has been proposed that aims to infer promising subregion ROIs, where full-resolution detection should be performed. However, many of these systems rely on clustering or density estimation, which limits their applicability to crowded scenes and compromises generalization to sparse layouts. Furthermore, existing methods often neglect the temporal structure of video data, treating each frame independently. This disregards valuable motion cues that could help recover missed detections, especially in low-contrast or occluded settings. Post-processing strategies such as Non-Maximum Suppression (NMS) also fall short in window-based systems, where overlapping tiles often produce redundant or fragmented detections that may not be properly filtered, as illustrated in Fig. 1.1d.

To address these limitations, this thesis proposes a modular object detection system that integrates: a segmentation-based `ROI Estimator` trained directly on masks extracted from detection labels, a lightweight `ROI Predictor` based on object tracking, and novel post-processing algorithms designed to operate globally across detection windows. Together, these components aim to improve detection accuracy, inference speed, and robustness across dense and sparse scenarios. The system is particularly well-suited for high-resolution video streams in computationally constrained environments, such as autonomous or embedded robotic platforms. Building on this motivation, the main research hypothesis investigated in this dissertation is as follows:

Selective full-resolution inference on image subregions identified by Region of Interest Estimator and Object Tracker improves the tiny object detection performance on high-resolution images, measured by Average Precision (AP), Average Recall (AR), and inference speed.

This main hypothesis is supported by three auxiliary theses, each corresponding to one of the key components of the system:

1. **ROI Estimation:** Estimating ROIs using a deep neural network model enhances both detection quality and inference speed compared to the naive sliding-window approach.

2. **ROI Prediction:** Incorporating an additional ROI source, such as an object tracker, into the ROI-based object detection system further improves detection quality, inference speed, and computational efficiency over state-of-the-art tiny object detection methods.

3. **Global Filtering:** False positive partial detections in window-based systems can be reduced through post-processing techniques such as `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`.

1.3 Proposed Method

To address the challenges of object detection in high-resolution images, this thesis proposes a modular detection system consisting of two complementary ROI selection stages. The first stage, the `ROI Estimator`, performs binary segmentation on low-resolution inputs to identify candidate regions. The `Estimator` is trained directly on masks generated from the detection labels, avoiding reliance on density or clustering assumptions and improving generalization to sparse and multi-scale scenarios. The second stage, the `ROI Predictor`, is implemented as an object tracker that leverages temporal information in video sequences to complement the ROI selection, enhancing recall over time, particularly for objects missed in individual segmentation masks. These two modules guide a lightweight, general-purpose object detector to focus exclusively on relevant high-resolution image regions likely containing objects. Detection windows are generated from ROI masks by the `Detection Window Proposal Block`, which applies multiple processing steps to effectively handle ROIs of varying sizes and shapes. The system is designed to robustly detect both densely clustered tiny objects and sparsely distributed objects across multiple scales. This robustness comes from two key mechanisms. First, the object tracker improves recall for tiny objects by compensating for frames where the segmentation-based `ROI Estimator` misses regions of interest. Second, the `Detection Window Proposal Block` processes large ROI regions using a hybrid approach that combines sliding-windows and crop-and-resize techniques. This strategy preserves fine details critical for tiny object detection while maintaining sufficient context for larger objects, which are otherwise prone to fragmentation by fixed detection windows. To further improve detection quality, two complementary algorithms are introduced within the `Global Filtering Block`. The `Overlapping Box Suppression (OBS)` algorithm filters out fragmentary false positive detections caused by overlapping detection windows and partial views of objects. Conversely, the `Overlapping Box Merging (OBM)` algorithm merges fragmented detections into complete ones when no full detection is present. This global filtering approach outperforms standard Non-Maximum Suppression (NMS) in window-based detection scenarios.

The modular design of the system allows for straightforward customization and independent updating of each component as advances are made in their respective areas. The method overcomes the low detection quality commonly found in traditional single-shot detectors that downsample high-resolution inputs, while offering significant computational savings compared to naive sliding-window approaches. The `ROI Estimator`, relying on binary semantic segmentation at relatively

large input resolutions, provides high-quality ROI proposals. However, in resource-constrained environments, this can be computationally expensive. Therefore, the `ROI Predictor`, based on object tracking, enables a reduction in segmentation input size and overall computational complexity, accelerating inference while maintaining detection quality. Throughout this thesis, it is demonstrated that this dual-ROI approach achieves superior detection performance compared to several state-of-the-art methods, with fewer trainable parameters. The `Detection Window Proposal Module` further accelerates processing by filtering initial detection windows, and the combination of multi-scale ROI handling with the `Global Filtering Block` ensures robust detection across object sizes.

1.4 Contributions

The novelty of the proposed system lies in its modular design that uniquely combines segmentation-based `ROI Estimation` with a lightweight tracking-based `ROI Prediction`, enabling efficient and accurate object detection in high-resolution images and video streams. Unlike many existing window-based methods that prioritize accuracy at the expense of runtime efficiency [75, 155, 157], this system emphasizes real-time processing suitable for resource-constrained environments. Its segmentation module learns ROI masks directly from detection labels, avoiding complex intermediate representations like density maps or clusters, as used in prior works [34, 71, 75], thus ensuring robust performance across both dense and sparse object distributions. The integration of a tracking-based `ROI Predictor` significantly reduces computational load by allowing lower-resolution segmentation inputs without sacrificing detection quality [69]. Furthermore, the system's dedicated handling of large ROIs balances the preservation of fine details for tiny objects with the necessary context for larger or multi-scale objects [132]. Complementing these advances are novel postprocessing algorithms, namely `Overlapping Box Suppression` and `Overlapping Box Merging`, specifically designed to address challenges in window-based detection by filtering redundant partial detections and merging fragmented ones, outperforming traditional NMS approaches. Together, these innovations deliver a flexible and computationally efficient detection framework with open-source implementation [70], making it highly suitable for practical applications such as robotics and embedded systems.

The main contributions of this doctoral dissertation can be summarized as follows:

- a modular, window-based tiny object detection system that combines lightweight local detectors operating on small sub-windows with an efficient ROI selection process, supporting real-time, high-quality detection of both tiny and multi-scale objects in high-resolution images,
- a data-driven `ROI Estimation Module` that leverages the low-resolution current frame to estimate regions of interest for fine-grained detection. Ground-truth masks are generated directly from the detection labels, enabling support for both dense and sparse object distributions, in contrast to density- or clustering-based ROI selection methods that favor high-density regions,

- a lightweight, non-trainable `ROI Prediction Module` that exploits the temporal consistency of real-time data streams using tracking. This module supports ROI selection for the most difficult-to-detect occluded and tiny objects, allowing recovery of regions missed by the `Estimator` operating on highly downsampled input,
- support for both tiny-only and multi-scale scenarios through a dual ROI handling strategy, which combines sliding-window detection within large ROIs with a crop-and-resize approach to maintain the integrity of larger instances while preserving fine-grained representation of smaller objects,
- a dedicated `Global Filtering Block` for window-based systems, integrating the novel `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)` algorithms, designed to merge and filter fragmented detections commonly encountered in focus-and-detect systems,
- a highly customizable, open-source framework for window-based detection.

1.5 Thesis Organization

To guide the reader through the structure and contributions of this work, the dissertation is organized into eight chapters. Each chapter addresses a distinct aspect of the research, beginning with the motivation and literature review, followed by a description of the proposed system and detailed experimental evaluation of the most important modules. The core of the thesis is structured around three main components of the system: `ROI Estimation` (Chapter 4), `ROI Prediction` (Chapter 5), and the `Global Filtering Block` (Chapter 6). This work does not include a standalone "Results" chapter. Instead, experimental results and ablation studies are presented within the individual chapters that introduce the corresponding methods, closely following the structure of the related publications. The following overview briefly summarizes the content and purpose of each chapter:

1. **Introduction** - This chapter introduces the problem of tiny object detection in high-resolution images, with a particular focus on its relevance to mobile robotics applications. It explains the challenges associated with detecting small objects, such as scale variance and limited visual cues, and justifies the need for specialized window-based approaches. The chapter outlines the research motivation, problem statement, and objectives of the dissertation. It also briefly presents the proposed method, situating it within the broader context of general object detectors and other window-based techniques. Finally, it states the main research theses and summarizes the key contributions of the work.
2. **Related Work** - This chapter reviews the existing literature relevant to the three core components of the proposed system: object detection, Region of Interest (ROI) selection, and object tracking. It begins with an overview of commonly used object detection metrics, including those specific to small and tiny object detection, and introduces the evaluation protocol applied throughout the thesis. Next, several high-resolution image and video datasets are compared, providing justification for the benchmark selection used in this

work. The chapter then offers an in-depth analysis of tiny object detection, emphasizing the specific challenges involved, particularly in real-time, high-resolution scenarios. Various method categories are introduced and systematically compared, with the proposed system positioned within the "Focus-and-Detect" family of approaches - those that direct the detector's attention to selected image regions likely to contain objects. The subsequent section on Region Generation explores this focus mechanism in more detail, laying the groundwork for understanding the `ROI Estimation` component of the system discussed in Chapter 4. Since the proposed method incorporates not only image-based cues but also temporal information, the chapter concludes with a literature review on object tracking, emphasizing its role in refining ROI selection and justifying its integration into the system. The effectiveness of the tracker-based `ROI Prediction` is further analyzed experimentally in Chapter 5.

3. **Proposed Tiny Object Detection System** - Provides a high-level overview of the SegTrackDetect framework, introducing its key components and their interactions. The main elements include segmentation-based `ROI Estimation`, `ROI Prediction` using object tracking, a `Fusion Module` that generates the final ROI mask, the `Detection Windows Proposal Module`, the `Local Object Detection Module`, and the `Global Filtering Block`. This chapter serves as an overview of the complete system pipeline, which is explored in more detail in the following three chapters, and also discusses the datasets used as well as the core contributions and novelty of the proposed system.

4. **ROI Estimation-based Tiny Object Detection System** - This chapter describes the segmentation-based detection pipeline, covering both the initial TinyROI prototype and its evolved estimation-only configuration within SegTrackDetect. It begins by presenting the key contributions introduced by the ROI estimation-based approach, followed by a detailed discussion of the relevant system components, with emphasis on the differences between the TinyROI prototype and the advanced SegTrackDetect architecture. The chapter focuses on Auxiliary Thesis #1: "Estimating ROIs using a deep neural network model enhances both detection quality and inference speed compared to the naive sliding-window approach". To support this, the SegTrackDetect variant is benchmarked against TinyROI and the sliding-window methods across several challenging datasets. Then, the TinyROI system is evaluated against single-shot detection with downsampling and sliding-window method. Finally, the ablation study provides justification for various architectural decisions, including the ROI estimator training pipeline, the `Detection Windows Proposal Module`, and the necessity of a dedicated `Global Filtering Block`.

5. **ROI Estimation and Prediction-based Tiny Object Detection System** - This chapter extends the previous design by incorporating object tracking for `ROI Prediction` into the ROI selection module. It begins with an overview of the main contributions and the proposed method, highlighting the key modifications compared to the SegTrackDetect variant described in Chapter 4, specifically the addition of tracking and `ROI Fusion`. Experimental results demonstrate that the fusion-based system outperforms several state-of-the-art methods (covering general, video, and tiny object detection) on two challenging datasets. The ablation study aims to validate Auxiliary Thesis #2, showing that integrating `ROI Prediction` reduces computational load, accelerates inference, and improves

detection quality beyond existing state-of-the-art approaches. By enabling a lower input resolution for the segmentation-based `Estimator`, the `ROI Prediction Module` reduces computational complexity and increases processing speed, while maintaining detection quality by recovering regions containing tiny objects missed at low resolution.

6. **Detection Filtering Methods** - This chapter introduces two novel algorithms developed to enhance post-processing of window-based object detection outputs: `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`. The chapter begins by motivating the need for improved filtering techniques tailored to the unique challenges of window-based detection systems, where overlapping detection windows often produce redundant or partial detections, particularly for larger objects. A detailed description of the `OBS` algorithm follows, explaining how it selectively suppresses partial detections while preserving full correct detections, thereby reducing false positives and improving precision. Next, the `OBM` method is presented as a complementary approach designed to merge fragmentary detections that together represent a single object but are detected separately across overlapping windows. The chapter compares the performance of these methods against the widely used Non-Maximum Suppression (NMS) baseline, demonstrating the superior ability of `OBS` and `OBM` to handle detection fragmentation and improve overall detection accuracy. Experimental evaluations on the multi-scale SeaDronesSee dataset illustrate how these filtering strategies contribute to the robustness of the proposed system. This chapter addresses Auxiliary Thesis #3 by demonstrating that specialized post-processing methods, such as `OBS` and `OBM`, can further enhance detection quality in window-based object detection systems.

7. **Implementation Details** - This chapter focuses on the practical aspects of the dissertation and the conducted research. It begins with a detailed discussion of the open-source, highly customizable SegTrackDetect framework, covering its core functionalities and design. The chapter then addresses the integration of the system on embedded devices, presenting a discussion of the trade-offs between detection quality and processing speed when deploying the system in resource-constrained environments. This section is crucial for demonstrating the system's applicability in real-world mobile robotics scenarios, emphasizing its practical relevance. Due to company confidentiality, details regarding the embedded system implementation remain proprietary.

8. **Conclusions** - This chapter summarizes the main findings of the dissertation with a focus on how the research theses were validated through the conducted studies and experiments. It highlights the key contributions made, emphasizing their significance within the context of tiny object detection in high-resolution images. The chapter also critically reflects on the strengths and limitations of the proposed methods and system design. Finally, it outlines promising directions for future research to further improve detection accuracy, efficiency, and applicability in real-world scenarios.

The research presented in this dissertation was supported by the Ministry of Education and Science under the "Doktorat Wdrożeniowy" program (grant number DWD/5/0203/2021). The author gratefully acknowledges this financial support, which was essential for the successful completion of this work.

1.6 Publications

The research carried out throughout this doctoral project has led to several scientific publications. Some of these publications emerged directly from the core topic of the dissertation, and the results they present are used and discussed in detail within the thesis. Others address related problems and were developed in parallel as part of a broader research effort in object detection and computer vision. While these latter works concern similar application areas, their specific results are not included in the dissertation. Together, these contributions have been published in peer-reviewed journals and international conference proceedings and reflect the iterative nature of the research process. The following lists include all publications that emerged during the course of the dissertation.

Journal Articles

1. Kos, A., Belter, D., and Majek, K. "Deep Learning for Small and Tiny Object Detection: A Survey." *Pomiary Automatyka Robotyka*, vol. 27, no. 3, 2023. MNiSW: 100.

   Summary: This paper presents a structured overview of deep learning methods for small and tiny object detection, covering relevant datasets, evaluation metrics, and the key challenges specific to detecting small objects in high-resolution images. It also distinguishes between relatively small objects in high-resolution contexts and absolutely small objects in terms of pixel count. The content is partially used in Chapter 2.

2. Kos, A., Belter, D., and Majek, K. "Enhanced Lightweight Detection of Small and Tiny Objects in High-Resolution Images Using Object Tracking-Based Region of Interest Proposal." *Engineering Applications of Artificial Intelligence*, vol. 153, 2025, 110852. IF: 7.5, MNiSW: 140.

   Summary: This paper presents a Region of Interest (ROI) proposal method that combines `ROI Estimation` with a tracking-based `ROI Prediction` component to enhance the detection of small objects. It demonstrates improved performance over systems that rely solely on estimation and is validated on real-world datasets against several state-of-the-art detection methods. The content is contextually aligned with Chapter 5.

3. Kos, A., Belter, D., and Majek, K. "SegTrackDetect: A Window-Based Framework for Tiny Object Detection via Semantic Segmentation and Tracking." *SoftwareX*, vol. 30, 2025, 102110. IF: 2.4, MNiSW: 200.

   Summary: The introduces an open-source, real-time framework for window-based tiny object detection. The system is used throughout this dissertation to implement both image- and video-based detection pipelines described in Chapters 4, 5, and 6. In subsequent chapters, the term *SegTrackDetect* refers to the final version of the detection system, including both estimation-only and fused configurations.

4. Kos, A. "Overlapping Box Suppression and Merging Algorithms for Window-Based Object Detection." *Foundations of Computing and Decision Sciences*, vol. 50(3), pp. 403-423, 2025. IF: 1.8, MNiSW: 40.

Summary: This paper introduces two algorithms, `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`, designed to improve post-processing in window-based object detection systems. Both methods address limitations of traditional Non-Maximum Suppression (NMS) by filtering out redundant or fragmentary detections across overlapping windows. The algorithms are evaluated on high-resolution datasets and are directly analyzed in Chapter 6.

Conference Proceedings

1. Kos, A., Majek, K., and Belter, D. "Where to Look for Tiny Objects? ROI Prediction for Tiny Object Detection in High-Resolution Images." In: *Proceedings of the 17th International Conference on Control, Automation, Robotics and Vision (ICARCV 2022)*, IEEE, 2022. MNiSW: 140.

   Summary: Presents the initial estimation-based system, referred to as TinyROI, evaluated on the Mapillary Traffic Sign Dataset. This system serves as the estimation-only baseline and later evolves into the SegTrackDetect estimation-only configuration, which is further analyzed in Chapter 4.

2. Kos, A., and Majek, K. "CNN-Based Traffic Sign Detection on Embedded Devices." In: *Proceedings of the 3rd Polish Conference on Artificial Intelligence (PP-RAI 2022)*, Gdynia, Poland, 25-27 April 2022, pp. 108-111. Uniwersytet Morski w Gdyni, 2022. MNiSW: 20.

   Summary: Presents a lightweight convolutional detection system for traffic signs optimized for embedded deployment. While not directly used in the dissertation, this work inspired the thesis topic and highlighted the limitations of standard detection methods when applied to small objects in high-resolution images with complex backgrounds.

3. Kos, A., and Majek, K. "BDOT10k-seg: A Dataset for Semantic Segmentation." In: Wojciechowski, A., Lipiński, P. (Eds.), *Progress in Polish Artificial Intelligence Research 4*, Monografie Politechniki Łódzkiej, no. 2437, Wydawnictwo Politechniki Łódzkiej, Łódź, 2023. MNiSW: 20.

   Summary: Describes the creation of an object detection and segmentation dataset derived from the BDOT10k database. Although not directly used in the thesis, the dataset is thematically aligned with this work and features extremely high-resolution images. It is included in the overview of the datasets in Chapter 2.

4. Kos, A. "Overlapping Box Suppression Algorithm for Window-Based Object Detection." In: *Progress in Polish Artificial Intelligence Research 5*, Proceedings of the 5th Polish Conference on Artificial Intelligence (PP-RAI 2024), Warsaw, Poland, 18-20 April 2024, pp. 325-330. MNiSW: 20.

   Summary: Introduces the initial version of the `Overlapping Box Suppression (OBS)` algorithm for filtering redundant detections in window-based object detection. This method is subsequently extended and comprehensively evaluated in Chapter 6.

5. Kos, A., Majek, K., and Belter, D. "Dataset Augmentation for Detecting Small Objects in Fisheye Road Images." In: *Lecture Notes in Artificial Intelligence*, Proceedings of the 24th

International Conference on Artificial Intelligence and Soft Computing (ICAISC 2025), Zakopane, 2025. MNiSW: 20. (in press)

Summary: Introduces a data augmentation pipeline aimed at improving the detection of small objects in distorted fisheye images from the FishEye8K dataset. While addressing related challenges, such as small object distortion in non-high-resolution images, this work is not part of the main dissertation.

Chapter 2

Related Work

2.1 Introduction

Recent advances in computer vision, driven by deep learning and the availability of large annotated datasets, have significantly improved performance across tasks such as image classification, segmentation, and object detection. Within object detection, considerable progress has been made on detecting medium and large objects, but tiny object detection remains an open challenge due to factors like low-resolution objects, complex backgrounds, and limited contextual information. To address these challenges, many existing approaches follow a focus-and-detect strategy, where specific regions of an image are prioritized for detailed analysis. The system that is introduced in this work builds on this principle by integrating both segmentation and tracking to guide ROI selection, enabling accurate detection of small objects while maintaining computational efficiency. To position the proposed system within the context of existing literature, this chapter begins with the review of general object detection with a focus on the challenges and advancements in tiny object detection (Section 2.2). This includes a discussion of commonly used evaluation metrics (Section 2.3), datasets (Section 2.4), and detection methods (Section 2.5). Existing tiny object detection approaches are categorized into seven distinct groups with a detailed comparison among them. Given that the proposed system leverages temporal information for ROI selection, the literature review also includes video object detection methods (Section 2.6). Furthermore, methods for ROI selection (Section 2.7) and object tracking (Section 2.8) are discussed, as the proposed approach combines both spatial and temporal cues to guide the detection process.

2.2 Object Detection

Object detection is a fundamental task in computer vision that involves two key subtasks: accurately localizing objects within an image and correctly classifying them. Localization is typically represented by bounding box coordinates, either in the form of horizontal bounding boxes (HBB)

or oriented bounding boxes (OBB), depending on the application's requirements. In this dissertation, the focus is on HBB - however, the proposed method could be extended to OBB by using a different `Local Detector` and making minor algorithmic adjustments, while following the same general paradigm.

In recent years, deep learning, especially convolutional neural networks (CNNs), has led to major improvements in object detection. Popular detectors perform well on standard benchmarks. On the COCO dataset [81], these models reach Average Precision (AP) scores of around 60% for medium objects and 70% for large ones [140, 190]. However, detecting small objects is still a major challenge. Small objects often contain fewer pixels, lack strong features, and can be blurred or distorted by sensor noise. These factors make both classification and localization more difficult. Additionally, most modern detectors rely on downsampling in their backbone networks, and images are often scaled down during preprocessing, which can cause small objects to disappear entirely [82, 84]. Traditional anchor-based detection methods are also less effective when objects are very small [170]. The proposed solution employs shallow, general-purpose detectors at the local level, which helps preserve the spatial detail present in high-resolution imagery and reduces the negative effects of aggressive downsampling in deeper architectures.

Object detection methods are typically divided into two categories: two-stage and one-stage detectors. Most traditional detectors are anchor-based, but more recent approaches have introduced anchor-free designs. Two-stage methods, such as R-CNN [44], Fast R-CNN [43], and Faster R-CNN [110], first generate region proposals and then refine them through classification and bounding box regression. These models are generally accurate but slower, since they require an extra step to generate proposals. One-stage detectors, like YOLO [12, 107–109], RetinaNet [83], and SSD [85], skip the proposal step and predict object locations and classes in one pass based on the predefined set of anchors. This makes them faster, though often slightly less accurate. Recent anchor-free methods, such as CenterNet [180] and FCOS [128], eliminate the need for predefined anchor boxes entirely, simplifying the architecture and improving performance in some scenarios. To better handle objects of different sizes, Feature Pyramid Networks (FPN) [82] combine feature maps from multiple levels of the network to provide richer multi-scale representations. The modular system proposed in this work employs lightweight one-stage architectures for `Local Object Detection`, combined with an additional ROI selection step to avoid exhaustive analysis of background-only regions common in high-resolution imagery.

In general detection, some key challenges include large appearance variations within object classes (intra-class), similarities between different classes (inter-class), and noisy backgrounds. The COCO dataset [81], alongside Pascal VOC [36], is a common benchmark for evaluating detectors. As evaluation metrics, it includes APs and average precision (AP) for objects smaller than 32×32 pixels. The performance gap between AP_s and AP_l (objects larger than 96×96 pixels) can reach around 30 percentage points [136], even for the models with the best performance. This highlights the difficulty of detecting small objects. Small objects tend to appear at low resolution, are often blurred, and contain limited visual information. Additionally, downsampling operations in backbone networks reduce the visibility of small objects in deeper feature maps, further degrading detection performance. In datasets that focus on small or tiny objects, particularly those with high-resolution images, foreground objects may occupy only a tiny fraction of the image. This

low signal-to-noise ratio significantly increases the risk of false positives. To address this issue, several new datasets have been introduced, specifically targeting small-scale object detection. Examples include AI-TOD [138, 153] and SODA [24], both of which emphasize extremely small objects. Many of these datasets, especially those in the remote sensing domain [24, 32, 66], contain ultra-high-resolution images, further increasing the complexity of the detection task.

The following sections of this literature review are organized as follows: first, the evaluation metrics used in object detection are discussed, followed by an overview of relevant datasets. Each section begins with a general object detection perspective before addressing the specific challenges unique to tiny object detection, clearly distinguishing between the two contexts. Additionally, definitions of object size in the literature are reviewed, comparing relative and absolute size definitions, along with a detailed analysis of seven groups of methods developed specifically for tiny object detection. Given the focus on developing a real-time video-based tiny object detection system, video object detection methods are also briefly reviewed. These approaches leverage the sequential nature of video data to improve detection quality, rather than applying image-based detection methods independently to each frame.

2.3 Object Detection Metrics

Metrics are a fundamental pillar of benchmarking, not only in computer vision but across all engineering and scientific disciplines. They enable fair and consistent comparison between methods, and the unification of evaluation metrics is crucial for ensuring uniformity across studies. In object detection, the most commonly used metrics are true positive count (TP), false positive count (FP), false negative count (FN), precision, recall, as well as Average Precision (AP) and Average Recall (AR). Object detection involves two main tasks: localizing objects within an image and classifying them correctly. For a detection to be considered a true positive, it must both predict the correct class and significantly overlap with a corresponding ground-truth object. This overlap is commonly quantified using the Intersection over Union (IoU), defined for two rectangles - A (the detected bounding box) and B (the ground truth) as:

$$IoU = \frac{|A \cap B|}{|A \cup B|}. \tag{2.1}$$

Although IoU is widely adopted for evaluating detection performance, it has a major limitation in training neural networks: its value is zero for non-overlapping boxes. To address this issue, several extensions of IoU have been proposed:

- Generalized Intersection over Union (GIoU) [111], which introduces a third rectangle (C), defined as the smallest rectangle enclosing both A and B. It is computed as:

$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}. \tag{2.2}$$

- Distance Intersection over Union (DIoU) [177], designed to improve the convergence of the training process. It adds a penalty term representing the normalized Euclidean distance

between the centers of A and B:

$$DIoU = IoU - \frac{\rho^2(a,b)}{c^2}. \tag{2.3}$$

where a, b are the center points of A and B, c is the diagonal length of C, and $\rho(\cdot)$ denotes the Euclidean distance

- Complete Intersection over Union (CIoU) [177] which in addition to penalizing the distance between A and B, also emphasizes the similarity of the aspect ratio between A and B:

$$CIoU = IoU - \frac{\rho^2(a,b)}{c^2} - \frac{v^2}{(1-IoU)+v}, \tag{2.4}$$

where:

$$v = \frac{4}{\pi^2}(arctan\frac{W_A}{H_A} - arctan\frac{W_B}{H_B})^2, \tag{2.5}$$

and W_A,H_A,W_B,H_B are the widths and heights of rectangles A and B, respectively.

The extension of the IoU for video data is the Spatio-Temporal Tube Intersection over Union (STT-IoU) defined as the ratio of the intersection volume to the union volume of all merged detections and ground truth objects.

One of the fundamental sets of metrics in object detection includes precision (precision = $\frac{TP}{TP+FP}$), recall (recall = $\frac{TP}{TP+FN}$), and the precision-recall (PR) curve. However, these metrics alone do not assess localization accuracy. For a prediction to be considered a true positive (TP), two conditions must be met: the predicted class must be correct, and the Intersection over Union (IoU) between the predicted and ground-truth bounding boxes must exceed a predefined threshold. Predictions that meet the class condition but fail the IoU threshold are counted as false positives (FP), while false negatives (FN) are ground-truth objects not matched by any prediction. The IoU quantifies the spatial overlap between predicted and ground-truth bounding boxes. Older datasets [32, 33, 158] typically use metrics introduced by the Pascal VOC benchmark [36], whereas more recent datasets [24, 73, 132, 138, 184] more commonly adopt the evaluation protocol defined by MS-COCO [81]. Both Pascal VOC and MS-COCO rely on Average Precision (AP), defined as the area under the precision-recall curve. The main differences lie in the interpolation method and the IoU thresholding. Pascal VOC computes AP using 11-point interpolation at a single IoU threshold of 0.5. In contrast, MS-COCO uses 101-point interpolation and evaluates performance at multiple IoU thresholds: 0.5 ($AP_{0.5}$), 0.75 ($AP_{0.75}$), and the average over thresholds from 0.5 to 0.95 in steps of 0.05 (reported as AP). Additionally, COCO reports AP and Average Recall (AR) for objects of different sizes (small, medium, and large), as well as AR metrics based on the number of detections per image: AR_1, AR_{10}, and AR_{100}. To gain deeper insight into detector performance under varying conditions, Average Precision (AP) is often reported separately for each class [24, 32, 73, 138, 153, 158] or object size [24, 81, 138, 153, 158, 163]. In aerial imagery datasets, AP is also frequently broken down by UAV altitude or viewing angle [33, 132]. Average Recall (AR) is sometimes customized to reflect the characteristics of a specific dataset. For example, VisDrone and AI-TOD report AR_{500} and AR_{1500}, respectively, due to the high average number of objects per image. In the TinyPerson dataset, a reduced IoU threshold of 0.25 is adopted to emphasize object detection over precise

localization, acknowledging the difficulty of accurately localizing extremely small objects. The proposed solution is evaluated using modified COCO metrics based on the relative size thresholds described later in this section. Chapter 3 provides a detailed overview of the exact metrics used for each evaluation dataset.

While the same metrics are usually used to evaluate the quality of Tiny and Small Object Detection as for generic object detection [36, 81], in [153] the reduced effectiveness of the IoU in small object detection was shown. This is because the IoU is highly sensitive to position deviation when applied to small objects. The authors note the following implications - anchor-based detectors that use IoU to discriminate between positive and negative samples tend to classify an anchor as negative based on a slight location shift, and tiny objects get few positive samples compared to normal-scale ones, making the detector biased towards detecting large objects. The issues were addressed with the introduction of the Normalized Gaussian Wasserstein Distance (NWD), which can model the position relation even for non-overlapping rectangles. [152] authors argue that an IoU as well as its all variations (Generalized IoU, Distance IoU and Complete IoU) are unsuitable for tiny object detection, and thus propose a Dot Distance (DotD) metric. DotD is defined as the normalized Euclidean distance between the central points of two bounding boxes. In [154] Receptive Field Distance (RFD) was introduced. RFD measures the distance between the ground-truth bounding-boxes and the receptive fields of the feature points by modeling their Gaussian distributions. It can be applied for both anchor-based and anchor-free detectors. The Normalized Wasserstein Distance (NWD), that was proposed in [153], measures the similarity between two bounding-boxes using Gaussian distributions placed at their centers, with the middle pixel having the greatest weight.

A common approach to evaluating detection performance is to assess it across different object sizes, where the size of an object is typically defined in one of three ways: by its absolute area [24, 81], by the geometric mean of its height and width [138, 163] (referred to as object size in this work), or by a single dimension such as height [158]. For consistency, all thresholds in this work are expressed in terms of object size defined as the square root of the area; when datasets use alternative definitions, thresholds were converted accordingly. Many datasets follow the COCO convention, which reports Average Precision (AP) and Average Recall (AR) for three size categories: small, medium, and large [14, 35]. In some cases, performance is reported per class [32, 184]. There is no strict definition of a tiny object. However, based on definitions used in datasets like TinyPerson and AI-TOD, the tiny category is typically introduced to distinguish objects that fall within the lower end of the small category in COCO. An exception is SODA, which defines objects larger than 32 pixels as small and sets the upper limit for tiny objects at 32 pixels.

The COCO evaluation protocol is widely accepted as a standard for evaluating object detectors. However, within the small object detection literature, it is common to introduce additional, finer-grained size categories [24, 138] by subdividing the small category. As shown in Tab. 2.1, datasets dedicated to tiny object detection usually exclude large instances. For example, TinyPerson, AI-TOD, and SODA have upper bounds of 32, 64, and 45 pixels respectively. These datasets often define a lower size limit as well, such as 2 pixels in TinyPerson and AI-TOD. Two scenarios in

TABLE 2.1: Threshold values for object size categories as defined by various datasets. Where applicable, values are converted to object size (i.e., the square root of the object area). For WIDER-FACE, object height thresholds are shown.

Dataset	tiny			small	medium	large
MS COCO [81]		—		0–32	32–96	96–inf
WIDER FACE [158]		—		10–50	50–300	300–inf
TinyPerson [163]	**tiny1** 2–8	**tiny2** 8–12	**tiny3** 12–20	20–32	—	—
AI-TOD [138]	**very tiny** 2–8	**tiny** 8–16		16–32	32–64	—
SODA [24]	**eT** 0–16	**rT** 16–24	**gT** 24–32	32–45	—	—

TABLE 2.2: Absolute and relative thresholds for six predefined object sizes. Relative thresholds are used as proposed in [67].

Size	Relative range	Range at COCO resolution
micro	0 – 0.38%	0 – 2 px
very tiny	0.38 – 1.52%	2 – 8 px
tiny	1.52 – 3.05%	8 – 16 px
small	3.05 – 6.10%	16 – 32 px
medium	6.10 – 18.29%	32 – 96 px
large	18.29 – 100%	96 – 525 px

which the problem of tiny object detection arises can be distinguished: small-only and multi-scale detection. Small-only datasets (TinyPerson, AI-TOD) explicitly cap object size, thereby reducing scale variation. In contrast, multi-scale datasets (DOTA, xView) span a wide range of object sizes (from very small to large) but are typically dominated by tiny instances. High-resolution images introduce additional challenges. Absolute thresholds, such as those in Tab. 2.1, do not account for image size. As a result, an object of a given pixel area is treated identically across datasets, regardless of image resolution. Datasets like DOTA, xView, and SODA contain high-resolution images that cannot be processed at full scale due to memory constraints. Downsampling leads to information loss, while patch-wise processing significantly increases runtime. These three challenges, small-only, multiscale (in both absolute and relative terms), and high-resolution detection, require different strategies to address effectively.

While some high-resolution small object detection methods retain COCO's fixed small object threshold of 32 pixels [24, 138], others increase this value to better suit datasets with larger average image sizes [32, 45]. In this work, a hybrid strategy that combines six size categories (micro, very tiny, tiny, small, medium, and large) is adopted together with relative size thresholds proposed in [67]. The absolute thresholds are derived from [163] for the very tiny, tiny, and small categories, and from [81] for medium and large. These are converted to relative thresholds with respect to the average image size in COCO, resulting in a scaling mechanism that adjusts to the resolution of each dataset. Using relative size thresholds follows a similar approach to [32, 45], with scaling performed automatically based on the image size in the dataset. This accounts for images that are significantly larger than those in COCO while emphasizing the focus on computational efficiency and lightweight detection methods for high-resolution images. Processing large images is challenging due to computational costs, particularly for small object detection, where excessive downsampling can degrade detection quality. The precise relative thresholds used are presented in Tab. 2.2.

Relative thresholds S_R are made independent of the actual image size S_I by normalizing the absolute thresholds S_O using the geometric mean image size in *coco_ test2017*, which is $S_I = 525$ pixels:

$$S_R = \frac{S_O}{S_I} \cdot 100\% = \sqrt{\frac{w \cdot h}{W \cdot H}} \cdot 100\%. \tag{2.6}$$

Here, w and h denote the width and height of the object, while W and H refer to the image dimensions[1].

2.4 Object Detection Datasets

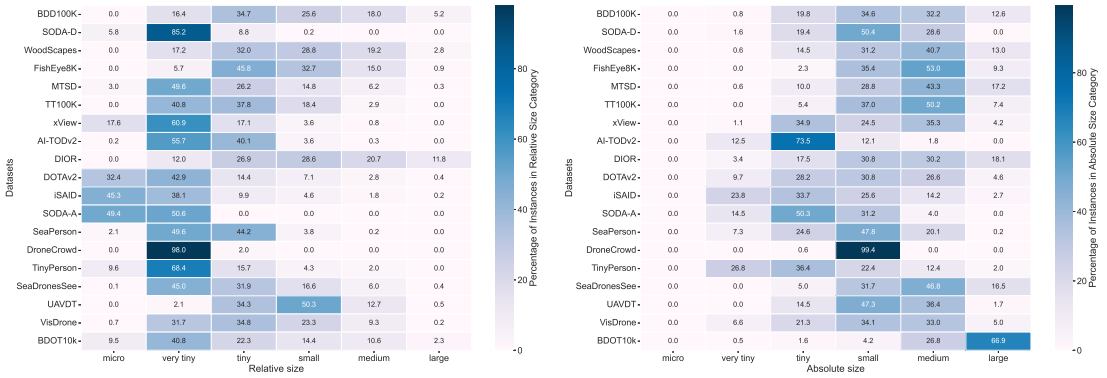
Publicly available benchmarks for small and tiny object detection largely reflect the primary applications of such systems. These benchmarks can be grouped into the following categories: remote sensing [23, 24, 32, 66, 73, 78, 131, 145, 153], where images are mostly captured by satellites for agricultural, urban planning, and environmental monitoring purposes. In these applications, images tend to be large, but processing is typically performed offline, so real-time performance is not a requirement. In contrast, UAV-based applications [33, 132, 147, 163, 164, 184], commonly used for urban traffic analysis, crowd detection and counting, or search and rescue missions, often require that all computations be performed onboard, under strict memory and time constraints. Similarly, in automotive applications [24, 35, 161, 162, 188] related to vehicle autonomization or traffic and traffic sign detection, real-time detection is essential. In most cases, the small size of objects results from the distance between the sensor and the object (e.g., the high altitude of a UAV or the long range from a vehicle). Particularly in automotive scenarios, early detection is critical to allow sufficient time for mechanical systems to respond. The primary focus of this work is real-time tiny object detection in high-resolution images for mobile robotics, emphasizing computational efficiency and fast processing.

Table 2.3 summarizes the key characteristics of several datasets that either explicitly focus on small/tiny object detection or contain a large number of instances that are small or tiny in terms of their relative size within the image. For each dataset, the average image size (S_{image}) and average object size (S_{object}) are computed as the geometric mean of height and width, also shown in Tab.2.3. Additionally, Fig. 2.1 presents size distributions across all datasets, analyzed in both absolute and relative terms. To facilitate comparison of object instance distributions across size categories, the percentage share of each category within the entire dataset is reported. The distinction between absolute and relative object sizes, as well as the importance of accounting for image resolution in real-time robotics applications, was previously discussed in Section 2.3. Objects that are small in absolute terms (i.e., few pixels) pose challenges related to visual feature extraction and detector resolution. In contrast, objects that are small in relative terms (i.e., compared to the image size) often raise issues related to computational efficiency, since large input resolutions are required to preserve detail. Importantly, objects that are small in relative terms (i.e., occupy a small portion of the overall image) can still remain difficult to detect even when cropped and processed at full resolution, as their absolute size in pixels may also be small.

[1] The full evaluation protocol is publicly available at: `https://github.com/Cufix/tinycocoapi`

TABLE 2.3: Comparison of key characteristics across object detection datasets. The number of images and annotated objects is reported for subsets with publicly available labels. S_{image} and S_{object} denote the average image and object sizes, respectively. Datasets that include video and multiple object tracking (MOT) data are marked with *.

dataset	tasks	images	objects	classes	S_{image}	S_{object}
BDD100K [162]	traffic	80k	1.5M	10	960±0	51.3±64.3
SODA-D [24]	traffic	25k	278k	9	2790±597	25.4±10.0
WoodScapes [161]	traffic, fisheye	8k	124k	5	1112±0	53.3±59.1
FishEye8K [45]	traffic, fisheye	8k	157k	5	1267±249	50.7±40.7
MTSD [35]	traffic signs	42k	206k	314	2967±840	63.1±71.6
TT100K [188]	traffic signs	10k	26k	182	2048±0	45.9±31.6
xView [73]	aerial	0.8k	602k	62	3148±317	34.9±39.9
BDOT10k [66]	aerial	61k	41M	286	12367±8939	442.1±1022.7
AI-TODv2 [153]	aerial	14k	377k	8	800±0	12.7±5.7
DIOR [78]	aerial	23k	193k	20	800±0	65.6±91.7
DOTAv2 [32]	aerial	2k	350k	18	3756±3536	33.0±49.0
iSAID [145]	aerial	2k	471k	15	3165±1606	22.7±35.2
SODA-A [24]	aerial	3k	873k	10	3627±162	15.6±7.7
SeaPerson [164]	aerial, people	4k	304k	1	1456±207	23.0±13.3
DroneCrowd* [147]	aerial, people	33k	4.8M	1	1440±0	20.0±0.8
TinyPerson [163]	aerial, people	2k	73k	2	1540±527	18.9±23.0
SeaDronesSee* [132]	aerial, people	10k	67k	5	2644±831	62.6±71.3
UAVDT* [33]	aerial, traffic	40k	764k	3	743±3	31.6±20.8
VisDrone [184]	aerial, traffic	9k	471k	12	1200±250	35.1±32.4



(A) Relative size thresholds.

(B) Absolute size thresholds.

FIGURE 2.1: Heatmap showing the percentage share of each size category across selected object detection datasets. Six predefined size categories are used, based on relative size thresholds (a): *micro* (0-0.38 %), *very tiny* (0.38-1.52 %), *tiny* (1.52-3.05 %), *small* (3.05-6.10 %), *medium*(6.10-18.30 %), *large* (18.30-100 %), and based on absolute size thresholds (b): *micro* (0-2 px), *very tiny* (2-8 px), *tiny* (8-16 px), *small* (16-32 px), *medium*(32-96 px), *large* (96-∞ px).

Among the traffic-oriented datasets, the Mapillary Traffic Sign Dataset (MTSD) [35] and SODA-D [24] are characterized by some of the largest image sizes. Although SODA-D explicitly focuses on small objects, with an average object size of only 25 pixels, MTSD exhibits greater variability in object scales. This is evident in the object size distributions shown in both Fig. 2.1b and Fig. 2.1a, where MTSD spans several size categories. SODA-D shares most of its 25k images with the Mapillary Vistas dataset [94]; however, instances with an absolute area larger than 2000 pixels, as well as heavily occluded objects, are typically annotated but tagged as ignored, leading to a smaller effective object size distribution. BDD100K [162] is similar to SODA-D in terms of camera perspective and the set of annotated classes. Both use images captured from vehicle-mounted cameras and focus on onboard traffic detection. However, the relatively small image size in BDD100K limits its relevance for evaluating methods targeting high-resolution,

tiny object detection as addressed in this work. Like BDD100K and SODA-D, MTSD [35] and TT100k [188] also use a vehicle-mounted perspective, but are dedicated to traffic sign detection. While SODA-D contains smaller objects, MTSD poses a greater challenge due to its large number of classes, high inter-class similarity, and broad variability in object and image scales. These characteristics make MTSD especially relevant for this study, which focuses on window-based tiny object detection in high-resolution images.

Detecting road objects is of particular importance in both autonomous driving and traffic surveillance. The FishEye8K dataset [45] is, to the best of current knowledge, the first to employ fixed-position fisheye surveillance cameras with a wide field of view for traffic analysis. While several other datasets also focus on traffic scenes [26, 33, 40, 161, 162, 184], most of them rely on standard cameras mounted either on UAVs [33, 184] or cars [26, 35, 40, 162]. WoodScape [161], similar to FishEye8K, uses fisheye optics, but the cameras are mounted on vehicles, resulting in a different composition and perspective of the scene. The key advantage of fixed-position fisheye cameras, as in FishEye8K, lies in their ability to cover a wide area with a single sensor. However, this introduces new challenges akin to those in small and tiny object detection. FishEye8K images often feature a top-down view and include numerous small objects, especially around the periphery, where heavy distortion occurs. This places it closer in character to aerial tiny object detection datasets such as VisDrone [184], UAVDT [33], TinyPerson [163], SeaPerson [164], AI-TODv2 [153], DOTAv2 [32]. Despite this resemblance, the strong distortion present in FishEye8K and WoodScapes means that full-resolution inference alone cannot compensate for lost details near image edges. Therefore, these datasets are excluded from experiments focused on high-resolution detection. However, a data augmentation pipeline was designed to simulate optical distortion, which successfully improved detection quality on FishEye8K using standard detectors [68]. This method is further discussed in Section 2.5.2.

Aerial datasets can be broadly categorized into two main groups: UAV-based and satellite-based. UAV datasets primarily focus on urban surveillance and traffic analysis (VisDrone [184], UAVDT [33]), people detection and counting (SeaPerson [164], DroneCrowd [147], TinyPerson [163]), or search and rescue applications, as in SeaDronesSee [132]. In contrast, satellite-based datasets such as xView [73], SODA-A [24], and BDOT10k [66] are generally aimed at urban and agricultural surveillance, and are typically characterized by significantly larger average image sizes. Some datasets, including DOTA [32] and iSAID [145], draw from both satellite and UAV sources, but their overall design and focus are more closely aligned with satellite-based collections (urban and engineering objects detection vs people and small vehicle detection). Satellite datasets (e.g., SODA-A and xView) tend to include multi-scale objects when considering absolute dimensions. However, due to the extremely high resolution of the images, most objects fall into the micro, very tiny, or tiny categories when a relative size definition is applied. A similar trend is observed in UAV-based datasets such as VisDrone, SeaDronesSee, SeaPerson, and TinyPerson. Yet, due to their lower image resolutions, these datasets generally lack micro-scale objects. Two exceptions stand out: AI-TODv2, which is explicitly designed for tiny object detection and therefore excludes large instances entirely, and UAVDT, where most objects fall into the medium, small, or tiny categories regardless of the size definition used. Overall, satellite datasets such as DOTA, xView, and SODA-A have significantly larger average image sizes compared to drone-based datasets. The low signal-to-noise ratio caused by massive and complex backgrounds,

the large variations in object densities and sizes, and nontrivial orientations make segmentation and detection in aerial images particularly challenging. For these reasons, despite the significant progress that has been made in generic computer vision in recent years, aerial-based computer vision is still an unsolved problem.

The DOTA-v2.0 [32] dataset stands out for its wide range of image sizes (from 800×800 px to nearly 30000×30000 px) and oriented bounding box (OBB) annotations, which suit the high density and rotation variability of aerial objects. Although DOTA includes tiny objects from version 1.5, its primary strength lies in its diversity of object classes (18 in v2.0) and detailed categorization of object sizes. In contrast, xView [73] features a much broader class diversity (60 subclasses under 6 parent classes), although with horizontal bounding boxes and a highly imbalanced class distribution. AI-TOD [153] is specifically designed for tiny object detection, built by filtering and merging several aerial datasets including DOTA and xView. It restricts object sizes to below 64 pixels and introduces finer granularity in object size categorization, making it a focused benchmark for evaluating detection of extremely small objects. Its updated version, AI-TODv2, refines the annotations while maintaining the same image set. Additionally, the DroneCrowd dataset focuses on small object detection in drone-captured crowd scenes, providing high-resolution images with varying crowd densities. SODA-A [24], another dataset targeting small object detection, differs in its inclusion of high-resolution Google Earth images and four clearly defined small object size categories based on area. It provides OBB annotations and focuses on variability in density, orientation, and scene complexity. TinyPerson [163] and its larger successor, SeaPerson [164], are both specialized in detecting people from afar in maritime environments. While TinyPerson uses internet-sourced video frames, SeaPerson collects its own high-resolution data, expanding significantly in both image count and instance annotations. Both datasets annotate challenging conditions such as crowds, ambiguous reflections, and uncertain poses. SeaDronesSee [132] is designed for maritime search and rescue tasks and provides high-resolution drone footage annotated with detection, tracking, and sensor metadata. Although its objects may not be absolutely small, relative size thresholds reveal a significant proportion of tiny objects. UAVDT [33] and VisDrone [184], meanwhile, focus on vehicle and pedestrian detection in UAV footage, but differ in scope. UAVDT emphasizes vehicle detection and tracking in video data, while VisDrone supports multiple tasks and includes both video and image data. BDOT10k [66], a recently released dataset, significantly expands the scale of aerial benchmarks. Covering nearly the entire territory of Poland (314,000 km^2), it provides over 60,000 high-resolution orthoimages annotated with more than 40 million instances across 286 topographical object classes. Unlike most datasets that tile images or downsample resolution, BDOT10k retains native orthoimage dimensions, resulting in an average image size of over 12,000 px, and offers annotations for both semantic and instance segmentation, as well as object detection. Object sizes vary significantly, with many tiny and small instances (Fig. 2.1a). Compared to other remote sensing datasets with satellite images, BDOT10k introduces the highest variability in both image and object scale, making it especially valuable for multi-scale and small object detection tasks.

Among the aerial datasets discussed, the focus is primarily on UAV-based datasets suitable for real-time detection systems. Video sequences, required for this purpose, are available only in SeaDronesSee, DroneCrowd, UAVDT, and VisDrone. DroneCrowd and SeaDronesSee were

selected due to their challenging characteristics. DroneCrowd is notable for its large number of tiny instances and complex backgrounds, while SeaDronesSee offers greater variability in object sizes and classes, larger images, and simpler backgrounds, making it an ideal complement to DroneCrowd. VisDrone and UAVDT are excluded due to their smaller image sizes and less complex scenarios, which do not present sufficient challenges for the objectives addressed.

Together, DroneCrowd, SeaDronesSee, and MTSD offer a diverse and challenging benchmark for the proposed method. DroneCrowd focuses on very small objects in crowded scenes, making it useful for testing detection in dense and cluttered environments with complex backgrounds. SeaDronesSee, on the other hand, has fewer objects per image but includes a wide range of object sizes and classes, larger images, and simpler backgrounds. These two datasets also provide video sequences, which are important for evaluating the `ROI Prediction` part of the system that works across consecutive frames. MTSD complements them by providing a large number of independent high-resolution images with more than 300 traffic sign classes. It includes objects of many different sizes and is collected under various sensor types and weather conditions, making it a strong benchmark for the image-based mode of the system. Together, these datasets help test the proposed method under a wide range of real-world conditions and use cases.

2.5 Tiny Object Detection Methods

Small/tiny object detection methods differ from standard object detection techniques due to the limited number of pixels representing the object in an image. In high-resolution images, the total pixel count may be higher, but processing such large images significantly slows down detection. To increase the relative size of small objects, sliding-window approaches can be applied to high-resolution images [151]. However, this method requires processing the entire image and is computationally expensive. Focus-and-detect methods aim to guide the detector to concentrate on specific regions of interest within high-resolution images. In aerial datasets, for example, objects often appear in clusters, while large parts of the image may contain no relevant objects at all. Visual recognition of tiny objects shares many challenges with generic object detection, such as intra-class appearance variation and cluttered or noisy backgrounds that can lead to false positives. However, there are also challenges unique to detecting small objects:

Limited appearance information Tiny objects occupy very few pixels, resulting in poor visual representation that makes both classification and localization difficult. Several methods address this by enhancing object features using contextual information from surrounding areas or by transforming small object representations (at the image, region, or feature level) to resemble those of larger objects.

Low signal-to-noise-ratio This issue is particularly pronounced in high-resolution aerial imagery, where objects are concentrated in specific regions and backgrounds are complex. Attention mechanisms are often used to emphasize informative features and suppress noise. Many solutions adopt a coarse-to-fine approach, performing full-resolution detection only in selected regions.

Pipeline-specific challenges Standard convolutional backbones downsample feature maps, which can cause small object features to disappear or become overwhelmed by background noise. To address this, specialized feature fusion techniques have been developed. In addition, standard anchor assignment strategies and the Intersection over Union (IoU) metric are often ineffective for tiny objects, which led to the development of alternative assignment methods and metrics.

Annotation-specific challenges Deep learning models require large, accurately labeled datasets. For very small objects, accurate annotation is difficult due to blurred object boundaries, increasing the likelihood of imprecise or inconsistent bounding boxes, even by human annotators.

High-resolution images Ultra-high-resolution images are common in aerial surveillance and satellite data, where small objects must be detected across large scenes. Although high detail helps preserve small object features, it also increases memory usage and slows down processing, posing a challenge for real-time applications. A typical workaround is to split images into smaller tiles, but this often wastes resources when many tiles contain only background. To improve efficiency, focus-and-detect methods use region selection and adaptive resizing to focus computation on the most relevant areas.

In the following sections, the major groups of small and tiny object detection methods are discussed: Focus-and-Detect (window-based), Data Augmentation, Sampling-Based, Attention-Based, Scale-Aware, Context-Aware, and Feature-Imitation methods. Focus-and-Detect methods guide the detector toward regions of interest, making them particularly effective when objects are sparsely distributed and clustered within specific areas of high-resolution images. The detection system described in this dissertation falls within this category. The Data Augmentation subsection explores various augmentation strategies tailored for small object detection. Sampling-Based methods address the limitations of anchor-based detectors for tiny objects and often include alternative metrics for anchor assignment. Attention-Based approaches enhance relevant features while suppressing background noise, improving detection under low signal-to-noise conditions. Scale-Aware methods, frequently based on the Feature Pyramid Network (FPN) [82], introduce additional modifications to better handle small-scale objects. Context-Aware methods leverage surrounding visual information to support detection. Finally, Feature-Imitation methods apply techniques such as image, region, or feature-level super-resolution, or use similarity learning to enrich small object representations. Many solutions combine elements from multiple categories to improve performance.

2.5.1 Focus-and-Detect

Focus-and-detect methods primarily target the detection of small objects in high-resolution images, i.e., objects with small relative size. In imagery captured from drones or satellites, objects often appear clustered in specific areas, while large portions of the image contain no relevant content. A straightforward approach to handling small objects is to apply a sliding-window over the full-resolution image. However, this is computationally inefficient, especially when many

windows contain only a background. To overcome this, various methods introduce an additional step to identify regions worth analyzing at high resolution.

In [116], a two-step method for detecting people in high-resolution images is proposed. In both steps, a full-resolution image is divided into tiles using a grid, and each tile is downsampled to a fixed resolution before being passed to the detector. The first step uses a coarser grid, and the second step uses a finer one. Bounding boxes from the first stage are projected onto the original image, and only fine-grid tiles overlapping with detections are used in the second stage. This method has notable limitations: if an object is missed during coarse detection, it will be skipped entirely in the final step; and the regular grid may cause objects to span multiple tiles, requiring merging and increasing the risk of errors. A similar two-stage strategy is used in [38], where downsampled images are used to predict coarse detections, and zoom-in regions are generated dynamically using a reinforcement learning strategy. This method selects regions most likely to improve detection accuracy while considering computational cost. In [96], a static tiling method based on a regular grid is applied on top of PeleeNet [139] in a micro aerial vehicle (MAV) scenario. The detector processes both the full image and the individual tiles. The confidence of detections is adjusted based on whether an object is detected in a tile (favoring small objects) or in the global image (favoring large objects). Unlike the previously mentioned works, this approach processes all tiles regardless of their content. Method proposed in [102] also uses fixed-size tiles, but since the data consists of videos, memory and attention mechanisms help select tiles in each frame. For example, tiles may be processed cyclically across consecutive frames to maintain coverage over time. Other methods attempt to reduce redundant processing by filtering tiles that do not contain any objects. In [151], an auxiliary neural network called the Objectness Activation Network (OAN) supports a simple sliding-window detector by discarding background-only tiles. A similar filtering step is employed in R^2CNN [99].

More complex systems incorporate region proposal and scale estimation. For instance, ClusDet [157] first predicts object clusters (regions with high object density) and then resizes each tile accordingly before final detection. Unlike grid-based approaches, tiles here may be padded or divided to preserve object scale. The results of both the tiles and the global image are merged. However, ClusDet requires cluster annotations, which are typically unavailable in standard datasets. A related idea is explored in [169], where a Difficult Region Estimation Network (DREN) selects regions with many hard-to-detect objects. These "difficult" tiles, along with the global image, are passed to the detector. Unlike ClusDet, DREN skips scale estimation and instead relies on preliminary detections to localize challenging regions. In [75], the authors also select tiles non-uniformly, but instead of using a separate detector, they predict a density map for each image. A sliding-window is used to find high-intensity areas, and regions that exceed a threshold are aggregated into tiles. These are processed together with the global image. The approach in [71] also relies on supervised region learning. A two-stage pipeline is used, with one detector trained to identify object-dense regions in the global image, and another trained to detect objects within those regions. Tile ground truths are generated via a Gaussian Mixture Model fitted to detection annotations. To handle partial detections at tile borders, the authors propose the Incomplete Box Suppression (IBS) algorithm in addition to standard NMS. CDMNet [34] introduces a lightweight dual-head network to predict low-resolution density and segmentation maps. These are fused to reduce noise, and only pixels marked as foreground are

used to generate tiles. After a closing operation and scale-aware rescaling, the tiles are sent to the detector. Unlike previous approaches, CDMNet omits global image detection entirely.

CRENet [142] proposes an unsupervised approach to tile selection by clustering initial detections to identify densely packed regions. A clustering algorithm is used to identify difficult regions with densely packed objects, while regions classified as easy are skipped to reduce computational cost. Like [75, 157, 169], final results combine detections from tiles and the global image. Notably, CRENet is anchor-free in both stages to better accommodate variable object sizes. Some methods focus on learning where to zoom without manual annotations. AdaZoom [155] addresses the ambiguity in defining ground-truth focus regions by using deep reinforcement learning. The tile selector is jointly trained with the detector, using detection-based rewards. The tiles vary in shape and scale, and the final predictions merge the output of both the tiles and the full image. SAIC-FPN [178] tackles the challenge of inter-image scale variation in UAV data. The method estimates the smallest object relative scale (ORS) for each image using a neural network. Based on this, the image is either upsampled via bilinear interpolation or super-resolution GAN, and then tiled uniformly for detection. This combines both focus-and-detect and feature-imitation techniques. A similar three-step framework is introduced in [31]. Here, a super-resolution network upsamples only selected tiles, not the entire image. Detection is performed twice: on the downsampled global image and on the upsampled tiles, with results merged. Training is end-to-end and incorporates original, cropped, and upsampled by super-resolution images as augmented inputs.

In general, many window-based detection strategies use full-image predictions to preserve performance on large objects [75, 96, 142, 155, 157, 169]. However, training with both full images and tiles leads to scale inconsistency, hurting detector accuracy. To mitigate tiling-related issues, some methods adjust the image resolution based on estimated object scales (either by upsampling the entire image [178] or selectively enhancing tiles [31]) before detection, while others address fragmented detections by introducing specialized post-processing techniques such as Incomplete Box Suppression (IBS) [71].

The system presented in this dissertation [67, 69, 70] belongs to the Focus-and-Detect category. A key challenge in two-step window-based systems is the risk of missing relevant regions due to false negatives in ROI selection. To address this, an additional ROI source is introduced that leverages the sequential nature of video data: tracking is incorporated into the architecture to compensate for errors in learned ROI selection, particularly for the smallest, hardest-to-detect objects and under occlusion. Unlike methods that rely on a regular grid [96, 116, 151], the proposed system places windows adaptively, reducing the risk of fragmentations for larger objects and ensuring that cropped sub-windows are better aligned with image content. In that sense, the proposed system is similar to more complex window-based architectures [142, 157, 169]. In contrast to approaches that process entire downsampled frames [75, 96, 142, 155, 157, 169], the system avoids quality loss on larger objects by employing a dual ROI strategy: within large ROIs, sub-windows are generated both by sliding-window and by crop-and-resize, which preserves spatial detail for tiny instances while maintaining continuity for larger ones. Finally, the proposed `OBS` and `OBM` algorithms ensure high-quality outputs. While `OBS` is conceptually related to IBS [71], the lack of an available IBS implementation prevents a direct comparison.

2.5.2 Data Augmentation

Data augmentation is widely used in generic object detection and other computer vision tasks to diversify training data and reduce the risk of overfitting [12]. In this section, the focus is on data augmentation methods specifically developed for improving the detection of small objects. In [63], a simple small-object enhancement method is introduced. The authors duplicate images containing small objects and apply a copy-paste operation to increase the number of small instances. This method is motivated by the under-representation of small objects in datasets such as COCO. However, to avoid introducing artifacts or noise, a semantic mask is required to accurately crop the object before pasting it into another location. Moreover, in domain-specific tasks such as traffic sign detection or aerial imagery, objects tend to appear in characteristic areas. Pasting them into random locations may affect performance. To address this, [19] uses an additional semantic segmentation network to extract road maps from UAV imagery. A size scaling factor is also computed based on nearby objects, ensuring that pasted objects are appropriately sized and positioned. RRNet also introduces a two-step detection strategy to improve the performance of anchor-based detectors on small objects. The first step uses an anchor-free coarse detector to predict object class, center point, and size. The second step refines the result using a regression module similar to Faster R-CNN [110]. In [141], each object is cropped from the image so that the center of the bounding box becomes the center of the new image, the crop size being proportional to the size of the object. In [172], a simpler cropping strategy is used, where the images are split into smaller patches using a fixed grid. Both methods incorporate multi-model fusion: [172] trains two separate detectors, one for frequent classes and another for rare classes, while [141] selects the model based on the resolution of the input image. A different augmentation strategy is introduced in [16], which uses a Downsampling-GAN (DS-GAN) to synthesize small objects from larger ones. These synthetic objects are then placed in appropriate positions and merged with the background using image inpainting and blending techniques. Like the methods in [63] and [19], this approach helps increase the number of small object instances in the dataset. In [68], the authors address small object detection in heavily distorted fisheye images. To reduce the effect of distortion on detection quality, they design a data augmentation pipeline that simulates such distortions in other traffic-oriented datasets. This approach improves detection performance on the original dataset. The augmentation pipeline includes pixel-level transformations and GAN-based style transfer, all applied offline, ensuring no negative impact on inference time during deployment.

While the proposed system does not introduce a novel data augmentation strategy, its modular architecture allows the `Local Detection Block` to be customized. This means that specialized augmentation techniques (such as those designed for specific domains or object types) can be seamlessly integrated during training to further improve performance on a given sub-problem.

2.5.3 Sampling-Based

Sampling-based methods address the limitations of anchor-based architectures when applied to small object detection. As noted in [170], anchor-based detectors struggle with small instances because the corresponding feature maps contain limited information and the anchor sizes are

often too large. In addition, small objects receive few positive anchor matches and the presence of many negative anchors increases the risk of false positives. To address these issues, several approaches have been proposed [152–154, 170, 182]. In [170], anchor matching is performed at multiple depths within the backbone network to ensure suitable anchor scales for objects of varying sizes. The matching strategy is modified to a two-step process using lower-than-usual IoU thresholds. Additionally, a max-out background label is introduced to suppress negative anchors. However, this method was tailored for face detection, so it uses square anchors, with the smallest size being 16 pixels. To improve anchor-object alignment, [182] introduces a new anchor design strategy alongside the Expected Max Overlapping (EMO) score, which measures the average IoU between anchors and ground-truth boxes. Their method reduces anchor stride by enlarging the feature map, uses shifted anchors, and applies random object shifts during training. In [159], the Sampling Fusion Network (SF-Net) combines feature up-sampling, multi-level feature aggregation, and an inception module to increase the receptive field and improve anchor sampling quality for small object detection.

In [152–154], new metrics were proposed to replace the Intersection over Union (IoU) in the label assignment process. IoU has been shown to be highly sensitive to small location shifts, which makes it unreliable for small object detection. To address this, the DotD, NWD, and RFD metrics were introduced as alternatives to IoU for anchor matching. These metrics have already been discussed in detail in Section 2.3.

In contrast to specialized sampling-based methods, the proposed system does not modify anchor assignment directly. Instead, its modular design allows `Local Detectors` that implement such strategies to be seamlessly integrated, making the system complementary to approaches that improve anchor design or label assignment for small objects.

2.5.4 Attention-Based

When detecting tiny objects in high-resolution images, the signal-to-noise ratio is often very low. To address this, many attention-based methods [37, 79, 88, 106, 159, 160] have been developed to suppress background noise and emphasize relevant features. These methods are frequently combined with multi-level feature fusion [37, 79, 88, 106, 159] to enhance the representation of small objects.

In [160], a Recurrent Neural Network (RNN) with attention is used to guide the detector to focus on relevant areas, such as roads for vehicles or roadsides for traffic signs. In [37], the authors propose an Attention-Guided Balanced Pyramid (ABP) that adaptively fuses features from different pyramid levels. This fusion is controlled by a two-part attention mechanism: a Level-Based Attention (LA) module that learns weights for each feature level, and a Spatial Attention Network (SA) that highlights regions likely to contain objects. Several works [88, 106, 159] adapt the channel attention mechanism based on the Squeeze-and-Excitation (SE) Block [52] to highlight channels important for detection and noise suppression. Other attention modules include pixel attention in [159] and spatial attention in [88]. In [79], the authors further model spatial relationships between pixel pairs to improve attention precision.

Unlike feature-level attention modules, the proposed SegTrackDetect system exploits the sequential nature of video data to improve focus on relevant regions. By incorporating object tracking into the ROI selection step, the system acts as a form of temporal attention, recovering regions with consistent motion across frames. This method reduces the risk of missing tiny objects that may be missed by the `ROI Estimator` and mitigates noise without requiring additional feature-level attention.

2.5.5 Scale-Aware

Tiny objects are typically represented by only a few pixels, and due to the downsampling in convolutional layers, high-level feature maps used for detection often contain little to no information about them. To mitigate this issue, many methods fuse feature maps from different layers [46, 49, 87, 168], combining the rich semantic information of deep layers with the fine spatial resolution of shallow layers. Feature Pyramid Network (FPN) [82] and Adaptive Spatial Feature Fusion (ASFF) [84], originally proposed for generic object detection, have since been adapted to tiny object detection. For example, in [168], global features are fused with multi-scale ROI features. In [87], the authors introduce the Image Pyramid Transformation Module (IPGT) to address misalignment between deep and shallow features, and then apply the Image Pyramid Fusion Module (IPGF) to combine them. Gong et al [46] demonstrate that naive FPN-based fusion degrades the performance for tiny object detection. To counter this, they propose a statistically-estimated fusion factor that dynamically controls the contribution of deep and shallow features. In [49], the authors combine an attention mechanism with feature fusion. The Context Attention Module (CAM) produces multi-scale attention heatmaps, while the Scale Enhancement Module (SEM) guides the network to focus on appropriate object scales in each layer. Features from subsequent layers are then fused. Other methods that integrate feature fusion with attention mechanisms include [37, 79, 86, 88, 106, 159], which are discussed in more detail in Section 2.5.4. R^2CNN [99] applies feature pyramid pooling to reduce false positives in high-resolution satellite imagery. It follows a focus-and-detect strategy using a sliding-window, but speeds up computation by pre-classifying tiles to filter out background-only regions. DSFD [76], the Dual Shot Face Detector, includes a Feature Enhancement Module (FEM) that processes a set of multi-scale feature maps through upsampling, concatenation, and dilated convolutions. These enhanced features are then passed to the detection head. DSFD also employs a sampling strategy to increase the number of positive anchors, combining feature fusion with anchor optimization.

Alternative approaches to scale-awareness avoid traditional feature pyramids. In [93], the Single Stage Headless Face Detector (SSH) replaces the feature pyramid with multiple detection modules, each applied to feature maps with different strides. SSH is fully convolutional and relatively lightweight. QueryDet [156] addresses the high computational cost of high-resolution feature maps by performing multi-stage fusion at the feature map level. Low-resolution features are first combined with higher-resolution ones to create a sparse representation that filters out background pixels. This approach resembles focus-and-detect methods, but operates entirely in feature space. In [141], multi-model fusion is used, with three detectors trained on images of different resolutions. DetectoRS [103] introduces feedback connections that link each pyramid layer

to its corresponding backbone layer. It also employs Switchable Atrous Convolutions, enabling each feature to be examined multiple times with different receptive fields.

Unlike most scale-aware methods that rely on feature fusion, the proposed system achieves scale robustness through its ROI handling strategy. Large ROIs spanning multiple windows are processed with sliding detection windows to preserve tiny object features, while crop-and-resize maintains continuity for larger instances. This dual mechanism ensures that both small and large objects are represented at appropriate scales. Moreover, thanks to the modular design, feature pyramid-based detectors can also be integrated as `Local Detectors` when needed.

2.5.6 Context-Aware

Due to the limited visual information present in small objects, especially in the deep layers of convolutional networks, numerous methods have been proposed to improve detection by incorporating contextual cues from surrounding regions [7, 20, 53, 54, 80, 126, 167]. In [20], the authors extend R-CNN [44] with a simple two-branch pipeline: one branch encodes the object proposal region, and the other encodes its surrounding context. The resulting features are concatenated and passed to the classification head. The Inside-Outside Net (ION) [7] extracts context features using a Recurrent Neural Network (RNN) and fuses them with multi-scale ROI features, making the model both context- and scale-aware. Similarly, [53], proposed for tiny face detection, combines context-aware and scale-aware strategies by employing a coarse image pyramid and dedicated detectors for each scale, all sharing a common backbone. The context is captured through enlarged receptive fields that include surrounding background regions.

One way to increase the receptive field is through skip connections, as in [137], where a Spatial Refinement Module (SRF) is introduced to recover spatial information lost through skip connections, thereby improving localization. In [126], a semi-supervised learning approach is used to learn relevant context classes (e.g. the human body for face detection), which help in the detection of associated small objects. The authors of SINet [54] argue that standard ROI Pooling in two-stage detectors harms contextual reasoning. They introduce Context-Aware ROI Pooling based on deconvolutions to retain broader contextual cues. SINet is also scale-aware, using separate detection heads tailored to different object sizes. CAD-Net [167] proposes a Context-Aware Detection Network for small object detection in satellite imagery. It combines global contextual features, a pyramid of local contextual features, and an attention mechanism to enhance detection accuracy. In [80], spatial context is used to re-detect low-confidence objects. Based on the observation that certain classes often cluster spatially in UAV imagery, the authors re-weight the class probabilities of low-confidence detections by considering their proximity to high-confidence ones.

While the system proposed in this thesis does not explicitly include dedicated context modeling modules, as many context-aware methods do, it preserves context through its `ROI Estimation` and `ROI Prediction` stages. The coarse ROI regions retain surrounding scene structure, and detection windows are placed at the centers of meaningful regions rather than on a regular grid, ensuring that the context around objects is maintained.

2.5.7 Feature-Imitation

To address the weak representation of small objects in deep feature maps, several methods [5, 6, 56, 57, 77, 95] leverage advances in Generative Adversarial Networks (GANs). The Perceptual Generative Adversarial Network (Perceptual GAN) [77] highlights that naive feature enhancement, such as combining features from shallow layers, does not always improve detection. Instead of using multi-level pyramids or feature/image upsampling, the authors propose a super-resolution approach at the feature level to make small-object representations resemble those of larger objects. The perceptual discriminator has a dual role: distinguishing between generated and real features, and evaluating whether the generated feature maps aid the detection task. A similar feature-level GAN approach is introduced in [95], which adds direct supervision during training. In this setup, a downsampled image is passed through a generator, while the original high-resolution image is fed directly into a feature extractor. The generated features are then compared with features extracted from the original image, enabling supervised training of the generator.

Other approaches apply GANs directly to image regions extracted by a baseline detector. In [5], small and blurry face regions are enhanced via super-resolution, and the discriminator is trained to distinguish generated from real images as well as to classify face versus non-face images. Classification loss is back-propagated through the generator to improve training. This idea is extended to multi-class object detection in [6], where the discriminator additionally outputs class probabilities and bounding-box offsets. Unlike the single-class case, both classification and regression losses are used to guide the generator. The image-level superresolution is also explored in [56, 57] to detect small objects in satellite imagery. These methods apply super-resolution directly to the input image before detection.

Beyond super-resolution, other strategies perform feature imitation in alternative ways. In [149], a knowledge distillation approach called Self-Mimic Learning (SML) is proposed to improve the features of small pedestrians. A mimic loss is used to teach small object features to resemble those of larger instances. This method does not increase the inference time and can be integrated into any detector. In [61], small pedestrian detection is enhanced by mimicking the cued recall process observed in human memory. Embedding learning is used to help detect small objects by recalling the appearance of similar larger objects. Unlike SML, this method can also leverage clues from larger instances during inference. Finally, the Self-supervised Feature Augmentation Network (SFANet) [97] uses a self-supervised learning paradigm. During training, the network receives a pair of images, one upsampled and one downsampled, and extracts features from both. Features from the higher-resolution image are then used to guide those from the lower-resolution image, improving the robustness of small-object representations.

Unlike feature-imitation methods that use GANs, super-resolution, or knowledge distillation to enhance small-object features, the SegTrackDetect system proposed in this thesis preserves high-resolution information through `ROI Estimation` and tracking-based `ROI Prediction`. This ensures that `Local Detectors` operate on regions that retain fine details without the extra computation required for feature enhancement. As a result, the system remains lightweight and suitable for real-time applications, while still achieving strong performance.

2.6 Video Object Detection Methods

Video object detection focuses on identifying objects across sequences of video frames, rather than in single images. Unlike static image detection, these methods can take advantage of the fact that objects usually move smoothly over time. By using information from nearby frames, video-based detectors can better handle issues like motion blur, occlusions, and low visibility. This has led to the development of methods that go beyond standard image detectors to improve accuracy in challenging video conditions.

A comprehensive review of video object detection methods is provided in [58]. One recent approach is DiffusionVID [113], which addresses these challenges using an object-centric cascade refinement architecture composed of multiple self-refinement modules. The model efficiently aggregates information from object proposals through Adaptive Condition Generation and Conditioned Refinement, improving denoising quality by using adaptive conditions based on a spatio-temporal coreset. TransVOD [179] proposes a spatial-temporal Transformer-based architecture that aggregates features across frames. Its encoder captures the relationships between objects in consecutive frames. However, this method is only evaluated on standard-sized objects and has not been tested for small or tiny object detection. Cui et al. [27] introduced modules that enhance feature representation by modeling temporal relationships between neighboring frames and feature maps. While effective, the approach achieves only 10.1 FPS on the ImageNet VID validation set, limiting its real-time applicability.

Another group of methods focuses on feature propagation via optical flow [186]. YOLOV [121] builds on this idea by introducing a feature aggregation strategy tailored for one-stage detectors. The resulting feature set is more robust to motion blur and occlusion than single-frame detectors. Similarly, the MEGA method consolidates global context features into local features and then uses a Long Range Memory (LRM) module to merge both global and local information into keyframes, thereby enhancing detection performance. Flow-Guided Feature Aggregation (FGFA) [185] aggregates features from neighboring frames to mitigate the effects of degraded object appearance. Relation Distillation Networks (RDN) leverage Region Proposal Networks (RPN) to generate object proposals, then compute relationships between proposals in a reference frame and a supportive pool, enhancing features through relational reasoning.

While these methods improve robustness under varying exposition and motion conditions, they often struggle with full or partial occlusions. In this work, this issue is addressed by using a Kalman filter to propagate previous detections into the current frame. This guides the detector, particularly in cases where small objects become partially or fully invisible, improving both the continuity and the accuracy of detection across frames.

2.7 Region of Interest Generation

The concept of Region of Interest (ROI) generation was introduced in early deep learning-based object detectors [43, 44, 47, 110]. In the first two-stage models, ROIs were generated using relatively simple algorithms - Selective Search [130] in [43, 44] and Edge Boxes [189]

in [47]. While these methods enable object localization, they significantly increase inference time, which is particularly problematic in time-sensitive applications such as mobile robotics. A major breakthrough came with Faster R-CNN [110], which introduced the Region Proposal Network (RPN) - a convolutional network that generates ROI and shares computations with the object detector. This allowed all processing to be performed on the GPU, substantially improving the detection speed to around 7 FPS. However, this is still insufficient for real-time applications by modern standards. Today, many state-of-the-art general-purpose object detectors [12, 18, 135, 136] adopt a single-stage architecture that eliminates explicit ROI generation in favor of faster inference. Nonetheless, ROI-based methods remain relevant, particularly in window-based approaches [38, 75, 102, 116, 157], where they are used to focus processing on selected regions and improve detection quality for small objects. In the context of this work, which focuses on window-based tiny object detection, the ROI generation is defined as the process of selecting detection window coordinates for full-resolution inference. Although window-based methods were already discussed in Section 2.5, this section delves deeper into their ROI generation step.

Uniform overlapping tiles are employed in several window-based methods [38, 96, 102, 116, 151]. These represent the simplest form of ROI generation - tiles are generated using a regular sliding-window grid, but additional mechanisms are applied to discard empty or uninformative regions. In [116], ROI selection follows a two-step process, both stages relying on a regular grid and image downscaling. Since the target objects are relatively large by modern tiny object detection standards, the authors can afford significant downscaling without severely harming detection performance. First, a coarse detection step is applied using large sliding-window tiles downscaled by a substantial factor. In the second step, a finer sliding-window grid is applied, and only the tiles that overlap with the initial coarse detections are selected for further processing. Importantly, the second stage does not operate at full resolution either - the selected tiles are also downscaled before detection, although to a lesser degree. Notably, like in the proposed system, this method processes video data and exploits temporal continuity by reusing ROIs across consecutive frames, which improves efficiency by avoiding redundant computations. One limitation, however, is that the same detector is used in both stages, potentially leading to inconsistencies in object scale between coarse and fine detections. Similar to [116], the Dynamic Zoom-in Network (DZN) [38] also uses a two-stage approach where coarse detections inform fine-grained inference. However, DZN introduces a key difference: instead of using coarse detections directly to select tiles, it employs a reinforcement learning (RL) policy that takes as input both the coarse detections and the history of previously selected tiles. The policy then selects regions that are likely to benefit most from high-resolution refinement. Unlike [116], DZN performs coarse detection on a downsampled full image without any sliding-window. While this aggressive downsampling may harm small object detection, the learned RL policy is better equipped to compensate for it through informed tile selection. Like the previous method, DZN also relies on a regular tile grid, though it allows for slight positional adjustments of selected tiles to maximize gain. Final detections are obtained by replacing coarse detections with those obtained from zoomed-in regions. Another approach to tile selection is proposed by Plastiras et al. [102], where, unlike in [116] or [38], there is no preliminary coarse detection stage on a downsampled image. Instead, the authors leverage the temporal continuity in video data to selectively process a subset of uniform full-resolution tiles in each frame. Initially, the tiles from the entire image

are processed, but in subsequent frames, tiles that contain no objects are skipped to reduce computation. To achieve this, the system employs attention and memory mechanisms: the attention module helps prioritize tiles, while the memory keeps track of past detections, under the assumption that objects are unlikely to appear or disappear abruptly between consecutive frames. To mitigate the negative impact of false negatives, where an object might be missed in a single frame and subsequently ignored, they introduce a reset mechanism, which ensures that each tile is periodically revisited after a fixed number of frames. In contrast to the two-step coarse-to-fine pipeline in [116], where tile selection is based on detections from a downscaled image, Plastiras et al. avoid downsampling entirely. Compared to DZN [38], which learns to select tiles using reinforcement learning and a history of zoomed-in regions, they rely on simpler heuristics and temporal consistency. Their method is particularly well-suited for scenarios where motion is minimal and frame-to-frame consistency can be exploited to reduce redundant computation. The Objectness Activation Network (OAN) proposed in [151] aims to filter out empty regions from processing in extremely high-resolution aerial images. OAN employs a uniform grid to divide the image into patches, similar to sliding-window approaches like those in [116] and [102]. However, instead of applying detection to every patch, it predicts an objectness activation map, which is then thresholded to decide whether a given patch should be passed to the detector. This enables selective processing without the need for coarse-to-fine pipelines or reinforcement learning strategies as in [38]. Moreover, OAN shares its backbone with the detector, which improves computational efficiency compared to two-stage systems where region proposal and detection are handled by separate models. Unlike methods that reuse temporal information (e.g., [102]), OAN is applied independently per image, without temporal memory. Nonetheless, its modular design allows integration into any detection architecture, making it a flexible filtering mechanism for large-scale images. In contrast, [96] applies the simplest possible approach: no filtering is performed, and all uniformly selected tiles along with the full-resolution image are passed to the detector. The final predictions are then obtained by merging detections from both the full image and all tiles. This approach may be suitable for dense scenes where objects are uniformly distributed, as the additional complexity of implementing a filtering mechanism may not offer significant benefit. However, in sparser scenarios, this strategy can introduce substantial overhead due to unnecessary processing of empty tiles.

In ClusDet [157], regions of interest are selected using the Cluster Proposal Network (CPNet), which predicts coarse cluster regions from the global image. These clusters are then passed to the Scale Estimation Network (ScaleNet), which estimates the object scale within each cluster to avoid downsampling areas with very small objects - a common cause of missed detections. Unlike earlier methods such as [38, 96, 102, 116, 151] that rely on uniform tiling, ClusDet uses a more targeted, data-driven approach to focus computation on dense object regions. Redundant clusters are refined using the Iterative Cluster Merging (ICM) module, but overlapping areas between clusters are not fully addressed. Instead, standard non-maximum suppression (NMS) is applied to the final detections, which can lead to false positives in overlapping zones. Another limitation is that ROI generation is supervised, requiring ground-truth cluster annotations, which are difficult to define and introduce ambiguity into training. To preserve object scale during detection, selected ROIs may be divided into smaller tiles and padded to fit the detector input size, rather than being directly resized. ScaleNet and CPNet share a backbone with the

initial global detector, but the fine detector used on selected clusters is a separate model, potentially increasing the complexity and inference time of the system. A similar non-uniform tiling strategy is employed in DMNet [75], where regions of interest are selected based on predicted object density. The method uses a density map generation network, inspired by crowd counting approaches such as MCNN [175], to estimate a density map from the input image. This map is processed using a sliding-window: for each tile, the number of objects is estimated by adding the density values. Tiles exceeding a predefined object count threshold are marked as active in a binary density mask. Connected regions in this mask are then converted into detection windows, while low-density areas are discarded to avoid wasting computation on the background. Like ClusDet [157], DMNet requires supervised training, including ground-truth density maps for the density prediction module. This dependence on additional supervision introduces extra labeling effort and potential uncertainty in how density annotations are defined. CDMNet [34] builds upon the idea of density-guided ROI selection but introduces a coarse-grained approach to reduce computational cost. Instead of producing a high-resolution density map and binarizing it using a sliding-window, as done in DMNet [75], CDMNet jointly predicts a low-resolution density map and a binary segmentation mask, which is used to identify object-dense regions. Like in ClusDet [157], CDMNet performs adaptive resizing of selected ROIs based on estimated object density to avoid excessive downsampling and better preserve object scale. The approach in [71] treats region selection as an object detection task, where ground truth labels for focal regions are generated using a Gaussian Mixture Model. Similarly to [34, 75], this introduces a dependency on additional supervision. However, unlike density-based maps, this method directly leverages the spatial distribution of objects, offering a more explicit formulation of ROI generation.

Some recent methods avoid relying on manually annotated regions by learning tile selection in an unsupervised or reinforcement learning manner. CRENet [142] proposes an unsupervised approach in which initial coarse detections are clustered to identify dense regions, and easy regions are skipped to save computation. Similarly, AdaZoom [155] uses reinforcement learning to address the ambiguity of defining ground-truth focus regions. It jointly trains a tile selector and detector using detection-based rewards, allowing adaptive selection of tile shapes and scales without annotated region labels. This differs from prior supervised approaches by providing an end-to-end learnable solution for zoom-in policies, eliminating the need for manual region annotations.

The quality of ROI generation is typically evaluated indirectly through general object detection metrics like Average Precision (AP) and Average Recall (AR). This thesis focuses specifically on `ROI Estimation` and `Prediction` by employing lightweight segmentation and tracking methods. By leveraging the sequential nature of video data, ROI selection is improved while relying on standard object detectors to perform the final detection step. This approach enables efficient and accurate detection of tiny objects without requiring complex or heavily supervised ROI generation networks.

2.8 Object Tracking

Object tracking is a core task in computer vision that focuses on maintaining the identities of objects as they appear in consecutive video frames. In Multiple Object Tracking (MOT), the goal is to link individual detections over time to form continuous tracks, despite challenges such as occlusions, missed detections, and changes in appearance. This capability is essential in many real-world applications, including autonomous driving, video surveillance, and robotics, where understanding the movement of objects over time is crucial to safe and effective system behavior. Bounding boxes are the most common label format used in object tracking, although segmentation masks (MOTSs) are sometimes also used. In Single Object Tracking (SOT) on the other hand, there is always one object to be tracked. The focus of this work is on MOT, due to dealing with real-life applications where several objects appear in each frame.

MOT algorithms can be divided into two groups: tracking-by-detection [1, 9–11, 21, 25, 50, 144, 148, 174, 183] and joint detection and tracking [124, 143, 150, 166, 173, 181] approaches. In the first case, the outputs of an object detector are used as inputs to a separate tracking algorithm, with both components operating independently. The tracking algorithm can be relatively simple and may not use any image information. In contrast, the second approach performs detection and tracking jointly within a single neural network. Given reliance on a modular system for object detection, only tracking-by-detection methods are considered.

As in object detection, the performance of object tracking is evaluated using standard metrics such as the number of true positives (TP), false positives (FP), false negatives (FN), as well as precision and recall. Additionally, the number of false alarms per frame (FAF) is often reported. For trajectory-level evaluation, identity switches (IDSW) are a key metric. An identity switch occurs when a track is lost and then re-initialized with a new ID after N frames, or when the ID changes between consecutive frames. [8] introduced two widely used metrics: Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP). MOTA is defined as

$$MOTA = 1 - \frac{\sum_0^T (FN_t + FP_t + IDSW_t)}{\sum_0^T GT_t}, \tag{2.7}$$

where T is the total number of frames in the sequence, FN_t, FP_t, and $IDSW_t$ are the numbers of false negatives, false positives, and identity switches in frame t, respectively, and GT_t is the number of ground-truth objects in frame t. MOTP, on the other hand, measures the localization accuracy and is defined as the average IoU over all true positive assignments across the sequence:

$$MOTP = \frac{1}{|TP|} \sum_{i \in TP} \text{IoU}(\hat{B}_i, B_i), \tag{2.8}$$

where $|TP|$ is the number of true positives and $\text{IoU}(\hat{B}_i, B_i)$ is the IoU between the predicted bounding box $\hat{B}_i$ and the ground truth bounding box B_i. To assess the quality of tracked trajectories, additional metrics are used. These include MT (Mostly Tracked), PT (Partially Tracked), and ML (Mostly Lost), which represent the number of ground-truth trajectories that are tracked

for more than 80%, between 20%-80%, and less than 20% of their lifetime, respectively. A fragmentation (Frag) is counted each time a ground-truth trajectory is interrupted and not matched to any tracker output.

Identification metrics were introduced in [112] to evaluate the accuracy of associating predicted trajectories with ground-truth ones. Each ground-truth trajectory is assigned to exactly one predicted trajectory. The number of true positive identity matches (IDTP) corresponds to matched trajectory points in overlapping regions. False positive matches (IDFP) refer to unmatched segments of predicted trajectories or predictions not assigned to any ground-truth trajectory. Similarly, false negatives (IDFN) are unmatched segments of ground-truth trajectories or ground-truth tracks without any corresponding prediction. Based on these values, identification precision and recall are defined as

$$IDP = \frac{IDTP}{IDTP + IDFP}, \tag{2.9}$$

$$IDR = \frac{IDTP}{IDTP + IDFN}. \tag{2.10}$$

Luiten et al. [89] proposed the Higher Order Tracking Accuracy (HOTA) metric, which explicitly accounts for association quality over time. For each true positive object c, they define sets of true positive associations (TPA), false positive associations (FPA), and false negative associations (FNA). The per-object association score is given by

$$A(c) = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|}. \tag{2.11}$$

The overall HOTA score is computed as

$$HOTA_\alpha = \sqrt{\frac{\sum_{c \in TP} A(c)}{|TP| + |FN| + |FP|}}, \tag{2.12}$$

where α is the localization threshold.

MOT algorithms are employed primarily to enhance detection quality rather than focusing on tracking itself. Consequently, the tracking component of the system is evaluated indirectly using the object detection metrics discussed in Section 2.3.

Among datasets dedicated to MOT, a significant portion belongs to the Multiple Object Tracking Benchmark collection [4, 28, 29, 74, 91, 125, 133, 134, 146]. The MOT15 [74], MOT16 [91], MOT17 [91], MOT20 [29], MOTS [133], Head Tracking [125], and STEP [146] datasets primarily focus on pedestrian tracking. Although MOT16 introduces 11 additional classes, these are not considered during evaluation. The Head Tracking dataset was created by replacing the MOT20 labels with head annotations, significantly improving object visibility in crowded scenes. The CTMC [4] dataset captures cell migration and division, characterized by chaotic motion. In MOTS, the annotations are segmentation masks, while in the STEP dataset, each pixel is assigned an object ID. TAO [28], GOT-10k [55], and TrackingNet [92] contain multiple object classes, but these are examples of single object tracking (SOT) datasets. MOT-style annotations are also present in datasets focused on autonomous driving [39, 123, 162], video surveillance [33, 147, 184], and search and rescue missions [132]. Most MOT datasets contain

either a limited number of object instances (e.g., MOT15, MOT16, KITTI) or feature large object sizes (e.g., MOT15, MOT16, MOT17, MOT20, CTMC, KITTI). As discussed in Section 2.4, the SeaDronesSee [132] and DroneCrowd [147] datasets are selected to evaluate the video object detection system due to their challenging conditions, wide variability in object sizes, and diverse background characteristics.

In object tracking algorithms, several subproblems can be identified, including trajectory estimation and association, similarity metrics, and re-identification. To predict the future location of objects based on past observations, the Kalman filter is commonly used [60]. One of the earliest methods to apply this approach was SORT [9], which uses the Kalman filter for motion prediction, followed by matching detections to predicted tracks using the Hungarian Algorithm [72] and Intersection-over-Union (IoU) as the similarity metric. In [148], IoU is replaced with a Deep Association Metric, which incorporates both appearance information (cosine distance in feature space) and motion information (Mahalanobis distance between predicted and measured states). In [10, 11], association is performed purely on the basis of IoU across adjacent frames, enabling high-speed operation but limiting the method to scenarios where the object displacements between frames are small. The methods proposed in [1, 21, 25, 50, 144, 174, 183] introduce new trajectory association strategies, while [42] presents a motion estimation approach based on repeated neural networks. The comparison of state-of-the-art object trackers is provided in [3]. The review includes Deep OC-SORT [90], ByteTrack [174], StrongSORT [165], and OC-SORT [17]. Cao et al. [17] show that the Kalman filter can obtain state-of-the-art tracking performance when the noise accumulated during occlusion is fixed. They use object observations provided by the detector to compute the trajectory of the object during the occlusion. Object localization is more accurate when derived from detection rather than from state estimations based on the linear motion assumption in the Kalman filter. If an object can be associated with the detection method after a period of being untracked, it cannot be associated with the previous trajectory estimated by KF due to the temporal error magnification. Thus, the OC-SORT method from [17] re-updates the Kalman gain and *a posteriori* estimate covariance using the virtual trajectory estimated for the objects that were not detected. In the improved Deep OC-SORT [90], visual appearance (deep visual embedding) is used to enhance the accuracy of visual object association for Multi-Object Tracking. The StrongSORT [165] method demonstrates that global association can be performed without relying on object appearance and that Gaussian process regression can effectively address the issue of missing detections. Zhang et al. [174] propose the ByteTrack method, which suggests that incorporating detection boxes with low detection scores and filtering out background detections provides better results compared to tracking only high-score detections.

The goal of this thesis is to propose a lightweight method that improves small object detection in the sequence of high-resolution images by utilizing object tracking. A similar system is presented in [59]. However, in [59], the goal of the TLD (Tracking-Learning-Detection) is to enable long-term tracking of an unknown object in advance. The system contains two standard components: a tracker that follows the object from frame to frame and a detector that localizes the object in the image. An additional learning component estimates the detector error and uses it to update the detector. The considered system is pretrained to detect known categories of objects. In the proposed system, the tracking component guides the detector to focus on specific areas of the

image. This strategy aims to improve the performance of small object detection in large images while avoiding computationally expensive detectors that operate on the entire image or low-quality detectors that work on images with reduced resolution. This can be achieved by utilizing lightweight pretrained object detectors and a standard tracking method, as demonstrated later in this dissertation. Taking into account the drawbacks of the discussed tracking methods, heavy reliance on the tracking module is avoided. Instead, a `ROI Prediction Module` is proposed to serve only as a guide, making it less sensitive to errors introduced by the tracking system.

Chapter 3

Proposed SegTrackDetect System for Tiny Object Detection

3.1 System Architecture

Detecting objects in high-resolution images presents considerable computational challenges. In theory, such images could be processed either at full resolution in a single pass or by applying an exhaustive sliding-window approach. However, the first option demands substantial computational resources, while the second leads to very slow processing times. These limitations make both strategies unsuitable for practical applications, especially in robotics, where real-time performance is essential and computational resources are often limited. A common alternative is to downscale the input image, but this causes small and tiny object features to vanish. To address these challenges, the modular system proposed in this thesis introduces a guidance mechanism that directs the detector to specific regions of interest, allowing it to perform full-resolution inference selectively. This strategy enables both accurate detection and efficient processing. The proposed framework is also highly customizable, allowing users to balance speed and accuracy based on the specific requirements of a given application. It supports both tiny object detection and multi-scale detection in high-resolution images, making it suitable for a wide range of real-world scenarios.

The proposed approach, shown in Fig. 3.1, is based on a focus-and-detect paradigm. Instead of relying on single-shot detection with downscaling or applying a naive sliding-window strategy, the system directs the detector's attention to specific, fixed-size Regions of Interest (ROIs) in high-resolution images for efficient object detection. These ROIs are derived from two complementary sources: (1) a lightweight semantic segmentation network that estimates foreground regions at a low resolution, and (2) a tracking-based `ROI Prediction Branch` that propagates previously detected objects across time. While `ROI Estimation` alone can achieve good detection quality when applied to high-resolution inputs, incorporating tracking-based `ROI Prediction` enables the system to operate on lower-resolution data without compromising recall. By merging these

two sources, the system maintains high detection performance while significantly improving computational efficiency. Once the fused ROI mask is obtained, the `Detection Windows Proposal Module` extracts fixed-size crops, which are then passed independently through a lightweight object detector. The detected bounding boxes are mapped back to the global image coordinates and postprocessed to remove redundant and partial detections using the `Global Filtering Module`. This filtering step is essential to mitigate common issues associated with window-based inference, such as duplicated or fragmented boxes.

While the system architecture presented in Fig. 3.1 illustrates the core components of the final refined version of the SegTrackDetect system, multiple variants and iterative improvements were developed throughout the research process. These versions are discussed in detail in Chapters 4, 5, and 6, where their differences are highlighted and justified through ablation studies that support key design choices. This chapter presents a detailed pipeline for each module, while the simplified diagrams in the following three chapters focus on high-level architectural differences that illustrate the evolution of the system.

All three chapters build upon the author's previous work published in [65, 67, 69, 70]. The final system is tailored for applications with constrained computational resources that require real-time performance, while also offering high adaptability for specific deployment scenarios. This chapter introduces the full system architecture (Fig. 3.1), detailing each component from ROI generation to detection postprocessing. It also includes an analysis of the datasets, focusing on spatial resolution and object scale, which provides the necessary foundation for the subsequent experimental validation. Throughout this thesis, the proposed system is evaluated using four challenging datasets: the Mapillary Traffic Sign Dataset (MTSD) [35], and three sequential datasets (SeaDronesSee [132], DroneCrowd [147], and ZebraFish [100]) used across the following three chapters due to their temporal structure. MTSD, which consists of individual images, is not suitable for experiments involving video object detection and is therefore limited to the evaluation presented in Chapter 4. For an in-depth discussion on architectural design and the influence of individual modules on performance, refer to Chapter 4 (`ROI Estimation Module`), Chapter 5 (`ROI Prediction Module`), and Chapter 6 (`Global Filtering Block`).

3.1.1 Region of Interest Estimation

The `ROI Estimation Module`, shown in Fig. 3.1, leverages visual cues from the current frame to estimate regions likely to contain foreground objects of interest. These regions are obtained using a segmentation network operating at reduced resolution, based on a U-Net-like architecture [114] with a shallow ResNet-18 backbone. This choice helps preserve details of smaller objects that might otherwise disappear in deeper architectures. Binary ground-truth masks are generated as exact representations of bounding box annotations. The effectiveness of these design decisions is validated experimentally in Chapter 4, using the initial version of the system (TinyROI).

In the subsequent chapters, several segmentation resolutions are evaluated. In the final version of the system, segmentation is performed at a lower resolution compared to the estimation-only variant. This reduction significantly increases the processing speed. Thanks to the `Prediction`

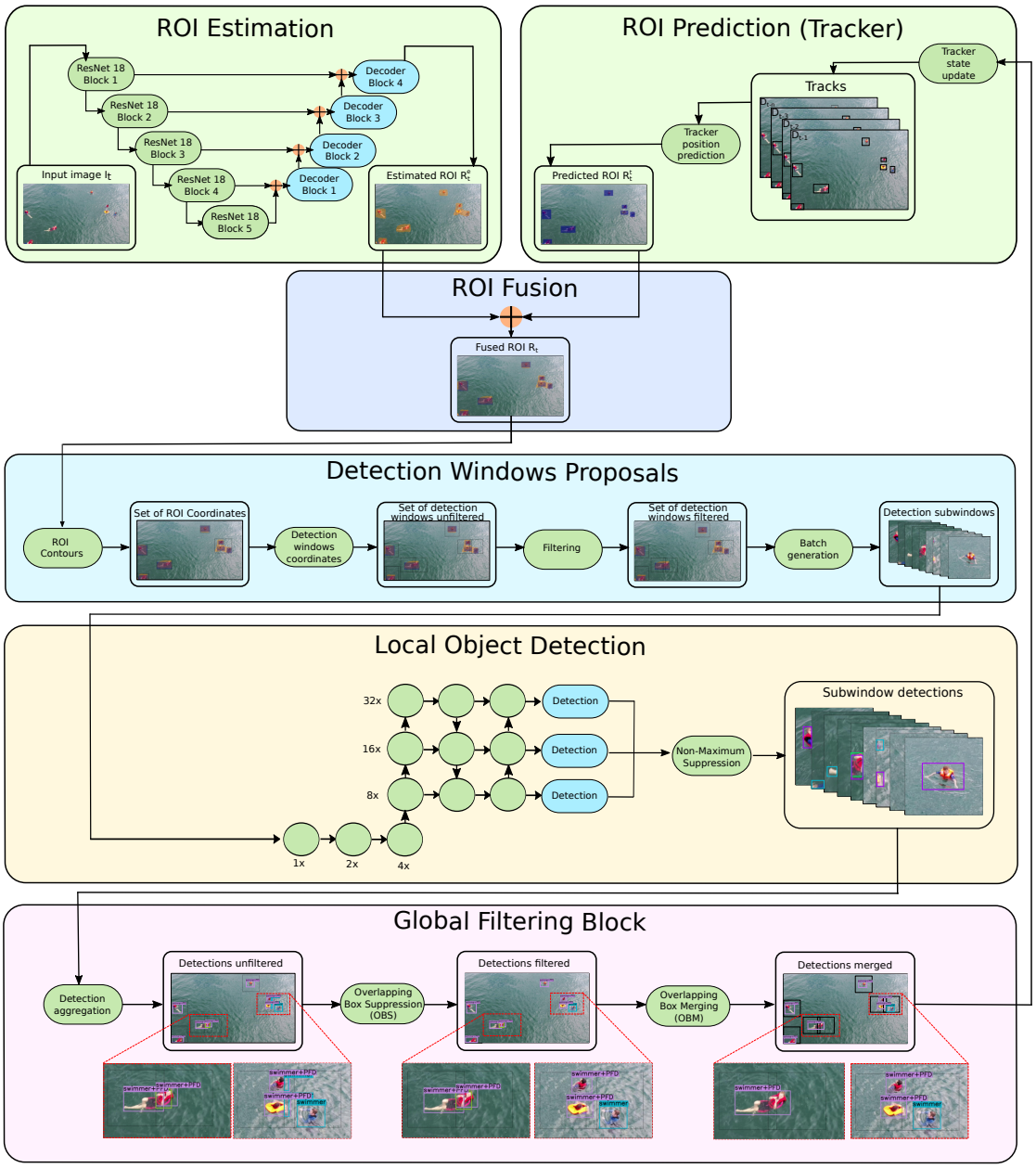


FIGURE 3.1: Overview of the proposed small and tiny object detection system architecture. The system focuses the detector's processing on selected fixed-size regions within high-resolution images to enable efficient detection of small objects. These regions are identified using two ROI sources: `ROI Estimation` generated by low-resolution binary segmentation of the current frame (R^e_t) and `ROI Prediction` derived from an object tracker (R^t_t). The combined ROI mask (R^f_t), created via element-wise OR, is transformed into detection windows by the `Detection Windows Proposal Module`. These windows are analyzed by a lightweight detector, and the resulting detections are projected back onto the original image coordinates (Detection aggregation). To reduce false-positive partial detections, the `Global Filtering Block` filters and merges the detections before updating the tracker's state, which supports `ROI Prediction` in future frames.

`Module` (discussed next), which leverages temporal cues from object motion, ROI quality (particularly for the smallest objects) can still be preserved through tracking. As a result, the combined use of lower-resolution segmentation and lightweight tracking enables a substantial boost in frame rate without compromising detection coverage.

3.1.2 Region of Interest Prediction

The `ROI Prediction Module`, illustrated in Fig. 3.1, complements the segmentation-based `Estimator` by leveraging object motion to recover regions that may be missed due to low resolution or challenging visual conditions. It initializes object tracks from detections found within estimated ROIs of the initial N frames and subsequently predicts object positions in later frames using a constant-velocity motion model. These predictions are converted into binary masks and fused with the current segmentation mask to guide the detection process. The output of the Tracker Position Prediction (Fig. 3.1) stage consists of bounding box coordinates, which are then converted into a binary mask format representing the predicted ROI. This mask is fused with the current segmentation-based ROI to guide the detection process.

Implemented using a lightweight multi-object tracker inspired by SORT [9], the module maintains per-object state using a Kalman filter and performs data association based on IoU. Although the tracker may occasionally lose accuracy in crowded scenes, its sole purpose is to propose ROIs rather than preserve object identities, which ensures minimal computational overhead. This design allows the system to benefit from temporal consistency and improved recall for small or occluded objects, while maintaining high processing speed. The effectiveness of this module is further analyzed in Chapter 5, where it is shown to stabilize detections and increase recall, particularly for objects prone to being missed by segmentation alone.

3.1.3 Region of Interest Fusion

The `ROI Fusion Module` combines the outputs of the segmentation-based `Estimation Module` and the motion-based `Prediction Module` to generate a unified ROI mask. Both branches produce binary heatmaps at a fixed resolution, which are merged using an element-wise logical OR operation. Since the `Prediction Module` outputs bounding box coordinates, these are first converted into a binary mask format aligned with the resolution of the segmentation map to ensure compatibility during fusion. The resulting fused ROI mask serves as the input for the `Detection Window Proposal` stage.

While the system supports operation with either branch individually, the default configuration employs both. This joint setup enables high detection recall while minimizing unnecessary computation in non-relevant areas of high-resolution video frames. Alternative configurations, such as estimation-only and tracking-only, are revisited in the ablation study in Chapter 5 to highlight the trade-offs between speed and detection performance. Notably, the tracking-only mode offers a lightweight alternative for fast inference in static scenes where new object entries are infrequent.

In addition, the sliding-window variant of the SegTrackDetect system is also evaluated. This configuration is implemented by replacing the fused ROI mask with an all-foreground mask, thereby prompting the `Detection Window Proposal Module` to perform exhaustive window-based coverage. This setup enables a fair comparison against conventional sliding-window methods, while minimizing differences in other system components and hyperparameters.

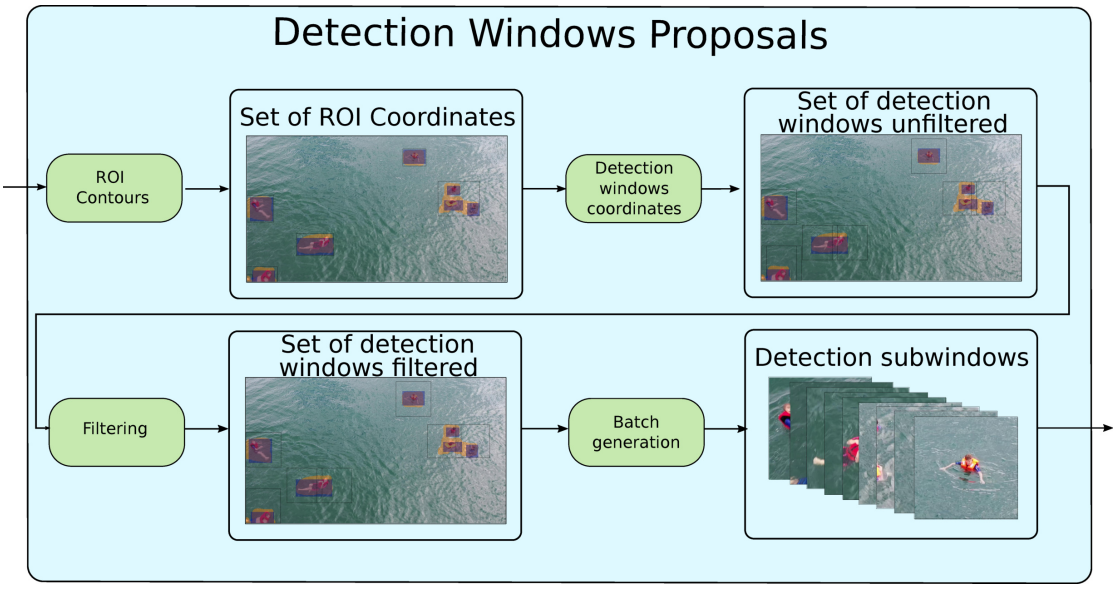


FIGURE 3.2: The zoomed-in architecture of the `Detection Windows Proposal Module`.

3.1.4 Detection Windows Proposals

For clarity, the more precise architecture of the `Detection Windows Proposal Module` is presented in Fig. 3.2. This module converts the fused ROI mask, produced by combining segmentation and tracking-based proposals into a set of fixed-size candidate image regions for the object detector (Detection Subwindows in Fig. 3.2). Connected foreground regions are extracted from the binary ROI mask, and their bounding boxes form an initial list of detection window candidates (Set of ROI Coordinates in Fig. 3.2). Based on these ROI bounding boxes and the detector's input size, an unfiltered set of detection windows is generated by placing windows centered within each region.

The final SegTrackDetect version of this module handles ROIs larger than the detector input size by combining two strategies: cropping with downscaling to preserve contextual integrity, and sliding-windows within ROIs to retain fine features of tiny objects. This approach enables lowering the detector input size close to the average object size in the dataset, while properly handling outliers. Consequently, the system supports multi-scale detection without compromising speed or accuracy, as smaller detector inputs can be effectively used. The unfiltered set of detection windows is then filtered to remove redundancy; for detailed information on the filtering method and its evaluation, please refer to Chapter 4. These filtered windows are subsequently passed to the Batch Generator, which crops the final detection windows from the original high-resolution image. Unlike earlier designs (e.g., TinyROI in Chapter 4) that allowed detection windows to extend beyond image boundaries requiring padding, this system constrains all windows to lie fully within the image frame. This modification eliminates padding artifacts that could degrade detection quality and reduces unnecessary background processing. This flexible `Window Proposal` strategy balances runtime efficiency and detection performance, enabling the system to process high-resolution images in real time while maintaining high-quality results for both tiny-object and multi-scale detection scenarios.

3.1.5 Local Object Detection

The `Local Object Detection Module` performs object detection within the proposed subregions extracted from the original high-resolution image. By restricting inference to these selected ROIs, the system significantly reduces computational load while maintaining full spatial resolution for tiny and small objects. The module leverages an adaptive detection window strategy that enables the use of lightweight detectors such as YOLOv7 Tiny [136] across diverse datasets.

For the SeaDronesSee and DroneCrowd datasets, detection operates at an input resolution of 512×512 pixels, while a smaller resolution of 160×256 is used for the simpler ZebraFish dataset. These relatively low input sizes are made possible by the earlier `Window Proposal Module`, which tiles or downsamples larger ROIs as needed to maintain coverage of all relevant regions.

YOLOv7 Tiny was chosen for its balanced trade-off between speed, accuracy, and stability, outperforming or matching newer variants in the evaluated scenarios while requiring fewer computational resources. Training follows the protocol described in Chapter 4, including preparation of cropped subwindows, scale augmentation, and inclusion of negative samples to reduce false positives. During inference, local detections are filtered via Non-Maximum Suppression (NMS) and mapped back to the full image coordinates (Detection Aggregation in the `Global Filtering Block` of Fig. 3.1), ensuring accurate global localization.

3.1.6 Global Filtering Block

The `Global Filtering Module` finalizes object detections across all proposed sub-windows by removing redundant and fragmentary results - common artifacts in systems that rely on overlapping detection windows (Fig. 3.3). An effective `Global Filtering` step is essential to improve detection quality in such setups, as shown in Chapter 6.

In SegTrackDetect, this is handled by two proposed complementary algorithms: `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`. `OBS` improves upon standard Non-Maximum Suppression (NMS) by incorporating information about the origin of each detection. This allows it to distinguish full-object detections from incomplete fragments caused by intersecting windows, helping to reduce false positives without discarding correct results - a common issue with NMS in dense scenes. While `OBS` works well when full detections are available, it cannot recover objects that are only partially visible across multiple windows. To address this, `OBM` is used to merge partial detections from overlapping windows when no complete detection is present. This is particularly useful when large ROIs are tiled into multiple sub-windows, as often happens when objects are large or ROI localization is imprecise. Together, `OBS` and `OBM` form the `Global Filtering` stage of the system, ensuring that both false positives and incomplete detections are effectively handled. This improves the system's ability to perform multi-scale detection without sacrificing precision or recall.

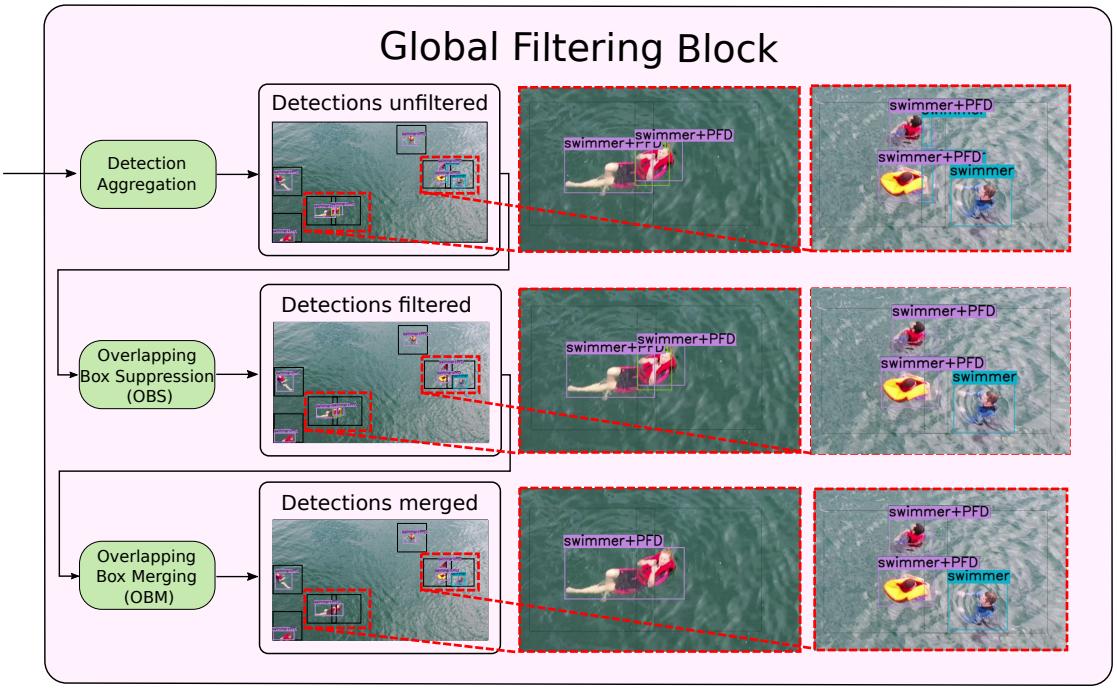


FIGURE 3.3: The zoomed-in architecture of the `Global Filtering Block`.

The experimental sections in the following chapters explore several configurations of the `Global Filtering Block`. In the baseline TinyROI system (Chapter 4), standard Non-Maximum Suppression (NMS) is used as the sole filtering mechanism. In contrast, both versions of SegTrackDetect, introduced in Chapters 4 and 5, adopt the more context-aware `Overlapping Box Suppression (OBS)` algorithm. The `Overlapping Box Merging (OBM)` method is introduced later in Chapter 6, where its integration with `OBS` forms the final version of the filtering block. This progression enables an in-depth comparison of global post-processing techniques, ultimately demonstrating that the combined `OBS`+`OBM` strategy provides the most robust and accurate detection results across a range of conditions.

3.2 Overview and Evaluation of Benchmark Datasets

The proposed modular object detection system is evaluated across four benchmark datasets: the Mapillary Traffic Sign Dataset (MTSD) [35], SeaDronesSee [132], DroneCrowd [147], and ZebraFish [100]. Each of these datasets presents distinct challenges and supports the evaluation of different aspects of the detection pipeline.

The Mapillary Traffic Sign Dataset (MTSD) is a large-scale dataset of traffic signs captured in diverse real-world conditions. Its complexity stems from the high intra-class variability, diverse backgrounds, and dense urban scenes. Despite its relevance to real-world autonomous driving applications, MTSD consists solely of individual frames, without any temporal continuity. Consequently, it is utilized exclusively in Chapter 4, where the evaluation is restricted to the estimation-based variant of the system operating on single images. A comparative analysis with

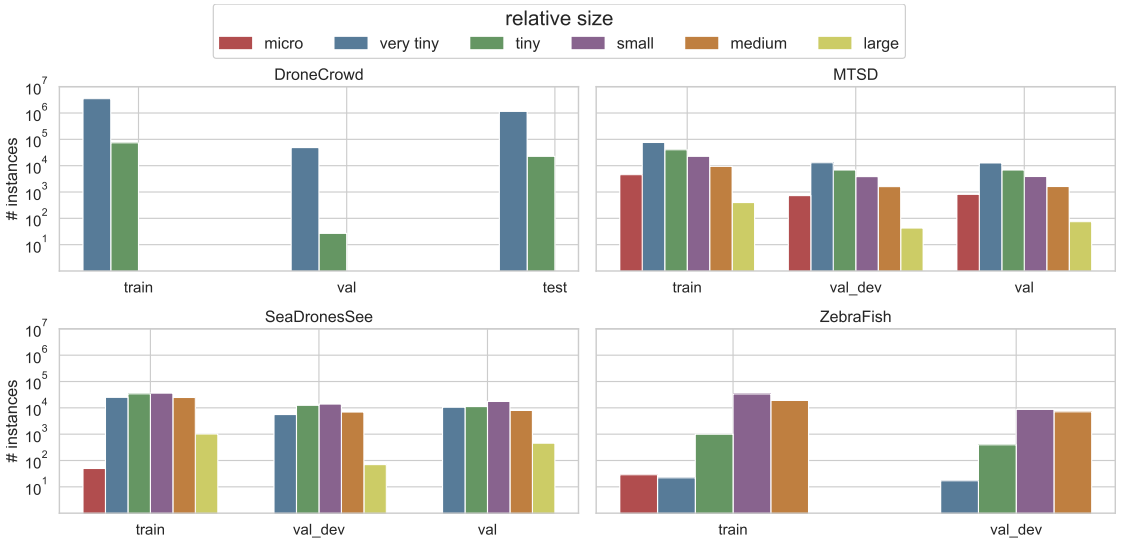


FIGURE 3.4: Comparison of object scale distributions (based on relative size thresholds) across all annotated splits of the four evaluated datasets. Since not all datasets contain labeled test sets, additional *val_dev* splits were extracted from the training sets to be used for evaluation in training and hyperparameter tuning, while the original validation sets served as test sets. The following mapping is used: in DroneCrowd, the original labeled test split is used as is; in MTSD and SeaDronesSee, *val_dev* was artificially extracted from the training set, with the original validation sets used as test; in ZebraFish, which only provides a training set and an unlabeled test set, two sequences from the train set were used as *val_dev*.

other traffic sign datasets is provided later in this chapter to illustrate the specific challenges MTSD presents.

Tracking-oriented datasets are used to evaluate the full system. This includes SeaDronesSee, which features high-resolution images and focuses on UAV-based search and rescue missions in open sea environments; DroneCrowd, initially developed for crowd detection and counting from aerial footage, but used here through its available 2D bounding box annotations; and the comparatively simple ZebraFish dataset, which centers on tracking a single fish class in tank environments. Despite its simplicity, the ZebraFish dataset includes large images and small object instances. It was used extensively during the development of the final SegTrackDetect system due to its visual clarity, which makes it particularly well suited for presenting qualitative examples and illustrating the algorithmic workflow in a transparent and interpretable manner.

All three datasets are analyzed in detail in the following sections, with a focus on image resolution, object scale, general characteristics, and the data splits used for training, validation, and testing (Fig. 3.4). The differences between them are highlighted to demonstrate that, taken together, they constitute a comprehensive benchmark. This diversity enables the evaluation of the proposed system across a range of detection scenarios, underscoring its versatility and adaptability to different real-world use cases.

Image Detection Dataset Tab. 3.1 summarizes key properties of five traffic sign detection datasets relevant to object detection in high-resolution images. The size of the image and the object is represented by the geometric mean of width and height ($s = \sqrt{w \cdot h}$). From

TABLE 3.1: Comparison of traffic sign detection datasets, with image and object dimensions summarized by their geometric mean $s = \sqrt{w \cdot h}$

Dataset	Images	Objects	Classes	Image size	Object size
BTSD [129]	9006	13480	117	1418	91±56
GTSDB [51]	506	852	43	1043	43±32
RTSD [118]	59188	104358	198	1052±189	39±23
TT100K [188]	16811	26349	182	2048	45±31
MTSD [35]	**41909**	**206388**	**314**	**2837±911**	**63±71**

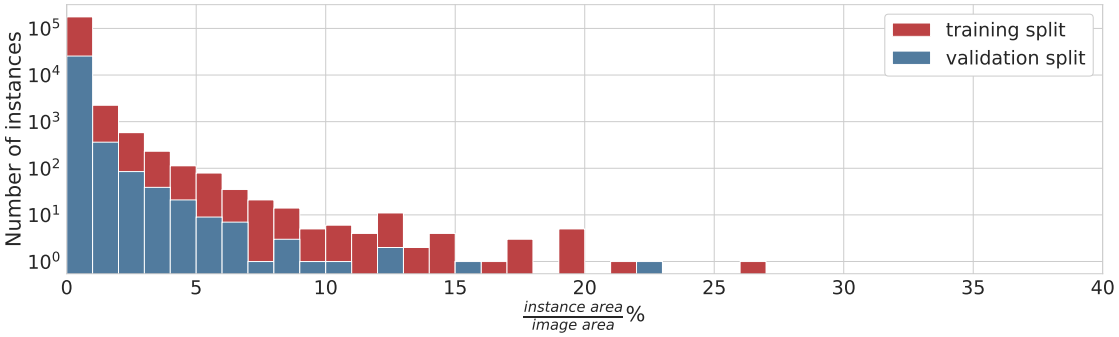


FIGURE 3.5: Distribution of relative object areas in the MTSD dataset.

these, the Mapillary Traffic Sign Dataset (MTSD) was selected due to its combination of high-resolution images, large object instance count, and diverse class set. MTSD contains 41909 labeled training images and 10544 unlabeled test images. Each object is annotated with an axis-aligned bounding box and assigned to one of 314 classes grouped into five semantic categories: *information*, *complementary*, *regulatory*, *warning*, and *other-sign*, with the latter accounting for roughly 70% of all annotated objects. The dataset's class imbalance is addressed in the proposed detection system by applying a weighted loss during training. Most images exceed 10 megapixels, while over 99% of objects occupy less than 1% of the image area, emphasizing the need for efficient small object detection strategies capable of handling large, high-resolution inputs. Figure 3.5 presents a histogram of relative object areas in the MTSD dataset, calculated as the ratio of each bounding box area to the corresponding image area. The distribution confirms that the majority of objects occupy less than 1% of the image area, with very few medium or large objects present. A strong predominance of small object sizes motivates a detection system design that emphasizes efficient processing of high-resolution images by focusing on selected Regions of Interest (ROIs) containing objects. This histogram highlights the challenge presented by MTSD and supports the need for ROI-based approaches combined with effective filtering and merging algorithms specifically tailored for the detection in high resolution images. The multiscale nature of the dataset pose additional challenges compared to tiny-only detection and facilitates the need for efficient large ROI handling strategy together with the measuring and filtering postprocessing algorithms. Figure 3.4 shows the distribution of object scales within MTSD, highlighting a strong dominance of small objects and only sparse medium-to-large objects. This characteristic aligns well with the target application of the ROI-based detection framework, which focuses detector attention on selected regions containing objects in high-resolution imagery. Since MTSD does not provide ground-truth annotations for the test split, performance evaluation is conducted on the official validation set. For training and hyperparameter tuning, a *validation_dev* split is extracted from the training data, equal in size to the official validation set, ensuring proper

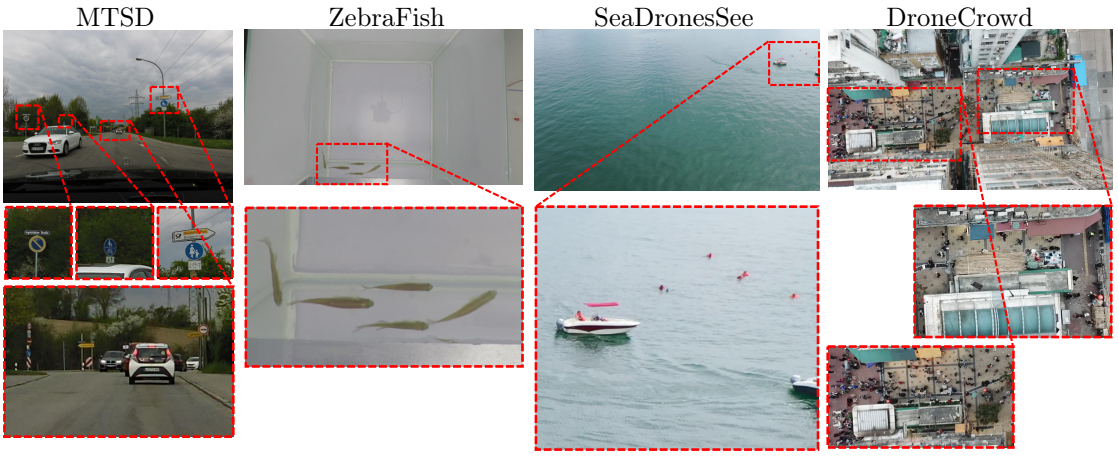


FIGURE 3.6: Example frames from the four datasets used in this thesis (MTSD, ZebraFish, SeaDronesSee, and DroneCrowd), illustrating the diversity of detection scenarios.

separation of training and evaluation.

Video Detection Datasets To evaluate the proposed detection system in varied real-world conditions, three publicly available video-based datasets are utilized: SeaDronesSee, DroneCrowd, and ZebraFish. Originally developed for multi-object tracking (MOT), all datasets provide frame-level annotations from which object detection labels are extracted.

The SeaDronesSee dataset [132] consists of drone-captured aerial imagery focused on maritime search and rescue operations. It includes multiple object classes such as people, life jackets, and boats, covering a wide range of object scales - from extremely small to large instances (Fig. 3.4). Due to the open-sea environment, the dataset is characterized by sparse object distribution and relatively uniform backgrounds, making it a representative benchmark for evaluating detection performance in low-density scenes with minimal visual clutter, but with multi-scale object settings. In contrast, the DroneCrowd dataset [147] features drone footage recorded in dense urban environments. It focuses on detecting individual human heads in crowded public spaces, where objects are consistently small and densely packed. The complex, cluttered backgrounds and heavy occlusions present a challenging scenario for tiny object detection algorithms, particularly in terms of robustness to noise. The third dataset, ZebraFish [100], captures underwater sequences of fish moving within aquarium tanks. It includes both 2D and 3D tracking annotations and features a small number of object instances per frame. Despite its simpler structure, ZebraFish provides a controlled environment in which to assess system performance, particularly the ability to detect small, fast-moving objects against dynamic backgrounds.

Figure 3.4 illustrates the distribution of object sizes across these datasets, confirming their relevance for tiny and multi-scale object detection in high-resolution images. Example frames from all datasets are presented in Fig. 3.6.

Evaluation Metrics The evaluation protocol used in this work follows the general structure of the COCO benchmark but extends it to better suit the challenges of small object detection in high-resolution imagery. In particular, detection quality is assessed using multiple object

TABLE 3.2: Object size thresholds used in the evaluation of the proposed tiny object detection system. Relative thresholds follow the image-size-adaptive scheme introduced in [67], enabling consistent scale categorization across datasets with varying resolutions. For reference, absolute size thresholds (in pixels) are also provided, computed based on dataset-specific image dimensions. The evaluated datasets include MTSD [35], SeaDronesSee [132] (SDS), DroneCrowd [147] (DC), and ZebraFish [100] (ZeF20). In the case of MTSD, average image size was used to derive pixel-based thresholds due to variability in resolution.

size	**range [%]**	**MTSD range [px]**	**SDS range [px]**	**DC range [px]**	**ZeF20 range [px]**
micro	0.00–0.38	0–12	0–10	0–5	0–7
very tiny	0.38–1.52	12–47	10–43	5–21	7–30
tiny	1.52–3.05	47–94	43–87	21–43	30–61
small	3.05–6.10	94-188	87–175	43–87	61–123
medium	6.10–18.29	188–564	175–526	87–263	123–370
large	18.29–100.00	> 564	> 526	> 263	> 370

size categories, defined by relative size thresholds rather than fixed pixel values. This strategy allows for consistent evaluation across datasets with varying resolutions and image dimensions. Inspired by previous work [24, 138], the standard COCO small category is divided into six finer-grained groups: micro, very tiny, tiny, small, medium, and large. These categories follow the relative thresholding scheme proposed in [67], where thresholds scale automatically based on image resolution. This scaling strategy highlights the system's focus on computational efficiency in high-resolution scenarios.

Table 3.2 presents the precise relative thresholds along with their corresponding absolute pixel values for each dataset. Relative object size distributions are illustrated in Fig. 3.4. The full evaluation protocol is available online[1]. For SeaDronesSee, MTSD, and ZebraFish, evaluation uses COCO-style IoU thresholds ($0.50 : 0.95 : 0.05$), with a cap of 100 detections per image. DroneCrowd, due to the high object density and annotation imprecision, is evaluated using a single IoU threshold of 0.5 and a maximum of 500 detections per image.

Together, the four benchmarks used in this work provide a diverse and well-balanced foundation for evaluating the proposed SegTrackDetect framework. They span a range of application scenarios, from high-resolution image detection to real-time video analysis, and cover both single-class and multi-class tasks across varying object scales and background complexities. This diversity ensures that all components of the system `Estimation`, `Prediction`, `Filtering`, and `Window Proposal Generation` are thoroughly validated under realistic and challenging conditions (see Fig. 3.6 and Fig. 3.4).

3.3 Novelty of the proposed system

The SegTrackDetect system [67, 69, 70] introduces a novel modular approach to high-resolution object detection with a strong emphasis on computational efficiency and real-time performance – capabilities largely absent in existing window-based methods such as ClusDet [157], DMNet [75], and AdaZoom [155]. These prior works often focus on accuracy at the cost of flexibility or runtime efficiency.

[1] `https://github.com/Cufix/tinycocoapi`

At its core, the system features a segmentation-based `ROI Estimation Module` that directly learns from detection labels, bypassing the need for intermediate representations such as density maps or region clustering, which are commonly used in methods like Focus-and-Detect [71], DMNet [75], and CDMNet [34]. This simplification enables consistent performance across both sparse and dense detection scenarios - conditions where crowd-based estimation techniques often by design struggle to detect isolated or non-clustered objects. The thesis (Chapter 4) shows that the proposed `ROI Estimation` strategy consistently outperforms three different sliding-window implementations, including a baseline uniform tiling approach and the publicly available SAHI framework [2], in terms of both computational efficiency and detection quality, as evaluated on four diverse benchmark datasets.

To further improve efficiency in sequential scenarios, a lightweight `ROI Prediction Module` based on tracking is introduced. This component enables substantial computational savings by reducing the resolution requirements of the `Estimation Network`, an advantage particularly relevant in robotics, where video streams are the norm and single-frame inference is rare. When the outputs of both ROI branches are fused, the resulting system outperforms several state-of-the-art models, including general object detectors [62, 135, 136, 140], tiny-object methods [103, 156], and video-based approaches [113], while maintaining a significantly lower parameter count. As shown in Chapter 5, ablation studies confirm that this improvement stems directly from the `Prediction Module`, enabling real-time operation with comparable detection quality to the estimation-only variant at much lower input resolution.

The system's versatility is enhanced by a dedicated large ROI handling strategy, which retains essential features for tiny objects while preserving structural context necessary for multi-scale detection. This dual benefit enables reduced detector input size, lowering computational complexity and increasing throughput, while maintaining robustness in real-world environments where scale variability and dynamic object motion are common [132].

In addition, two new post-processing algorithms are introduced: `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`. Designed specifically for window-based detection systems, these methods exploit spatial window layout to eliminate redundant detections and recover fragmented instances. Although conceptually related to Incomplete Box Suppression (IBS) from Focus-and-Detect [71], `OBS` and `OBM` are distinct, have open-source implementations, and are validated in multiple experiments. As shown in Chapter 6, the proposed methods outperform traditional NMS when applied globally across detection windows, improving both precision and recall.

The complete SegTrackDetect system is released as an open-source, end-to-end framework with extensive customization capabilities and support for embedded deployment. To date, it is the only publicly available modular framework for efficient, window-based object detection with support for segmentation- and tracking-driven ROI generation, aside from SAHI [2], and is designed for practical integration in robotic systems, as discussed in Chapter 7.

The effectiveness of each major component of the proposed system is validated through extensive experiments: Chapter 4 supports the first auxiliary thesis on efficient `ROI Estimation`, Chapter 5 confirms the benefits of the lightweight `ROI Prediction Module` for real-time video processing,

and Chapter 6 demonstrates the superiority of the proposed filtering and merging strategies. Together, these results provide comprehensive evidence to support the three auxiliary theses presented in this work.

Chapter 4

ROI Estimation-based Tiny Object Detection

4.1 Introduction

Object detection involves localizing and classifying objects in images, typically using axis-aligned bounding boxes. While considerable progress has been made in detecting medium and large objects, tiny object detection remains a persistent challenge, particularly in high-resolution scenarios where targets occupy only a few pixels. This limitation is especially relevant for real-world applications like mobile robotics, aerial imagery, or traffic analysis, where detectors must operate under strict runtime constraints while maintaining high accuracy.

High-resolution input images are often downscaled to meet the memory and speed requirements of real-time applications. However, this downsampling affects the smallest objects, rendering many of them undetectable due to the loss of spatial detail. Increasing input resolution or model complexity could improve detection, but at a significant computational cost, which is undesirable in embedded or real-time settings. An alternative approach is the sliding-window method, which divides the image into overlapping patches that are processed independently. While this improves recall for tiny objects, it also introduces substantial computational overhead due to the large number of background-only tiles evaluated (Fig. 4.2b). To address these limitations, this chapter proposes a detection pipeline based on learned `Region of Interest (ROI) Estimation`. Instead of processing all parts of the image equally, the system predicts which regions are likely to contain objects using a lightweight semantic segmentation model operating on a downscaled version of the input. The identified ROIs are then used to guide full-resolution detection, preserving important visual details only where needed and greatly reducing the number of detector calls compared to the naive sliding-window approach. This strategy achieves a balance between detection accuracy and computational efficiency, outperforming single-shot detection with downsampling while being significantly faster than the sliding-window method (Fig. 4.1).

To validate the proposed estimation-based approach, two variants of the proposed system are introduced:

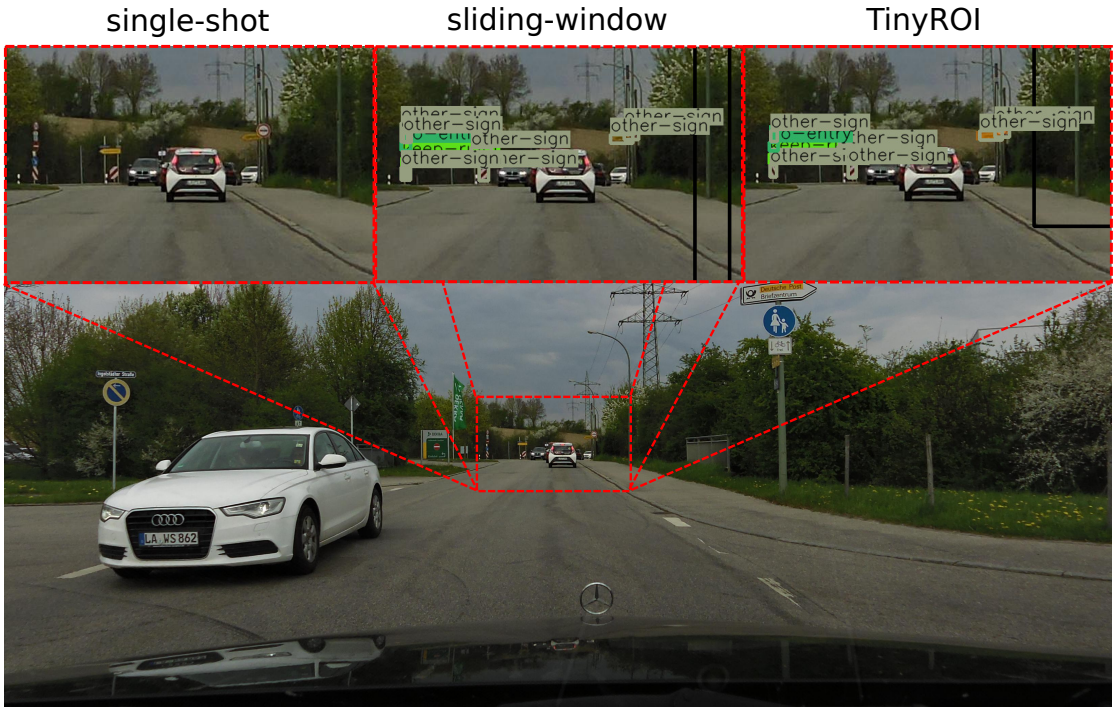


FIGURE 4.1: Detection of small objects in high-resolution images. Single-shot detection on a downsampled image fails to capture tiny objects. The sliding-window approach improves accuracy but is computationally intensive and slow. The proposed method achieves accurate results more efficiently, with significantly reduced computational cost.

- TinyROI, a minimal implementation of the learned ROI estimation pipeline,
- SegTrackDetect, an enhanced framework that integrates large ROI handling strategy, `Global Filtering` via `Overlapping Box Suppression (OBS)`, and several runtime optimizations.

Both systems are assessed against traditional baselines, including single-shot detection with downsampling and multiple implementations of the naive sliding-window method. These baselines represent two extremes on the speed-accuracy spectrum: one prioritizes computational efficiency at the cost of small object recall, while the other maximizes recall through exhaustive search at high computational cost. To ensure a robust and meaningful comparison, all systems are evaluated on datasets featuring a wide range of object densities and sizes. This includes scenes with both clustered and sparsely distributed targets, as well as significant scale variation across objects of the same class. By maintaining a consistent detection backbone and preprocessing pipeline across all experiments, the evaluation isolates the impact of the `ROI Estimation` strategy itself. While SegTrackDetect includes a more sophisticated `Global Filtering Block` implementing `OBS` (compared to NMS in TinyROI), this does not affect the comparison: within each framework, both the sliding-window and estimation-based versions use the same `Global Filtering Block`. This ensures that any observed differences in detection quality and speed are solely due to the `ROI Estimation` method, demonstrating the advantage of the estimation-based pipeline over the naive sliding-window approach. Results are reported in terms of standard

detection metrics (AP, AR), computational cost (FPS, detector calls), and qualitative visualizations. Additionally, the framework is benchmarked against SAHI [2], a widely used open-source sliding-window implementation.

Through the comprehensive evaluation, the chapter aims to validate Auxiliary Thesis #1: "Estimating Regions of Interest (ROIs) using a deep neural network enhances both detection quality and inference speed compared to the naive sliding-window approach". The findings support this claim by showing that the proposed ROI-based method reduces the number of windows required for high detection performance, achieves comparable or better recall of tiny objects, and significantly improves runtime efficiency, making it well suited for embedded and real-time applications.

4.2 Contribution

The system proposed in this chapter introduces a learned `ROI Estimation` stage into the window-based tiny object detection pipeline. It is developed as an alternative to the naive sliding-window approach, which is often used to improve detection quality for small and tiny objects. While effective at detecting smaller objects compared to single detection with downsampling, this method introduces significant computational overhead, especially when objects are sparsely located, by processing a large number of background-only tiles. The proposed alternative is designed to offer similar or even better improvements in detection quality while reducing computational cost by discarding empty regions using a deep learning-based `ROI Estimation` stage that selects promising areas for full resolution inference.

All three approaches are illustrated in Fig. 4.2. Single detection (Fig. 4.2a) requires only a single image to be passed through the detection network. However, this approach does not allow the entire high-resolution image to be processed at once due to memory and runtime constraints, so downsampling is required. While fast and effective for detecting large objects, it reduces the visibility of small object features, making them undetectable. The sliding-window method (Fig. 4.2b) addresses this limitation by dividing the image into fixed-size tiles and processing each in full resolution. This allows the system to detect tiny objects, but significantly increases processing time. Moreover, the regular grid layout can cause larger objects to be fragmented across multiple tiles, reducing detection quality in multi-scale scenarios. To address this while maintaining real-time performance for tiny and multi-scale object detection in applications like mobile robotics, this work proposes integrating a learned ROI selection stage.

The author argues that with standard deep learning-based architectures, it is possible to improve detection quality over both Fig. 4.2a and Fig. 4.2b, while substantially reducing computation time compared to the sliding-window approach (Fig. 4.2b), potentially achieving real-time performance. Instead of processing detection windows selected on a regular grid, the proposed system introduces an additional segmentation-based `ROI Estimation Block`. This block operates on a low-resolution version of the input image and produces foreground regions, which are then converted into non-uniformly distributed detection windows by the `Detection Window`

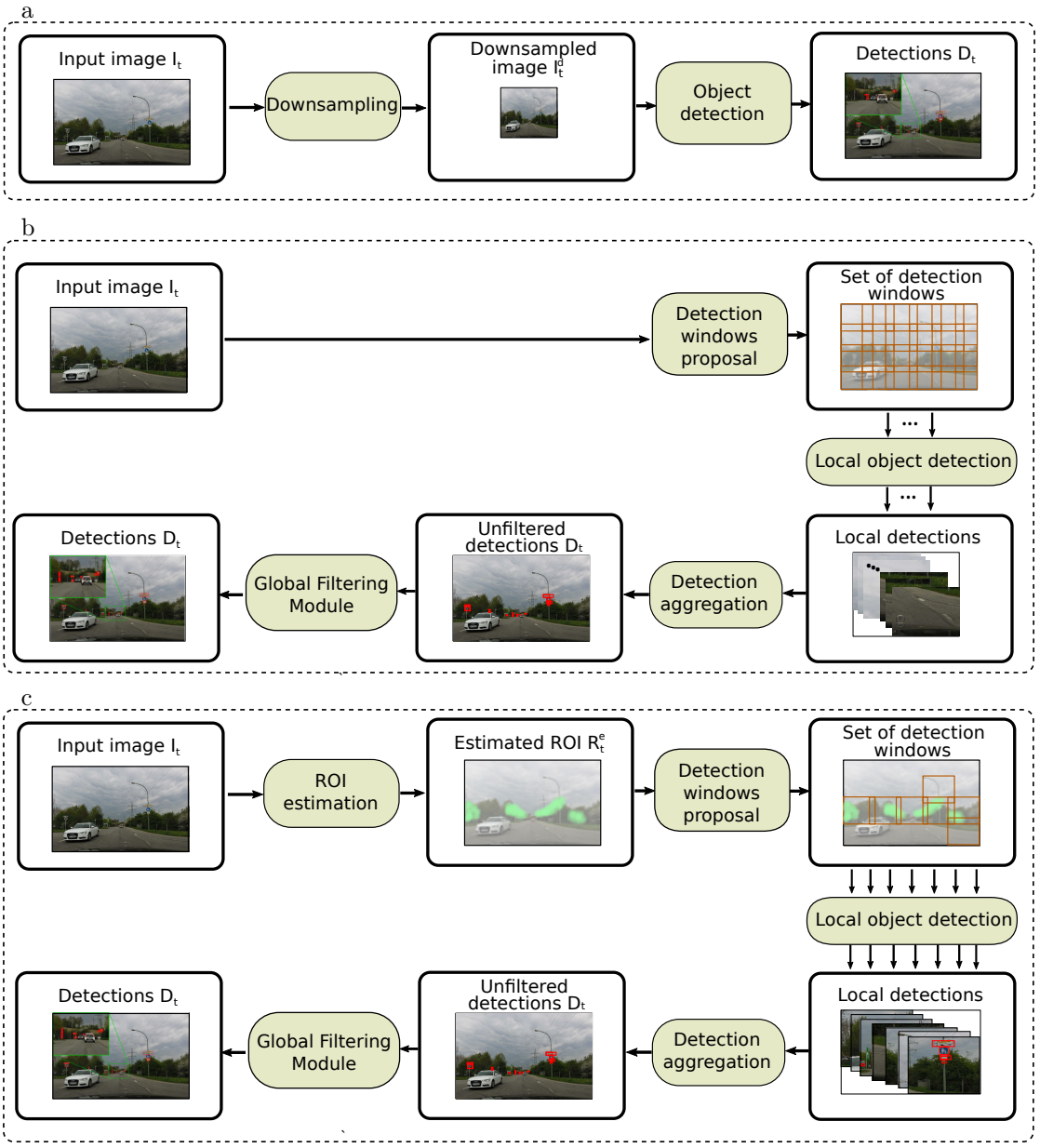


FIGURE 4.2: General architectures of the detection systems compared in this chapter: single detection with downsampling (a), often fast but misses tiny objects; naive sliding-window (b), which enhances tiny object detection recall but is slow and suboptimal due to the large number of empty windows processed; and the proposed ROI-based tiny object detection system (c), which introduces a learned `ROI Estimation` step that improves speed compared to the naive sliding-window while maintaining quality.

`Proposal Block`. To improve efficiency, the module also suppresses redundant windows, reducing the number of detector calls during inference. The remaining pipeline is shared between the sliding-window and the proposed system. Both approaches process multiple detection windows (`Local Object Detection` in Fig. 4.2b and Fig. 4.2c), then aggregate the detections and pass them through a `Global Filtering Module` to remove false positives caused by overlapping detection regions. In the following sections, two versions of the proposed system are discussed. These versions primarily differ in the `ROI Estimation`, `Detection Window Proposal`, and `Global Filtering` components. The differences between them are explained in detail later

in this chapter.

In contrast to existing window-based methods, which often rely on uniform grids [2, 38, 102, 116, 151] or supervised density and cluster annotations [34, 75, 157], the proposed system employs a lightweight semantic segmentation model to estimate regions of interest directly from full frames in low resolution. This approach enables class-agnostic, per-pixel objectness estimation, bypassing the need for manually labeled cluster or density maps. Unlike methods that treat ROI generation as a supervised regression or classification problem, the proposed segmentation-based estimation offers an interpretable and modular mechanism for selecting high-resolution tiles. While the overall framework integrates temporal cues through a separate `Prediction Branch`, this section focuses exclusively on the `ROI Estimation Branch`, which processes each frame independently. Compared to unsupervised or reinforcement learning-based methods [142, 155], the segmentation-based strategy remains fully deterministic, easier to train, and readily adaptable to new datasets. Moreover, since the segmentation backbone can be extremely lightweight, this component introduces minimal overhead, making it suitable for real-time or embedded applications.

In this chapter, the proposed ROI estimation-based tiny object detection system (Fig. 4.2c) is examined and compared comprehensively with two widely used baselines: the naive sliding-window approach (Fig. 4.2b) and single detection with downsampling (Fig. 4.2a). The goal is to validate Auxiliary Thesis #1 that estimating Regions of Interest (ROIs) using a deep neural network improves both detection quality and inference efficiency compared to the naive sliding-window strategy. To demonstrate this:

- the limitations of single-shot detection with downsampling are analyzed in the context of high-resolution images, where tiny objects are often missed due to reduced spatial detail,
- an ROI estimation-based detection framework is introduced and compared against the naive sliding-window baseline. To ensure a fair and controlled comparison, both approaches are integrated into a unified detection pipeline, keeping all other parameters and components consistent. The evaluation includes two variants of the proposed system, each differing primarily in their `Detection Window Proposal` strategy and `Global Filtering` mechanisms, in order to show consistent improvements across configurations,
- a comparative evaluation is performed against SAHI [2], a state-of-the-art open-source implementation of the sliding-window approach, to benchmark the system against an external baseline,
- a detailed experimental analysis of the `ROI Estimation` and `Window Proposal` modules is conducted to validate the architectural choices made in the design of the system.

The following sections build upon the author's previous works [67, 69, 70], with a specific focus on comparing the purely ROI estimation-based tiny object detection system against the sliding-window approach and single-shot detection with downsampling.

4.3 Proposed method

The proposed system addresses the challenges of tiny object detection in high-resolution images by improving detection quality over single-shot detectors while significantly reducing processing time compared to the sliding-window approach. This is achieved by limiting the number of detection windows processed during inference. The general architecture of the proposed object detection framework is illustrated in Fig. 4.2c. During inference, the system comprises the following core components:

- `Region of Interest (ROI) Estimation Network` - operates on a low-resolution version of the input image to suppress background-only regions,
- `Detection Window Proposal Module` - converts the binary segmentation mask into a set of non-redundant detection windows,
- `Local Object Detector` - runs independently on each selected window, followed by Non-Maximum Suppression (NMS) as a local post-processing step,
- `Detection Aggregation Module` - Transforms local detections into the coordinate system of the original image and merges outputs,
- `Global Filtering Block` - Filters out redundant detections in overlapping regions to produce the final result.

While the overall system also incorporates temporal cues in its full configuration, this section focuses exclusively on the segmentation-based ROI Estimation pipeline. Two versions of the ROI Estimation-based detection system are evaluated, both representing successive developments of the same core idea:

- TinyROI - a simplified system introduced in [67] that includes a basic `Detection Windows Proposal Module` and applies standard NMS in the `Global Filtering Block`,
- SegTrackDetect - an advanced system based on [69, 70], featuring a refined `Detection Windows Proposal Module` designed to support both tiny and multi-scale objects. This version integrates the novel `Overlapping Box Suppression (OBS)` method within the `Global Filtering Block`.

Detailed descriptions of each component are provided in the following sections. Differences between the two versions are discussed individually per module. Evaluation results clearly indicate which system variant is being referenced.

4.3.1 ROI Estimation Module

In high-resolution imagery, small and distant objects often shrink to just a few pixels when the input is downscaled for inference, making detection challenging. Increasing model size and

input resolution can improve performance but at a significant computational cost during both training and inference. To address this, the proposed system incorporates a `Region of Interest (ROI) Estimation Module` that operates on downsampled images to predict regions with a high likelihood of containing objects. The `ROI Estimation Network` suppresses background-only regions and highlights areas likely to contain objects. This segmentation-based prior guides the detection stage, enabling the main object detector to operate selectively at full resolution within the identified ROIs. As a result, fine-grained spatial details of tiny objects are preserved without the need to process the entire image at full resolution. To minimize computational overhead, this step utilizes shallow backbones and operates on downscaled inputs. Depending on the dataset, the `ROI Estimator` is implemented either using U^2-$Net^\dagger$ [104] for MTSD [35], or a UNet [114] with a ResNet18 backbone for all other datasets.

ROI generation is formulated as a binary segmentation task, where foreground pixels (objects) are labeled as 1 and background pixels as 0. In both TinyROI and SegTrackDetect, ground-truth segmentation masks are derived from object detection annotations. By default, the ground-truth binary masks are constructed as rectangular regions corresponding to bounding boxes, with extremely small objects (below 1 px) represented as single pixels. Section 4.5 further investigates alternative label generation strategies, including morphological expansion of ground-truth masks by a fixed number of pixels (N), where N is defined either globally based on image size or dynamically based on both image and object dimensions. Additionally, the influence of different input resolutions on the performance of the `ROI Estimation Module` is also analyzed.

During the analysis of optimal training settings for the `ROI Estimator` (detailed in Section 4.5.2), additional metrics were introduced to evaluate the relationship between standard segmentation performance and the correctness of the returned ROI regions. Unlike conventional semantic segmentation tasks, ROI estimation does not require precise pixel-wise masks or accurately defined object edges. Instead, the goal is to ensure that foreground regions sufficiently cover object locations, even with coarse resolution. To better reflect this objective, the following task-specific metrics were defined:

$$precision_{ROI} = \frac{|TP_{ROI}|}{|TP_{ROI}| + |FP_{ROI}|}, \tag{4.1}$$

$$recall_{ROI} = \frac{|TP_{ROI}|}{|TP_{ROI}| + |FN_{ROI}|}. \tag{4.2}$$

These metrics operate at the ROI level rather than at the pixel level. A true positive (TP_{ROI}) is registered if the estimated ROI mask for the i-th sample (P_i) intersects with the ground-truth binary mask generated for a single ROI ($GT_{o,i}$), such that $\sum GT_{o,i} \cap P_i > 0$. In this formulation, $precision_{ROI}$ reflects the proportion of proposed regions that actually contain relevant objects, closely related to the number of unnecessary detector runs, while $recall_{ROI}$ captures the ability to include all object-containing regions, corresponding to the number of missed detections due to incomplete ROI coverage.

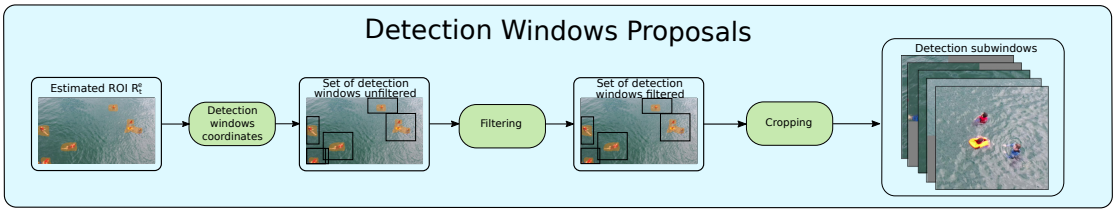


FIGURE 4.3: Architecture of the `Detection Windows Proposal Block` in the TinyROI [67] system.

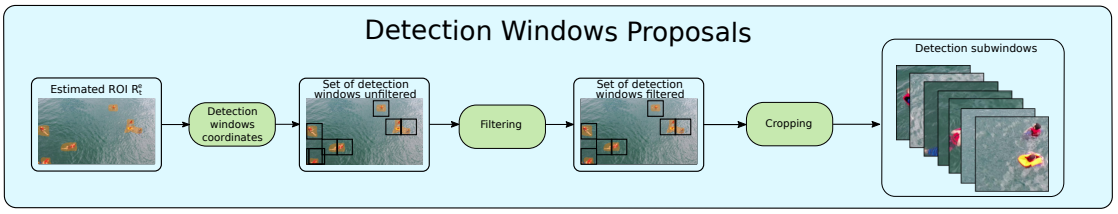


FIGURE 4.4: Architecture of the `Detection Windows Proposal Block` in the SegTrackDetect [69, 70] system.

4.3.2 Detection Windows Proposal

The `Detection Windows Proposal Block` transforms the segmentation mask produced by the `ROI Estimation Network` into a set of detection windows, based on the detector's input size, the original image, and the predicted foreground regions. Figures 4.3 and 4.4 illustrate the differences between the TinyROI and SegTrackDetect variants of this module. While the overall pipeline remains consistent, the specific implementations of detection window generation and cropping differ, as highlighted by the intermediate outputs. In both systems, the heatmap from the `ROI Estimation Module` is first binarized to identify individual ROI regions. The boundaries of these regions are used to generate an initial set of unfiltered detection windows. These are then aggregated and filtered to reduce redundancy (see "Detection Windows Filtered" in Figs. 4.3 and 4.4). The resulting refined subwindows are cropped from the original image and forwarded to the `Local Object Detection Module` for inference.

The key differences in the `Detection Windows Proposal Module` between TinyROI (Fig. 4.3) and SegTrackDetect (Fig. 4.4) lie in their handling of ROI size variability and window cropping strategy. In TinyROI, there is no mechanism to handle ROIs larger than the detector's input size, so the detector window must be chosen based on the expected maximum object size. This assumption works for static, closed-world datasets with sufficiently large detection windows. In TinyROI, detection windows are centered on each ROI and padded with gray pixels if they extend beyond the image boundaries (Fig. 4.3).

In contrast, SegTrackDetect employs a more adaptive and robust strategy tailored to real-time applications with significant object scale variation. It combines crop-and-resize and sliding-window techniques within the same ROI when necessary. For large ROI regions, a sliding-window is applied across the cropped area to preserve detection quality and mitigate the effects of downsampling. In cases where the ROI contains medium or large objects, moderate downsampling of the cropped region may be used, particularly when a single object exceeds the default detector

Algorithm 1 `Detection Windows Proposal`; parts of the algorithm used only in SegTrackDetect and not in TinyROI are highlighted in blue.

Require: $F_{ROI} \in (0,1)^{H_{low} \times W_{low}}$ ▷ Low-resolution fused ROI mask
Require: $I_{orig} \in \mathbb{R}^{H_{high} \times W_{high} \times 3}$ ▷ Original high-resolution RGB image
Require: (H_{det}, W_{det}) ▷ Detector input size
Ensure: $X \in \mathbb{R}^{B \times 3 \times H_{det} \times W_{det}}$ ▷ Batch of detection subwindows

Step 1: Extract ROI Contours
$F_{ROI}[F_{ROI} > 0] \leftarrow 255$ ▷ Threshold F_{ROI}
$\mathcal{C} \leftarrow \text{FindContours}(F_{ROI})$ ▷ Extract contours from F_{ROI}
$\mathcal{R} \leftarrow \text{ContoursToXYXY}(\mathcal{C})$ ▷ Convert contours to bounding boxes (xyxy)
$\mathcal{R} \leftarrow \text{RescaleToOriginal}(\mathcal{R}, H_{high}, W_{high})$ ▷ Scale bounding boxes to I_{orig} shape

Step 2: Generate Unfiltered Detection Windows
$\mathcal{W}_{unfiltered} \leftarrow \emptyset$ ▷ Initialize unfiltered detection windows set
for each ROI bounding box $r \in \mathcal{R}$ **do**
  if $\text{Size}(r) > (H_{det}, W_{det})$ **then** ▷ If ROI is larger than detector input size
    $\mathcal{W}_{unfiltered} \leftarrow \mathcal{W}_{unfiltered} \cup \{r\}$ ▷ Include original large ROI
    $\mathcal{W}_{unfiltered} \leftarrow \mathcal{W}_{unfiltered} \cup \text{SlidingWindows}(r, H_{det}, W_{det})$ ▷ Generate sliding windows
  else ▷ If ROI is smaller than detector input size
    $w \leftarrow \text{CenterWindow}(r, H_{det}, W_{det})$ ▷ Create centered detection window
    $\mathcal{W}_{unfiltered} \leftarrow \mathcal{W}_{unfiltered} \cup \{w\}$ ▷ Only add centered window
  end if
end for

Step 3: Filter Detection Windows
for $w \in \mathcal{W}_{unfiltered}$ **do**
  $s(w) \leftarrow \text{count}(r \in \mathcal{R})$ ▷ Compute score as number of r each w encapsulates
end for
Sort $(r, w) \in (\mathcal{R}, \mathcal{W}_{unfiltered})$ by $s(w)$ descending ▷ Prioritize windows covering multiple ROIs
$\mathcal{W}_{filtered} \leftarrow \emptyset$ ▷ Initialize filtered detection windows set
for each (r, w) in sorted $(\mathcal{R}, \mathcal{W}_{unfiltered})$ **do**
  if r is not enclosed by any $w' \in \mathcal{W}_{filtered}$ **then** ▷ Check if ROI is not already covered
    $\mathcal{W}_{filtered} \leftarrow \mathcal{W}_{filtered} \cup \{w\}$ ▷ Add non-redundant detection window
  end if
end for

Step 4: Generate Batch of Detection Subwindows
$B \leftarrow |\mathcal{W}_{filtered}|$ ▷ Number of final detection windows
$X \leftarrow \emptyset$ ▷ Initialize output batch
for each $w \in \mathcal{W}_{filtered}$ **do**
  $x \leftarrow \text{CropAndResize}(I_{orig}, w, H_{det}, W_{det})$ ▷ Crop and resize subwindow from I_{orig}
  $X \leftarrow X \cup \{x\}$ ▷ Add subwindow to batch
end for
return $X \in \mathbb{R}^{B \times 3 \times H_{det} \times W_{det}}$ ▷ Final batch of detection subwindows

window size. When the ROI is smaller than the detector's input dimensions, a single window is centered on the region, following the same approach as in TinyROI. This hybrid strategy enables SegTrackDetect to operate with smaller detection windows without losing coverage, as large or complex ROIs are dynamically processed using multiple subwindows or scaled appropriately. To avoid passing empty or invalid regions to the detector, window positions are also adjusted to remain within image boundaries, eliminating the need for gray padding used in TinyROI. The discussion of TinyROI highlights the design decisions and evolution that led to the adaptive, multi-scale approach of the final SegTrackDetect system.

Both systems use the same filtering strategy, with naive filtering with sorting adopted as the default configuration. This approach is detailed in Alg. 1, which describes the procedure used in SegTrackDetect. Section 4.5.3 evaluates this component using TinyROI to compare alternative strategies, no filtering and naive filtering without sorting, in terms of runtime and detection quality. The results support the use of the default filtering method in both TinyROI and SegTrackDetect. While Algorithm 1 reflects the more advanced and adaptive pipeline of SegTrackDetect, TinyROI relies on a simplified version based on fixed-size, center-aligned windows with

gray padding when needed.

4.3.3 Local Object Detection

This component is responsible for performing object detection within the proposed subregions of the original image. Rather than applying the detector to the full resolution frame, detection is executed only on selected ROIs, significantly reducing computational cost while preserving fine spatial details. Both TinyROI and SegTrackDetect use lightweight YOLO models but differ in their strategies for window sizing, scaling, and runtime optimization.

In the TinyROI system, object detection is carried out within fixed-size cropped regions centered on each proposed ROI. This design assumes that a single detection window can adequately capture each region of interest, simplifying the pipeline by eliminating the need for dynamic resizing or multi-scale handling. However, the approach incurs higher computational cost per window and lacks flexibility when dealing with large or densely packed ROIs. SegTrackDetect introduces several optimizations to improve detection efficiency and scalability, particularly for real-time applications and scenarios involving variable object sizes. SegTrackDetect allows for using lower input sizes due to the adaptive handling of large ROIs in the `Detection Windows Proposal Block`, which ensures full ROI coverage without compromising detection quality.

In all experiments presented in this Chapter, MTSD uses YOLOv4 [12] with an input resolution of 960×960 pixels, while for all other datasets the detector is replaced with YOLOv7 Tiny [136], a lightweight architecture with only 6 million parameters and an input size of 512×512 pixels. Despite the availability of newer YOLO variants such as YOLOv10 [135] and YOLOv12 [127], YOLOv7 Tiny was selected due to its balance between runtime efficiency, architectural stability, and consistent performance across tasks. While some newer models offer fewer parameters, they often require more operations (FLOPs) and are harder to train or deploy. Moreover, as shown in Tab. 5.1 and Tab. 5.2, the largest YOLOv10 variant (YOLOv10x) does not exhibit a significant quality advantage over YOLOv7-e6e.

To train both systems, datasets are generated by cropping annotated full-resolution images into detection windows, adjusting the bounding boxes to the local coordinate system. Negative (empty) windows are included to discourage false positives in background regions. A scale-augmentation procedure is also applied to enrich the training data with multiple object size variants, mitigating potential mismatches between training and test distributions.

During inference, preliminary post-processing is applied using Non-Maximum Suppression (NMS) within each detection window. The final Detection Aggregation step maps detected bounding-boxes back to the original image space. In SegTrackDetect, this transformation includes a scaling factor if the crop-and-resize method was used, while in TinyROI, no such scaling is needed due to fixed-resolution processing.

4.3.4 Global Filtering Module

After aggregating detections from multiple sub-windows, a second-stage filtering step is required to eliminate redundant or partial detections. These typically arise from objects being only partially visible in individual windows, especially in overlapping regions. In TinyROI, this `Global Filtering` step uses standard Non-Maximum Suppression (NMS), identical to the method applied during local filtering. An evaluation of alternative filtering strategies is presented in Section 4.5.3, which justifies the use of NMS and its hyperparameters in the final TinyROI design.

In contrast, SegTrackDetect introduces a more tailored solution, `Overlapping Box Suppression (OBS)`, which significantly enhances the robustness of the `Global Filtering` process. The design and evaluation of `OBS`, including comparisons to NMS, are detailed in Chapter 6. The focus of this Chapter, however, is to demonstrate that incorporating ROI estimation improves both detection quality and speed relative to the sliding-window approach. To enable a fair comparison, sliding-window implementations are included in both TinyROI and SegTrackDetect. This ensures that any differences in the `Global Filtering Blocks` do not affect the isolated analysis of ROI estimation versus sliding-window performance.

4.4 Experimental results

This section presents the main results of TinyROI and SegTrackDetect, each compared against their respective sliding-window baselines. To ensure fairness, the sliding-window methods were integrated directly into the proposed pipelines, as illustrated in Fig. 4.2b, minimizing architectural differences across setups. Both proposed systems are evaluated in their final, optimized configurations, as described in Section 4.3 and quantitatively justified through the ablation study in Section 4.5. Both frameworks are further compared with the open-source framework SAHI [2], omitting the single-shot baseline due to training issues discussed later. All sliding-window experiments use an overlap fraction of 0.01 to minimize discrepancies across frameworks. For each dataset, identical pre- and post-processing steps and consistent hyperparameter configurations are applied - such as the IoU thresholds used in the `Global Filtering` stage (`OBS` for SegTrackDetect, NMS for TinyROI). The results demonstrate that incorporating a learned region estimation step into the focus-and-detect pipeline not only improves inference speed compared to the traditional sliding-window approach but also can yield superior detection accuracy. Quantitative results are reported following the evaluation protocol and the relative size definition introduced in [67].

The comparisons in this section are designed to directly validate Auxiliary Thesis #1, while a broader comparison including all three detection strategies from Fig. 4.2 is provided in Section 4.5.1.

TABLE 4.1: Quantitative results comparing TinyROI and SegTrackDetect frameworks in various configurations with SAHI [2] using three datasets: MTSD, SeaDronesSee and ZebraFish. All experiments were performed on NVIDIA RTX A6000 and evaluated with the protocol proposed in [67]. The confidence threshold of 0.1 was used for all experiments.

Dataset	Framework	Method	AP	AP_{50}	AR	AP_{μ}	AP_{vt}	AP_{t}	AP_{s}	AP_{m}	AP_{l}	FPS
MTSD	SAHI [2]	sliding-window	35.5	77.1	46.7	13.9	34.6	43.6	40.0	37.3	34.3	1.2
	TinyROI	sliding-window	45.9	**82.2**	**56.9**	**15.7**	44.4	52.7	53.2	47.6	**39.5**	3.5
	SegTrackDetect	sliding-window	43.4	75.1	54.3	13.3	44.7	51.3	47.8	36.1	26.5	4.1
	TinyROI	estimation	**47.4**	80.8	56.6	14.1	**45.0**	**53.7**	**55.8**	**49.4**	36.7	12.5
	SegTrackDetect	estimation	46.2	77.6	56.6	15.0	43.9	53.4	54.4	48.9	35.7	**14.9**
SeaDronesSee	SAHI [2]	sliding-window	42.1	77.5	53.5	-	**33.6**	42.4	49.4	18.6	0.1	1.7
	TinyROI	sliding-window	40.9	76.4	52.6	-	31.5	37.9	50.3	16.4	0.1	6.0
	SegTrackDetect	sliding-window	42.7	79.4	52.5	-	33.2	40.8	50.9	17.8	0.1	17.3
	TinyROI	estimation	49.9	83.4	59.3	-	31.9	46.6	58.1	34.3	36.6	20.7
	SegTrackDetect	estimation	**52.6**	**83.5**	**60.9**	-	32.0	**51.9**	**61.6**	**36.4**	**67.5**	**44.5**
ZebraFish	SAHI [2]	sliding-window	18.2	52.3	33.1	-	-	2.1	15.8	21.0	-	1.6
	TinyROI	sliding-window	21.3	56.3	37.5	-	-	4.2	17.8	25.1	-	2.4
	SegTrackDetect	sliding-window	22.1	58.0	37.9	-	-	7.3	18.7	25.3	-	18.0
	TinyROI	estimation	52.7	78.1	60.1	-	-	20.1	54.3	51.9	-	41.5
	SegTrackDetect	estimation	**80.7**	**95.7**	**83.8**	-	-	**41.2**	**76.1**	**87.7**	-	**78.0**

4.4.1 Quantitative Results

The primary results for all three datasets are summarized in Tab. 4.1, reporting overall detection quality metrics (AP, AP_{50}, and AR_{100}), size-specific Average Precision values, and inference speed in frames per second (FPS). These metrics are also presented in Fig. 4.5 to provide a clearer and more intuitive comparison between methods.

In the MTSD dataset, both ROI estimation and sliding-window methods achieve relatively high average precision (AP), with TinyROI (`ROI Estimation`) reaching the top score of 47.4%. Interestingly, sliding-window methods perform unusually well compared to the other datasets; TinyROI sliding-window achieves an AP of 45.9%, only slightly below its estimation variant. This can be explained by the large detection window size (960×960) used across all MTSD methods. Such a large window increases the chance of fully capturing even large or mid-size objects in a single tile, thereby reducing the dependency on precise region selection. The high performance of all sliding-window methods supports this interpretation. However, SegTrackDetect with `ROI Estimation`, despite being more efficient (14.9 FPS), slightly underperforms in terms of AP (46.2%) compared to TinyROI. This is attributed to suboptimal detector training, which involved the use of crops and downsampled images, reducing the ability to accurately recover large object quality, clearly visible in the relatively low AP_l across all methods. This represents a limitation of the detector itself rather than the framework, as evidenced by SegTrackDetect's strong performance for larger objects on the SeaDronesSee and ZebraFish datasets. Furthermore, it is important to note that SegTrackDetect's key improvements primarily target three areas: handling large ROIs that exceed the detector's input size, filtering out false positive partial detections, and enhancing inference efficiency. On MTSD, where detection windows are already large (960×960) and typically contain full objects, the benefits of partial detection filtering and large ROI management have reduced effectiveness. For fairness, the same detector was used in both TinyROI and SegTrackDetect; however, SegTrackDetect could operate with smaller input sizes thanks to its ROI handling strategy, which on other datasets has been shown

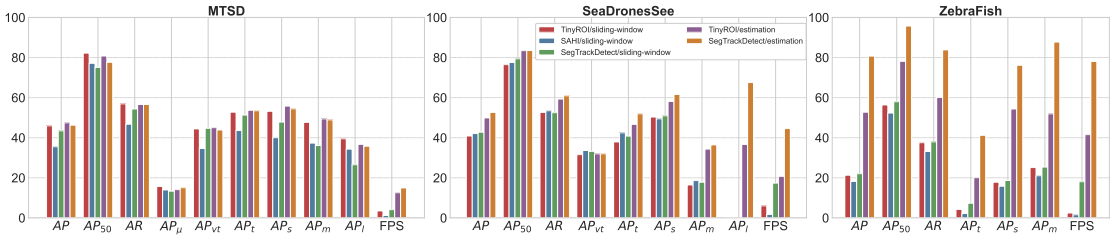


FIGURE 4.5: Quantitative comparison of multiple sliding-window methods with the proposed estimation-based approaches in TinyROI and SegTrackDetect frameworks. Quality and speed was measured for three datasets: MTSD, SeaDronesSee and ZebraFish. All experiments were performed on NVIDIA RTX A6000 and evaluated with the protocol proposed in [67]. The confidence threshold of 0.1 was used for all experiments.

to improve inference speed without compromising detection quality. Since large objects cannot be fully restored due to detector limitations, no framework-level improvement can overcome that bottleneck. Additionally, the SegTrackDetect relative improvement in FPS on MTSD is the smallest among all datasets, because the large input size of the detector leads to considerable background being processed regardless of ROI selection. In contrast, on SeaDronesSee and ZebraFish, SegTrackDetect achieves significantly higher speedups by leveraging smaller input sizes and applying efficient ROI management, which reduces redundant background processing and maximizes inference speed.

Although SeaDronesSee has image resolutions similar to MTSD, it uses a much smaller detection window size (512×512), which limits the effectiveness of sliding-window methods. These approaches struggle with large objects that get fragmented across tiles and also waste computation in the background, leading to modest performance (AP 41-43%) and poor results for medium and large objects. SegTrackDetect with `ROI Estimation` overcomes these issues by accurately identifying regions of interest and adapting its processing strategy. It achieves this through dedicated large ROI handling logic, which combines sliding-window and crop-and-resize strategies to manage oversized ROIs. The method splits large ROIs into smaller overlapping tiles while preserving object integrity. This approach enables the use of small detector input sizes without losing the context of large objects, effectively balancing detection accuracy and computational efficiency. Additionally, partial detection filtering further boosts precision by suppressing redundant detections near tile boundaries. As a result, SegTrackDetect reaches an AP of 52.6%, AR of 60.9%, and strong performance across all object sizes (e.g., $AP_l = 67.5\%$), all while running at 44.5 FPS - over 25 times faster than SAHI. These results demonstrate how smart ROI management and filtering enable efficient and accurate detection, even for large objects, without relying on large input sizes or excessive background processing. In both TinyROI and SegTrackDetect, the estimation-based approach outperforms the sliding-window baseline, thereby confirming Auxiliary Thesis #1.

The ZebraFish dataset, while not the most challenging, represents a relatively simple case where the smallest detection windows are used. This setup highlights the importance of choosing detector window sizes according to the dataset's characteristics. With a strategy that effectively handles large ROIs, smaller windows can be employed without sacrificing the detection of large outliers. Sliding-window methods struggle here, yielding low AP scores between 18% and 22% and almost no large object detections. In contrast, SegTrackDetect with `ROI Estimation`

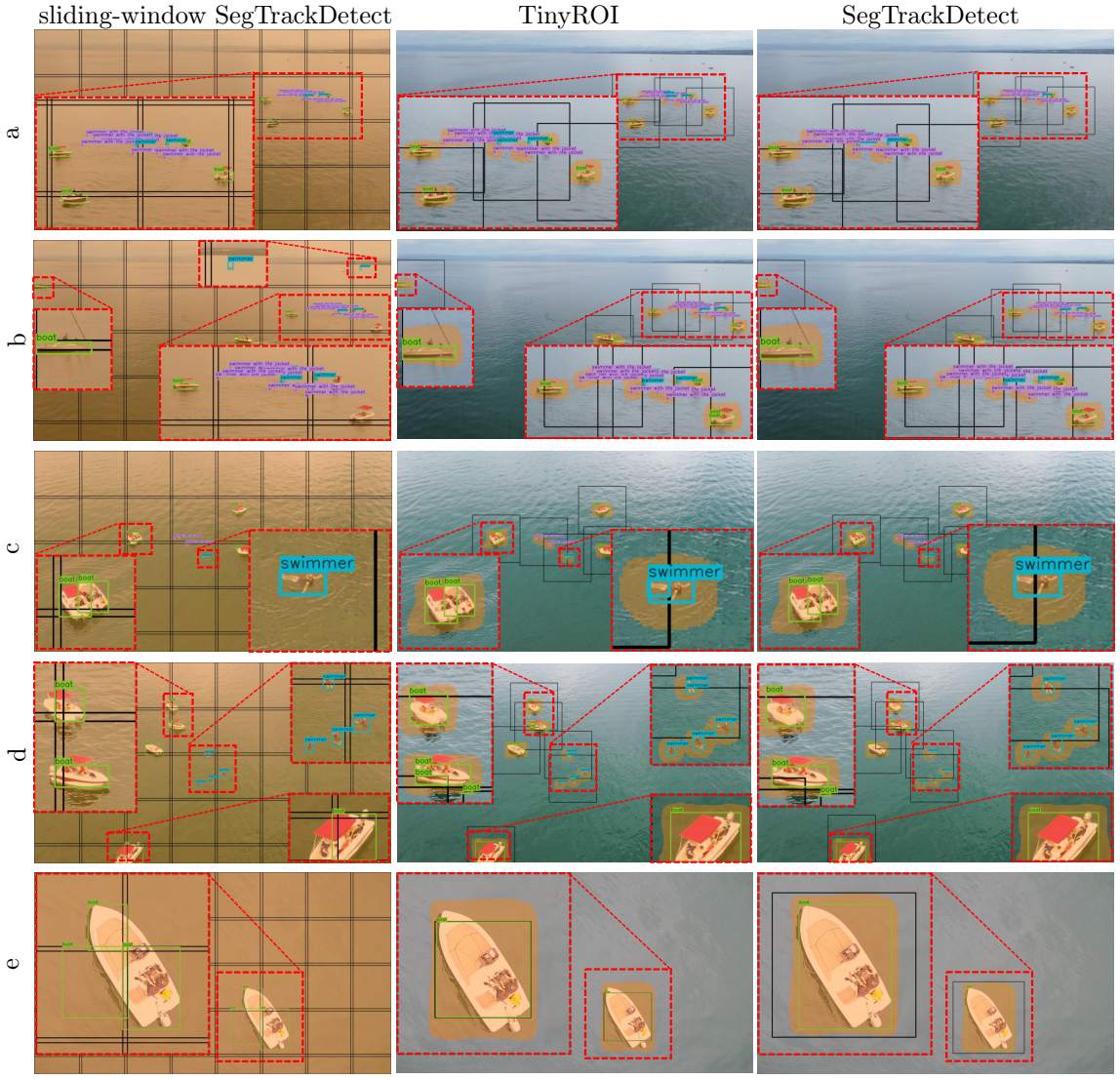


FIGURE 4.6: Qualitative comparison of the sliding-window approach (within the SegTrackDetect framework) with TinyROI and the SegTrackDetect system. All visualizations use a confidence threshold of 0.1 for consistency.

achieves an AP of 80.7%, AP_{50} of 95.7%, and 83.8% AR, significantly outperforming baselines like TinyROI (52.7% AP) while running at 78 FPS.

4.4.2 Qualitative Results

Figure 4.6 presents a qualitative comparison between the SegTrackDetect system, the initial estimation-based variant (TinyROI), and the best-performing sliding-window method implemented within the SegTrackDetect framework. All examples are taken from the SeaDronesSee dataset, and a uniform confidence threshold of 0.1 is applied across all methods to ensure a consistent comparison. In some cases, particularly when objects are sparse and well-separated, as shown in Fig. 4.6a, all three methods achieve similarly high detection quality. However, the limitations of the sliding-window approach become more evident when objects are fragmented across uniform tiles, as illustrated in Fig. 4.6d-e. These particular results were obtained before

the integration of the `Overlapping Box Merging (OBM)` algorithm into the `Global Filtering Block`. As a result, fragmented detections are sometimes incorrectly filtered (e.g., Fig. 4.6d), leading to imprecise bounding boxes for larger boats, or fail to be merged into a single detection (Fig. 4.6e). The latter clearly demonstrates the necessity of a dedicated merging mechanism in window-based pipelines, which is addressed later in Chapter 6. The TinyROI variant does not include the large ROI handling strategy used in SegTrackDetect, and as ROI sizes increase, its detection quality degrades more noticeably (see Fig. 4.6e). Some of the qualitative results also highlight the benefits of the advanced `Global Filtering Module` used in SegTrackDetect, particularly its ability to suppress partial false positives, as seen in the rightmost column of Fig. 4.6 (rows c and d) - this is also analyzed in more detail in Chapter 6. Additionally, the exhaustive nature of the sliding-window method occasionally produces false positives in background regions, as shown in row b. In contrast, the segmentation-based ROI selection in SegTrackDetect acts as a filter, helping to prevent such errors. Still, the number of false positives for the sliding-window method remains low in the examples shown, which confirms the effectiveness of the adopted training strategy that included background-only tiles to discourage false-positive detections in case of false-positive ROIs.

Across datasets, `ROI Estimation` consistently improves both accuracy and inference time, especially where object scale variability or density make naive sliding-window methods ineffective. Sliding-window approaches can still perform relatively well when detection windows are large enough to contain entire objects, as in MTSD, but this comes at the cost of slower inference and poorer scalability. Overall, SegTrackDetect's learned region selection demonstrates clear advantages, particularly on challenging datasets, confirming that deep `ROI Estimation` enhances detection precision and enables faster inference in high-resolution, small-object detection scenarios. These findings directly prove Auxiliary Thesis #1: "Estimating ROIs using a deep neural network model enhances both detection quality and inference speed compared to the naive sliding-window approach". The slight drop in AP on MTSD is likely due to training data limitations rather than any deficiency in the framework itself.

4.5 Ablation study

This ablation study documents the key experiments conducted during the development of TinyROI and SegTrackDetect, with the goal of optimizing the ROI estimation-based pipeline and refining its architectural components. The study begins with a comparison of the three detection strategies illustrated in Fig. 4.2, using the MTSD dataset and the initial TinyROI framework. It then analyzes design choices in both systems.

For TinyROI, experiments focused on how to best train the `ROI Estimator`, including input resolution and binary mask label generation, as well as evaluating ROI filtering and detection filtering strategies in terms of accuracy and efficiency. In SegTrackDetect, the focus shifted to methods for handling large ROIs. Together, these analyses informed the design of the final versions of both frameworks, which were used to report the main results in the previous section.

TABLE 4.2: Comparison of detection quality, inference speed (measured on a GeForce RTX 3080 Ti), and the average number of detector calls (detection windows) per image (DW). Three methods are evaluated: single-shot detection with downsampling (SD), sliding-window (SW), and the proposed approach (TinyROI). All methods use the same NMS settings and a detector input resolution of 960^2 px with a batch size of 1. Results are reported on the MTSD validation set. AP and AR metrics are presented in separate subtables for clarity.

(a) Detection performance - AP metrics

Method	**DW**	**FPS**	**AP**	**AP$_{50}$**	**AP$_{75}$**	**AP$_{\mu}$**	**AP$_{vt}$**	**AP$_{t}$**	**AP$_{s}$**	**AP$_{m}$**	**AP$_{l}$**
SD	**1**	**17.4**	33.8	60.1	34.5	0.2	10.4	45.2	**58.6**	**63.2**	**73.8**
SW	14.7	1.2	46.2	**82.6**	46.9	**15.9**	44.7	53.0	53.6	47.9	39.5
TinyROI	1.8	7.9	**47.7**	81.3	**51.4**	14.4	**45.4**	**53.9**	56.1	50.4	36.9

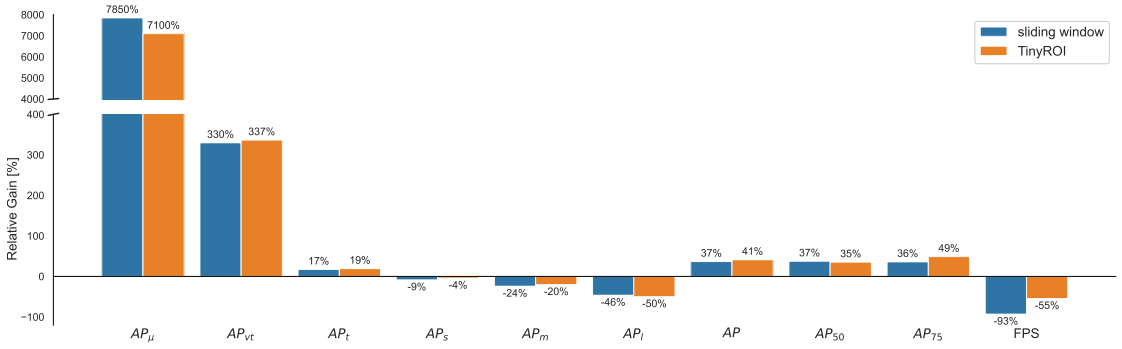
(b) Detection performance - AR metrics

Method	**DW**	**FPS**	**AR$_{1}$**	**AR$_{10}$**	**AR$_{100}$**	**AR$_{\mu}$**	**AR$_{vt}$**	**AR$_{t}$**	**AR$_{s}$**	**AR$_{m}$**	**AR$_{l}$**
SD	**1**	**17.4**	40.7	47.1	47.2	1.1	22.9	58.3	**67.6**	**69.0**	**75.3**
SW	14.7	1.2	51.4	**58.5**	**58.5**	**25.1**	**55.2**	**62.0**	62.4	53.4	40.1
TinyROI	1.8	7.9	**51.6**	58.2	58.2	19.9	54.1	61.6	63.0	56.7	37.6

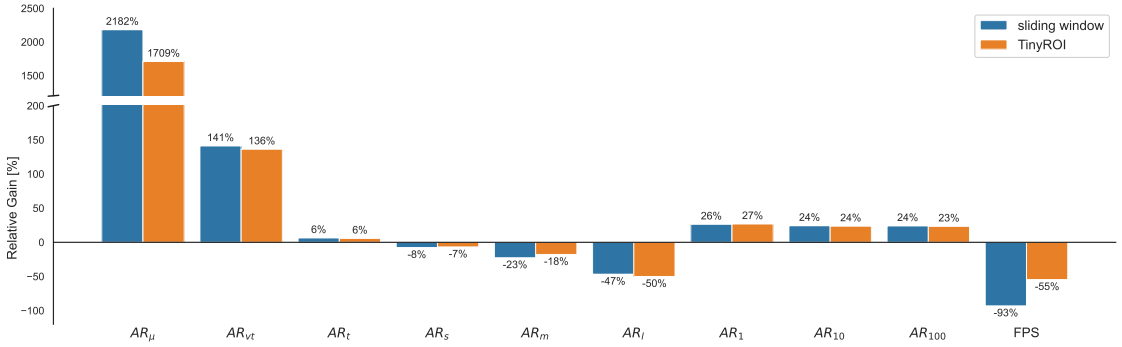
4.5.1 Comparison of Detection Strategies

Table 4.2 summarizes the final evaluation of the TinyROI system in comparison to two baseline approaches: single-shot detection with downsampling (SD) and the traditional sliding-window method (SW). For a fair comparison, all three approaches employ the same YOLOv4 detector trained on the MTSD dataset, using input samples at 960^2 pixels composed of both downsampled images and random crops. The single-shot approach involves resizing the entire input image to 960^2 pixels, performing detection, then upscaling the resulting bounding boxes to the original resolution. The sliding-window method uses overlapping detection windows, with an overlap fraction of 5%, and processes each window individually at full resolution. All methods share the same Non-Maximum Suppression (NMS) parameters: an IoU threshold of 0.45 and a confidence threshold of 0.01 with weighted merging. In addition to the detection metrics, Tab. 4.2 reports the average number of detector invocations (detection windows) per image and inference speed measured in frames per second (FPS). Importantly, the inference timing includes not only the detector runtime but also NMS execution, `ROI Estimation` and `Global Filtering`.

Described results demonstrate that TinyROI achieves detection accuracy nearly equivalent to the sliding-window approach while delivering a substantial speedup, approximately sevenfold faster inference. Specifically, TinyROI reaches a COCO-style mean Average Precision (AP) of 47.7%, outperforming the single-shot downsampling method by nearly 15 percentage points and exceeding the sliding-window method by 1.5%. The detailed AP scores at IoU thresholds 0.50 (AP_{50}) and 0.75 (AP_{75}) are 81.3% and 51.4%, respectively. These represent significant gains over the single-shot method and closely match the sliding-window performance. The relative improvements in both AP and AR metrics when switching from single-shot detection to window-based approaches (represented by the relative gain) are visualized in Fig. 4.7. Notably, both TinyROI and sliding-window strategies yield substantial enhancements in detecting micro, very tiny, and tiny objects. For example, the AP for micro-objects (AP_{micro}) improved by 7850% with



(A) Average Precision (AP) metrics



(B) Average Recall (AR) metrics

FIGURE 4.7: Relative metric gain caused by replacing the single-shot detection with window-based full resolution approaches - simple sliding-window and TinyROI.

the proposed method compared to single-shot detection, demonstrating the efficacy of focusing detection at full resolution on selected ROIs. As shown in Tab 4.2 micro scale objects are practically undetectable in a single-shot approach. While TinyROI and sliding-window deliver nearly identical AP_{vt} and AP_t scores, TinyROI incurs a slight drop relative to sliding-window in some very small object categories. This minor difference is likely attributable to the ROI selection, which while efficient, may occasionally omit relevant regions captured by exhaustive sliding-windows. As illustrated in Fig. 4.8 and Fig. 4.9, the `ROI Estimation Network` sometimes fails to highlight areas containing very tiny objects. This limitation is likely due to the relatively deep architecture of the estimation backbone, which may overlook low-level cues associated with minuscule targets. This observation was an important factor in the development of the SegTrackDetect system, where more recent versions employ shallower `ROI Estimation Networks` designed to better preserve fine-grained spatial details and improve sensitivity to tiny objects. Despite this occasional omission, the overall impact is mitigated. In real-world scenes such as traffic environments, relevant objects, especially traffic signs, often appear in spatial clusters. As a result, even if a specific object is not directly detected by the `ROI Estimator`, its neighboring context is likely to be included, enabling the detector to recover most of the missed content during inference.

For larger object categories (small, medium, and large), the single-shot detection method achieves the highest AP and AR scores among the three. This is presumably because both the sliding-window and TinyROI methods process images at a single scale, which may be suboptimal for

FIGURE 4.8: Qualitative results comparing ground-truth annotations with single-shot detection using downsampling, and window-based approaches: naive sliding-window and the proposed TinyROI system incorporating a learned `ROI Estimation Module`. Black rectangles indicate detection windows, while orange regions in the TinyROI row represent returned ROIs. Single-shot detection often misses tiny objects, which can be recovered by full-resolution inference in both the sliding-window and TinyROI approaches. However, the sliding-window method introduces more false positives, likely due to object fragmentation and complex backgrounds. In contrast, the learned `ROI Estimation Module` in TinyROI acts as an initial filtering step, helping to reduce false positives. While this may occasionally result in missed detections, the significantly reduced processing time justifies the trade-off.

larger objects spanning multiple detection windows. Larger objects are prone to being fragmented across window boundaries, and despite the use of NMS, the absence of more sophisticated merging or filtering methods can lead to duplicated or missed detections, reducing overall precision. Additionally, it is suspected that the training process (using both downscaled images and full-resolution crops) hindered the detection of larger objects, especially in such unevenly distributed classes as in MTSD. This setup introduces disparity in object scales, as the dataset includes a wide variety of image sizes. In SegTrackDetect, the training procedure was revised to exclude downscaled images. Instead, full-resolution crops are combined with slightly larger and smaller regions, to mitigate the impact of size distribution imbalance between the training and test subsets.

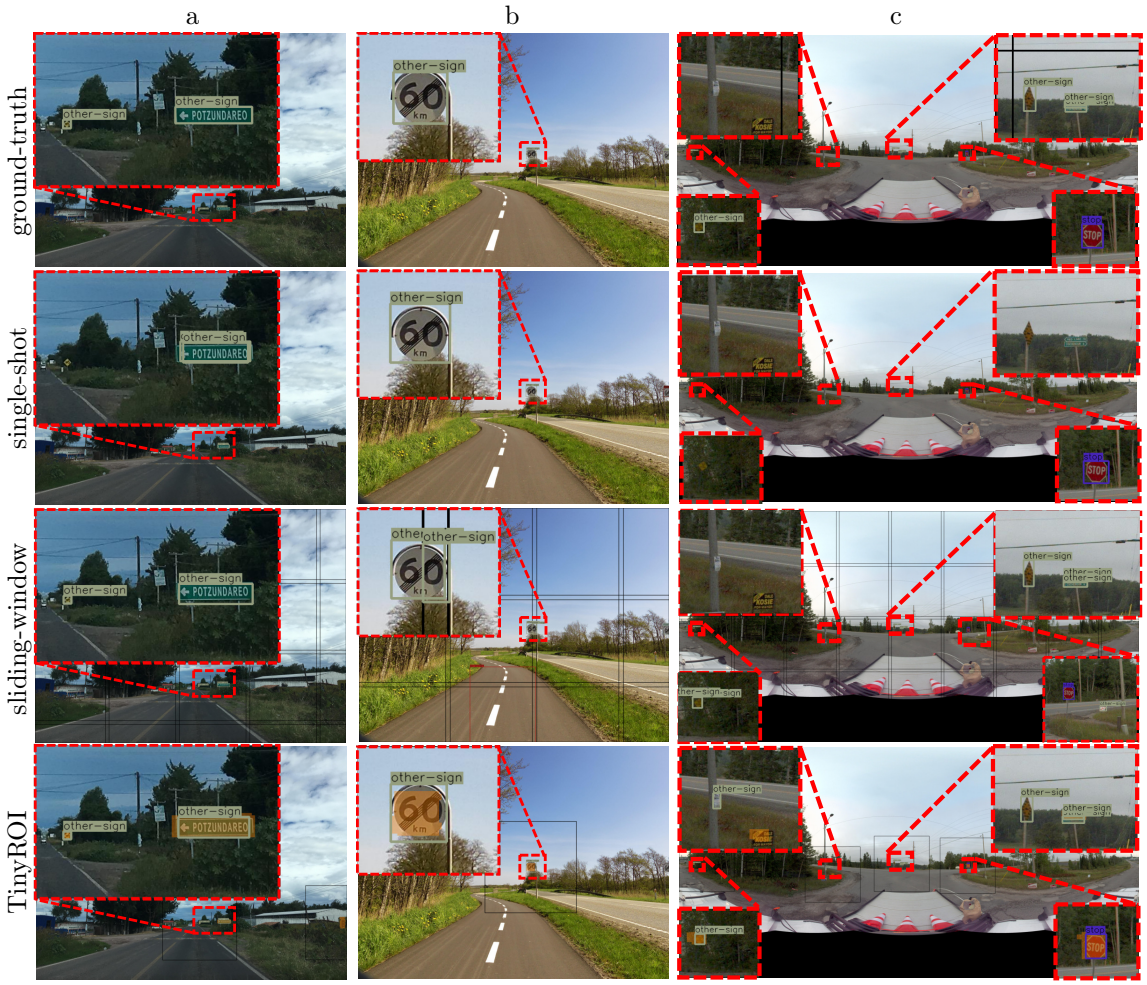


FIGURE 4.9: Qualitative results comparing ground-truth annotations with single-shot detection using downsampling, and window-based approaches: naive sliding-window and the proposed TinyROI system incorporating a learned `ROI Estimation Module`. Black rectangles indicate detection windows, while orange regions in the TinyROI row represent returned ROIs. Single-shot detection often misses tiny objects, which can be recovered by full-resolution inference in both the sliding-window and TinyROI approaches. However, the sliding-window method introduces more false positives, likely due to object fragmentation and complex backgrounds. In contrast, the learned `ROI Estimation Module` in TinyROI acts as an initial filtering step, helping to reduce false positives. While this may occasionally result in missed detections, the significantly reduced processing time justifies the trade-off.

Figures 4.8 and 4.9 present qualitative results for the TinyROI system on the MTSD dataset, comparing its outputs to the ground-truth, single-shot detection, and the sliding-window approach. Both figures show results from the aforementioned systems; however, due to the complexity of the scenes and the high resolution of the images, the visualizations were split into two separate figures for clarity. To avoid ambiguity, all examples presented in Figs. 4.8, 4.9 were generated using single-label NMS. As illustrated, single-shot detection consistently fails to detect tiny objects. In Fig. 4.8a, no objects are detected in the zoomed-in region, whereas both the sliding-window and TinyROI approaches successfully recover all of them. A similar situation is shown in Fig. 4.8b, where larger objects are detected correctly in the single-shot result, but three smaller objects are completely missed. Although both the sliding-window and TinyROI detect these missed objects, the sliding-window method introduces several false positives - highlighting

the need for a more robust filtering strategy (discussed in Chapter 6), as standard NMS is unable to handle such fragmentary detections effectively. Similar false positives are observed in Figs 4.8e and 4.9c, where the sliding-window yields more noisy detections than TinyROI. In Fig. 4.8c, a false positive (building) appears due to a lack of contextual information, an issue mitigated in TinyROI by placing windows more centrally within relevant ROIs, which better preserves surrounding context. Figure 4.8c also shows multiple overlapping detections of the same object class ("other-sign"), where small object sizes lead to insufficient suppression by IoU-based NMS, as visually overlapping small objects often yield low IoU values that fail to trigger suppression. In Figs. 4.9a and 4.9b, small objects missed by single-shot detection are accurately detected using both window-based methods. Finally, Fig. 4.9c shows that single-shot detection captures only one stop sign, whereas both window-based approaches successfully detect more.

Overall, TinyROI produces fewer false positives than the sliding-window and achieves similar or better detection quality in terms of false positives, while requiring significantly fewer detection windows, highlighting a clear advantage in efficiency and robustness. From these experiments, several key insights were gained: better filtering techniques are necessary to improve detection quality; shallower networks might be more suitable for the `ROI Estimator`; training should avoid using downsampled images to maintain true object sizes and aspect ratios; and more effective methods for handling large ROIs are needed, since increasing the window size often results in detection windows containing predominantly background. All these improvements have been incorporated into the SegTrackDetect framework, presented in the previous section, which demonstrates that the ROI estimation-based system achieves higher detection quality and speed than the sliding-window approach.

4.5.2 Training of the ROI Estimator

During the development of the TinyROI detection system, optimal training settings for the `ROI Estimator` were established. This configuration was subsequently adopted in the SegTrackDetect system. All experiments discussed in this section employ the ROI-level metrics introduced in equations 4.1 and 4.2, alongside standard segmentation metrics. These ROI-level metrics focus primarily on assessing the coverage of regions of interest rather than the precision of the segmentation masks.

Image Resolution To determine the optimal training and inference resolutions for the `Estimator`, three variants of the U^2-$Net^\dagger$ model were trained for 120 epochs using the *Adam* optimizer (learning rate = 0.001) and a batch size of 16. The experiments were conducted at three different resolutions: 144^2, 288^2, and 576^2 pixels. Regardless of the input resolution used during training, validation was performed at all three resolutions to assess the model's scalability and generalization across different input sizes. For each model, results reported in Tab. 4.3 correspond to the checkpoint selected based on the highest recall of ROI generation (r_{ROI}) at the resolution the model was trained on. This metric was prioritized because, within the scope of this work, achieving full coverage of all objects of interest is more critical than minimizing false positives that could trigger additional detector runs.

TABLE 4.3: Segmentation and ROI-level metrics for `Estimator` models trained with three input resolutions ($\text{train}_{\text{res}}$ 144^2, 288^2, 576^2) and evaluated at each of them (val_{res}). Results from the TinyROI detection system with the Mapillary Traffic Sign Dataset.

$\textbf{Train}_{\textbf{res}}$	$\textbf{Val}_{\textbf{res}}$	$\textbf{iou}_{\textbf{FG}}$	$\textbf{p}_{\textbf{FG}}$	$\textbf{r}_{\textbf{FG}}$	$\textbf{F1}_{\textbf{FG}}$	$\textbf{p}_{\textbf{ROI}}$	$\textbf{r}_{\textbf{ROI}}$	$\textbf{F1}_{\textbf{ROI}}$
	144	32.6	79.6	55.9	65.7	71.0	27.6	39.7
144	288	35.8	87.9	46.2	60.3	76.8	31.2	44.3
	576	32.0	88.0	23.3	36.9	76.0	31.8	44.8
	144	31.8	72.1	60.7	66.0	69.0	26.7	38.5
288	288	50.7	85.0	71.0	77.4	73.9	46.2	56.9
	576	57.9	**89.9**	61.5	72.8	74.4	60.0	66.3
	144	12.1	81.5	34.1	48.1	**87.1**	8.6	15.6
576	288	43.4	81.2	70.3	75.4	78.1	37.4	50.4
	576	**61.3**	87.0	**77.7**	**82.1**	76.4	**60.2**	**67.3**

As shown in Tab. 4.3, reducing the inference resolution generally degrades segmentation and ROI generation quality. However, training at a lower resolution allows for inference at resolutions up to eight times higher while improving overall performance across all models. Notably, the model trained at 288×288pixels delivers strong performance when inferred at both 288^2 and 576^2 resolutions. The difference in F1_{ROI} between training at 288^2 and inference at 576^2 versus training at 576^2 and inferring at 576^2 is marginal (approximately one percentage point), indicating comparable quality. From a practical standpoint, especially for deployment considerations, the model trained at 288^2 is preferable. It maintains robust performance across both 288^2 and 576^2 inference resolutions, offering greater flexibility to balance the trade-off between computational speed and accuracy. Conversely, the model trained at 576^2 suffers a significant drop in quality when inferred at 288^2, limiting its usability in scenarios requiring lower inference resolution. These results highlight the benefit of training models at moderate resolutions (288^2), which strike an effective balance between inference flexibility and detection quality, especially for ROI selection within the scope of tiny object detection.

Label Generation Method Using an input resolution of 288^2 pixels, the impact of different label generation methods on the quality of `ROI Estimation` was examined. Four experiments were conducted where object masks were generated by expanding bounding boxes with fixed pixel margins N set to 0, 2, 8, or 16 pixels. Additionally, two adaptive methods were tested: one with N varying inversely with object size (N_{A} in Tab. 4.4) and another with N varying directly with object size (N_{D} in Tab. 4.4), with N constrained between 2 and 16 pixels in both cases. All models were evaluated against the original dataset (N_0), in which masks precisely correspond to the original object detection labels. Since the ROI precision (p_{ROI}) and recall (r_{ROI}) metrics assess region coverage rather than exact mask precision, they remain valid and comparable even when training and validation labels differ.

The version based on N_0 was selected as the default configuration due to its consistently strong performance across both ROI-related and object detection metrics. As shown in Tab 4.4, although all configurations with N>0 exhibit higher ROI recall (r_{ROI}), they tend to suffer from a decrease in ROI precision (p_{ROI}), with the exception of N_{16}. This indicates that as N increases, the generated ROI masks may become less specific, potentially leading to a higher number of false positives. This is likely due to the reduced alignment between the ROI labels and the actual

TABLE 4.4: Comparison of ROI and segmentation metrics for models trained on datasets with differently generated labels. Validation was performed on the N_0 dataset, with all models trained at an input resolution of 288^2 pixels. The TinyROI detection system was evaluated on images from the MTSD dataset. Additionally, basic object detection metrics are reported for each `ROI Estimation Model` to analyze the relationship between ROI selection quality and object detection performance within these regions.

Model	iou_{FG}	p_{FG}	r_{FG}	F1_{FG}	p_{ROI}	r_{ROI}	F1_{ROI}	AP	AP_{50}	AP_{75}
N_0	**59.0**	**88.6**	64.2	**74.4**	72.9	61.7	66.8	**48.5**	**80.7**	**53.8**
N_2	46.0	69.3	68.2	68.7	67.1	62.5	64.7	46.4	77.4	51.3
N_8	21.9	33.5	**71.1**	45.4	67.7	70.5	69.1	48.0	80.2	53.1
N_{16}	12.5	17.1	68.6	27.4	**73.2**	**75.5**	**74.3**	43.3	71.8	48.5
N_A	43.2	64.2	68.3	66.2	72.4	61.8	66.7	46.9	78.7	51.7
N_D	14.5	21.9	**71.8**	33.5	72.8	71.6	72.2	46.0	76.7	51.6

TABLE 4.5: Effect of detection window filtering methods on the total number of detection windows (and average per image), as well as detection quality measured by AP and AR in the TinyROI system evaluated on MTSD.

Filtering	Detection Windows	AP	AP_{50}	AP_{75}	AR_{100}
—	19669 (3.7)	**51.0**	**81.3**	**58.9**	**62.5**
Naive	10297 (1.9)	48.4	80.8	53.4	59.4
Naive, sorted	**9536 (1.8)**	48.5	80.7	53.8	59.3

object features at larger dilation levels. Interestingly, the relationship between N and ROI quality is non-linear, while N_{16} achieves the best scores in both precision and recall, its object detection performance is the weakest, highlighting an important insight from system development: strong ROI metrics do not necessarily translate to strong object detection results.

Throughout the development of SegTrackDetect, each component was assessed using final object detection performance as the guiding criterion rather than intermediate ROI scores alone. This ensures that design choices reflect their true impact on end-task performance. The current results were obtained using simple Non-Maximum Suppression (NMS) in `Global Filtering Module`, which may not adequately handle overlapping detection windows, potentially leading to suppression of valid detections or presence of partial detections. To mitigate this, the `Overlapping Box Suppression (OBS)` method was introduced later in the development process. `OBS` was specifically designed to filter out partial and redundant detections arising from object fragmentation, and its integration into the pipeline allows for more reliable detection outcomes.

4.5.3 Detection Windows Proposal

Filtering Algorithms To justify the design choices, an ablation study was conducted comparing different window filtering techniques in the `Detection Windows Proposal Block` (Fig. 4.3). The method of region filtering directly influences the number of detector runs, and consequently, the inference time. For an ideal `ROI Estimator`, however, this should not impact detection quality, assuming partial detections are properly filtered. Tab. 4.5 presents a comparison of detection quality and the number of detection windows on the full validation set (5320 images) of the MTSD dataset using the TinyROI system. The naive filtering algorithm nearly halved the number of required regions from 19669 to 10297. Further reduction was achieved by adding a sorting step, which lowered the number of ROIs to 9536. This sorted filtering approach is detailed in Algorithm 1. As the number of windows decreased, both Average Precision (AP) and Average Recall (AR) showed slight declines, potentially due to imprecise regions. Another possible reason is the

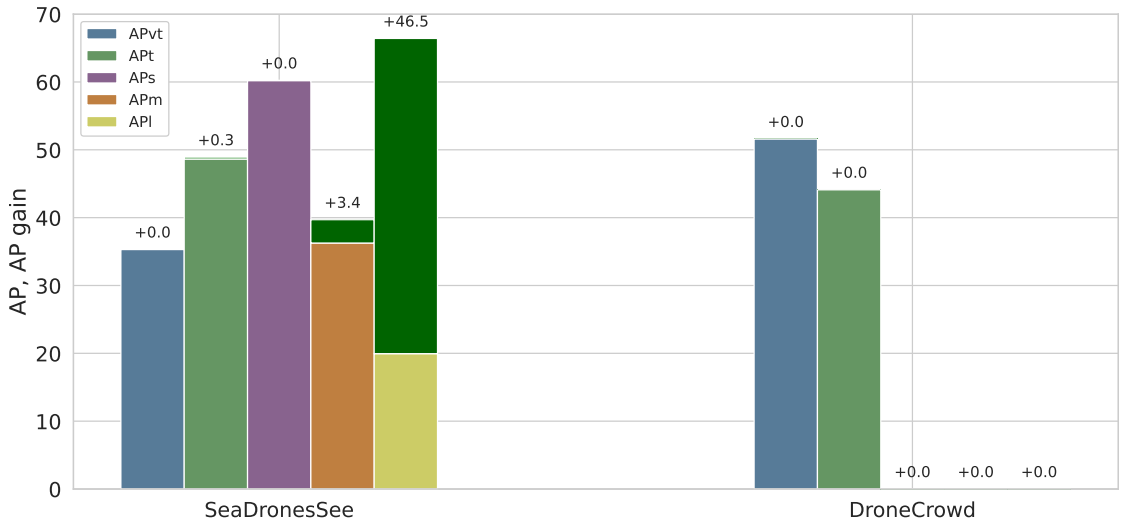


FIGURE 4.10: Effect of combining sliding-window and resizing strategies for oversized ROIs on detection quality across different object sizes. Results shown for SeaDronesSee and DroneCrowd datasets using the SegTrackDetect system.

use of Non-Maximum Suppression (NMS) rather than `Overlapping Box Suppression (OBS)`, which can mishandle partial detections. However, NMS in a `Global Filtering Block` would more likely degrade detection quality as the number of detection windows increases. A comprehensive analysis of `OBS` and its advantages over NMS in window-based systems is provided in Chapter 6. It is suspected that the observed quality drop stems from imprecise ROIs. However, a considerable reduction in processing time is prioritized over a marginal decline in detection performance.

This window filtering method was used in the final implementations of both TinyROI and SegTrackDetect.

Large Region Handling Approaches While the system was originally tailored for detecting tiny objects in high-resolution imagery (TinyROI), SegTrackDetect extends its capabilities to handle objects of varying scales. This enhancement ensures that detection quality is preserved not only for small, but also for medium and large objects commonly present in more diverse datasets. Evaluation was conducted on two benchmarks: SeaDronesSee, which features objects of various sizes, and DroneCrowd, which is focused on densely packed tiny objects. In contrast to TinyROI, which handled larger objects by increasing the detector's input resolution, without explicitly managing oversized regions of interest, SegTrackDetect incorporates a hybrid mechanism. For ROIs exceeding the detector's input size, the system combines a sliding-window approach with downsampling and cropping strategies. This design allows for precise detection across all sizes. The impact of this hybrid method is illustrated in Fig. 4.10. In SeaDronesSee, relying solely on the sliding-window technique results in reduced performance for medium and large objects. Incorporating resized ROIs enhances detection performance across all object sizes. In DroneCrowd, where large ROIs typically correspond to crowded scenes rather than large individual objects, the sliding-window approach alone remains sufficient for high detection quality. This combined approach enables SegTrackDetect to provide accurate detections in complex

TABLE 4.6: Detection post-processing results under different configurations. Merge = weighted averaging of overlapping boxes. Evaluated on MTSD (validation subset) with the TinyROI system.

$\mathbf{NMS_{local}}$	$\mathbf{NMS_{global}}$	**AP**	$\mathbf{AP_{50}}$	$\mathbf{AP_{75}}$	$\mathbf{AR_{100}}$
Non-Merge	—	23.8	48.8	19.2	49.4
	Non-Merge	32.7	73.1	21.7	43.9
Merge	—	32.3	50.4	38.6	**61.5**
	Non-Merge	46.2	78.4	50.8	54.7
	Merge	**50.0**	**78.8**	**58.0**	58.6

visual scenes, retaining the core advantages of TinyROI for small objects while addressing its limitations in handling large ROIs and multi-scale object distributions.

4.5.4 Local and Global Detection Filtering Methods

High-resolution inference often requires splitting the input into smaller overlapping windows. While Non-Maximum Suppression (NMS) is typically applied within each window independently (denoted as $\text{NMS}_{\text{local}}$ in Tab. 4.6), this local filtering does not account for redundancies across windows. Overlapping detection windows frequently produce multiple partial detections of the same object, which degrades precision and leads to redundant bounding boxes in the final output. This observation motivates the need for a `Global Filtering` stage that operates after aggregating detections across all windows.

Early experiments in TinyROI considered global NMS ($\text{NMS}_{\text{global}}$) as a straightforward extension to $\text{NMS}_{\text{local}}$. Results in Tab. 4.6 confirm that applying a second-stage NMS on the aggregated detections improves AP, although it may slightly reduce AR due to its tendency to suppress full detections instead of the partial ones. This trade-off highlights the limitations of conventional NMS when used in a multi-window detection pipeline. The evaluation in Tab. 4.6 includes variations in NMS stages and bounding box merging strategies (weighted averaging). All experiments use a 576^2px `ROI Estimator`, 960^2px input resolution of the `Local Detector Module`, and an IoU threshold of 0.45 and a confidence threshold of 0.01. The validation set from the annotation version N_0 was used.

The results show that omitting global NMS maximizes Average Recall (AR_{100}), but significantly lowers Average Precision (AP). Introducing global NMS increases AP at the cost of recall. Further gains are achieved when using merged box coordinates. These findings make a strong case for incorporating `Global Filtering` mechanisms in window-based detection systems; a concept designed and refined beyond standard NMS in this dissertation (Chapter 6). The final version of TinyROI adopts the best-performing configuration from Tab. 4.6, applying both local and global NMS with merged box coordinates.

4.6 Conclusions

This chapter examined the role of learned `Region of Interest (ROI) Estimation` in improving the efficiency and accuracy of window-based object detection systems. Two system variants were evaluated: TinyROI [67], a simple initial version of the framework, and SegTrackDetect [69, 70], a full-featured, ROI-based detection system. Both were compared against conventional sliding-window approaches across multiple datasets with diverse object scales and densities.

The primary contribution of SegTrackDetect over TinyROI lies in its large ROI handling strategy. By dynamically combining crop-and-resize with a sliding-window applied only within oversized ROIs, SegTrackDetect supports smaller detector input sizes, reduces background processing, and improves runtime efficiency without compromising detection quality. In addition, several architectural refinements and runtime optimizations were introduced to stabilize performance across scales and to further improve the inference speed. Although SegTrackDetect also integrates the `Overlapping Box Suppression (OBS)` algorithm within its `Global Filtering Block`, this component does not influence the direct comparison between `ROI Estimation` and sliding-window methods. To ensure fairness, sliding-window baselines were implemented in both TinyROI and SegTrackDetect, making the `Global Filtering` identical across estimation-based and non-estimation pipelines. The specific impact of `OBS` and other `Global Filtering` improvements is analyzed separately in Chapter 6.

The ablation study evaluated three object detection pipelines: single-shot detection with downsampling, a sliding-window method integrated into TinyROI, and the estimation-based version of TinyROI. This comparison revealed key limitations of the initial framework that motivated the development of SegTrackDetect. The study then examined the design choices underlying both frameworks.

The first part of the evaluation focused on TinyROI, using the Mapillary Traffic Sign Dataset (MTSD). Two baselines were considered: single-shot detection with downsampling and a sliding-window implementation embedded in TinyROI to ensure fairness. The single-shot baseline, while fast and effective for large objects, failed to detect smaller ones (e.g., AR_{μ}=1.1%, AP_{μ}=0.2%, AR_{vt}=22.9%, AP_{vt}=10.4%), highlighting the limitations of moderate downsampling, even with relatively large inputs (960×960 px). While the sliding-window method can restore small-object features by performing full-resolution inference over a regular grid, this comes at the cost of significantly slower inference (1.2 FPS on an RTX 3080 Ti) and reduced performance on larger objects (AP_{m}=47.9%, AP_{l}=39.5%). In contrast, TinyROI's learned `ROI Estimation Block` provides an almost seven-fold speed-up (7.9 FPS) while maintaining comparable quality to the sliding-window method.

These findings established `ROI Estimation` as a promising alternative to sliding-window, though TinyROI mainly delivered speed improvements rather than clear quality gains. The subsequent SegTrackDetect framework was designed to overcome these limitations, introducing further architectural refinements that allow it to outperform sliding-window approaches in both efficiency and accuracy.

Building on the ablation study results, the SegTrackDetect framework addresses several architectural limitations identified in TinyROI. First, by incorporating a large ROI handling strategy, SegTrackDetect significantly reduces the required input size of the detector, aligning it more closely with the average object size. This increases computational efficiency without sacrificing the detection quality of larger objects. Furthermore, larger objects (more prone to fragmentation and lacking contextual cues) benefit from `OBS`, which replaces NMS in the `Global Filtering` stage, improving quality across scales. As shown in Tab. 4.1, this particularly benefits datasets such as SeaDronesSee and ZebraFish. The lack of performance improvement on MTSD by SegTrackDetect is attributed not to limitations of the framework, but rather to a suboptimal training process. Specifically, the training data involved random cropping and aggressive downscaling, which led to underrepresentation of large objects in full resolution. As a result, the detector was not trained effectively to recognize them on a full scale. In contrast, SeaDronesSee and ZebraFish used cropped training images with slight up/downscaling, helping the network generalize better across object sizes and resolutions, which explains the strong performance of SegTrackDetect in these datasets.

SegTrackDetect consistently outperforms both TinyROI and all sliding-window methods in detection quality and inference speed. For example, on SeaDronesSee it achieved AP=52.6%, AP_{50}=83.5%, AR_{100}=60.9%, with strong results across all object sizes (AP_{vt}=32.0%, AP_t=51.9%, AP_s=61.6%, AP_m=36.4%, AP_l=67.5%) at 44.5 FPS. On ZebraFish, it achieved AP=80.7%, AP_{50}=95.7%, AR_{100}=83.8%, with small and medium-scale metrics reaching AP_t=41.2%, AP_s=76.1%, AP_m=87.7%, at 78.0 FPS. In MTSD, while SegTrackDetect was the fastest, quality improvements were not observed, again likely due to the training regime discussed above.

These results strongly support Auxiliary Thesis #1: "Estimating ROIs using a deep neural network model enhances both detection quality and inference speed compared to the naive sliding-window approach". The proposed ROI estimation-based system demonstrates that incorporating learned region selection not only reduces computational overhead but also improves overall detection quality by preserving contextual cues and avoiding redundant processing. The next chapter introduces a tracking-assisted ROI selection mechanism. Together, these allow the `ROI Estimation Network` to be further reduced in size without degrading performance, thereby improving runtime efficiency and enabling broader deployment of high-resolution small-object detection systems in real-time resource-constrained applications.

Chapter 5

ROI Estimation and Prediction-based Tiny Object Detection

5.1 Introduction

The SegTrackDetect system, introduced in the previous chapter, uses the `ROI Estimation Module` to select regions for high-resolution inference. This approach provides a strong baseline for real-time, resource-constrained environments, consistently outperforming sliding-window baselines in terms of detection quality, computational efficiency, and inference speed. However, despite operating on inputs significantly smaller than the original high-resolution images, the `ROI Estimation Module` may still be too resource-demanding for the most constrained platforms. Some applications require even lower GPU memory usage or faster processing speeds, which can only be achieved by further reducing the `Estimator's` input resolution. While this reduction decreases computational demand and speeds up inference, it comes at the cost of detection quality. Another limitation of the previous design is that it processes each frame independently, without exploiting the sequential nature of video data. In many real-time scenarios, objects persist across frames, and their trajectories provide valuable cues that could be leveraged to refine the region selection process. By ignoring this temporal continuity, the system may risk missed detections when objects are temporarily occluded or blurred.

To address these limitations, this chapter introduces an additional ROI source that leverages temporal information. The proposed `ROI Prediction Module` exploits tracking across frames, enabling lower computational demand without sacrificing detection quality. In resource-constrained scenarios, it allows the `Estimator` to operate at even further reduced resolutions while recovering missed regions. Conversely, when accuracy is prioritized, it complements a relatively high-resolution `Estimator` by improving ROI coverage with minimal overhead. In summary, the `ROI Prediction Module` improves both efficiency and accuracy by exploiting a fundamental characteristic of real-time applications – the continuity of the data.

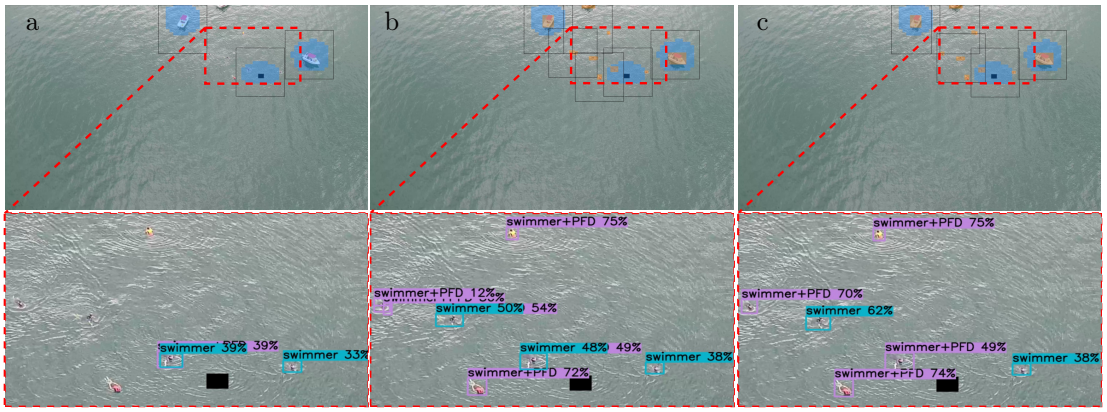


FIGURE 5.1: Illustrative application scenario: this work addresses the challenge of detecting small objects in high-resolution imagery. The method in (a), corresponding to the estimation-only approach from Chapter 4, uses segmentation-based estimated ROIs (blue masks) and fails to capture certain regions containing tiny objects. Incorporating tracking-based predicted ROIs (orange masks), as shown in (b), helps recover these missed regions. The proposed system in (c) fuses segmentation- and tracking-based ROIs to improve coverage of small objects and applies a `Global Filtering` stage using the `Overlapping Box Suppression` algorithm, resulting in a clean, false-positive-free output. Detection windows are indicated by black rectangles.

Processing full-resolution images in a single pass demands extensive computational power often unavailable in embedded or robotic platforms. To achieve real-time speeds, many methods reduce image resolution or simplify detector architectures. Popular general-purpose detectors such as YOLOv7 [136], YOLOv12 [127], and transformer-based models like DETR [18, 190] operate on downscaled inputs, trading off detail for speed. However, downsampling frequently eliminates the subtle features of tiny objects, resulting in missed detections. Alternatively, dividing images into tiles and processing each independently [2, 96, 151, 178] preserves detail but is computationally expensive and unsuitable for real-time constraints. Attention-based methods improve efficiency by focusing detection on promising regions [15, 31, 34, 67, 71, 75, 142, 155, 157, 169], significantly reducing detector invocations. Yet, even these methods risk overlooking some objects if their corresponding regions are not selected.

Tracking algorithms have been widely used to associate small objects in video frames [17, 90, 147, 176, 184, 187]. However, tracking is rarely utilized as a feedback mechanism to guide and improve detector focus and accuracy, this represents the main innovation presented in this chapter.

The approach simultaneously improves detection accuracy and computational efficiency by leveraging an attention mechanism to prioritize the most promising image regions for detection. To recover missed detections from initially unselected areas, tracking-based `ROI Prediction` identify regions likely to contain hard-to-detect objects. By combining segmentation-driven `Estimation` with tracking-guided `Prediction`, the method increases the probability of detecting small objects missed due to resolution limits or occlusions. This results in a novel window-based detection framework tailored for tiny and multi-scale objects in high-resolution images. The system integrates two ROI modules: a segmentation-based branch analyzing a downscaled current frame to estimate candidate regions, and a tracking-based branch leveraging historical detection data to predict object locations. Fusing their outputs recovers areas missed by segmentation alone, reducing the number of detector invocations compared to exhaustive sliding-window methods

while maintaining speed. Within large ROIs, a hybrid of downscaling and sliding-window detection is applied, complemented by an `Overlapping Box Suppression (OBS)` algorithm to suppress partial false positives. An overview of these components and their effect on detection output is illustrated in Figure 5.1. Evaluations on challenging datasets such as SeaDronesSee and DroneCrowd demonstrate that this framework achieves superior performance compared to state-of-the-art detectors. While this Chapter focuses on isolating the impact of the `ROI Prediction Module`, the advancements in the `Global Filtering Block` are analyzed in detail in Chapter 6.

5.2 Contribution

While the detection systems presented in Chapter 4 focused on improving detection speed and quality over naive sliding-window approaches through per-frame `ROI Estimation`, this section extends that concept by leveraging temporal consistency, which is inherent in many real-world video-based applications. Such consistency is especially relevant in domains like mobile robotics, autonomous navigation, UAV surveillance, and marine or aerial monitoring, where objects tend to persist across consecutive frames and their motion is often predictable. This characteristics may be used as an advantage to improve the ROI selection, while maintaining or improving the computational efficiency of the detection system with lightweight tracking component. By exploiting this consistency, the proposed system improves ROI selection while maintaining, or even enhancing, computational efficiency through the integration of a lightweight tracking component.

The proposed SegTrackDetect framework is enhanced with an `ROI Prediction Branch` that uses previous detections to predict object locations in the current frame [69, 70]. This dual estimation- and tracking-based ROI approach brings several key benefits: improved detection recall, especially for small objects that may be missed in a single frame, reduced computational complexity by allowing a lower input resolution for the `Estimation Branch` without sacrificing accuracy, and increased robustness through fusion of estimated and predicted ROIs, enabling more complete object coverage. Compared to the estimation-only variant of SegTrackDetect, the main contributions presented in this chapter are the introduction of the `ROI Prediction Branch` and the `ROI Fusion Module`, both of which are discussed in detail in the following sections. These architectural enhancements, particularly the integration of temporal `ROI Prediction` with frame-wise `Estimation`, combined with a robust `Detection Windows Proposal Module` and postprocessing strategies, allow SegTrackDetect to achieve a strong balance between accuracy and efficiency. Instead of relying on exhaustive tiling, the system selectively processes relevant regions, improving recall for small or previously missed objects while significantly reducing computational overhead. These improvements make the system especially well-suited for real-time, resource-constrained applications.

To put the proposed method into context, it is important to highlight how it differs from and improves upon existing small and tiny object detection approaches. These methods face unique challenges, as small objects occupy only a few pixels, making them hard to detect. In high-resolution images, although the total pixel count increases, processing the entire image becomes computationally expensive. A common solution is to apply a sliding-window strategy over high-resolution images to artificially enlarge the relative size of the objects. However, this approach

processes all regions of the image, including background, making it inefficient, especially in domains like aerial imagery, where objects often appear in small clusters and large parts of the image are empty. Focus-and-detect methods attempt to address this by identifying promising regions before detection. For example, [151] supplements the sliding-window approach with a lightweight neural network to filter out background-only tiles. However, the detection windows in that method are aligned to artificial grids rather than actual object clusters. In contrast, SegTrackDetect aligns detection windows with estimated and predicted regions containing objects. This approach is more adaptive and better maintains the detection context.

Several coarse-to-fine strategies also follow this general idea [34, 71, 75, 142, 157, 169], first locating candidate regions using low-resolution features and then performing high-resolution detection. ClusDet [157], for instance, predicts coarse locations using a low-resolution network and refines them using high-resolution features, but it focuses only on clusters containing at least three objects. This biases the model towards dense regions and makes it ineffective for detecting isolated or sparse objects. Similarly, [34, 75] use density maps to prioritize densely populated areas, missing sparse detections. Moreover, these approaches often require training labels for clusters or density maps, information not directly available in standard detection datasets. In contrast, SegTrackDetect leverages binary segmentation masks generated directly from detection labels, and combines them with a lightweight, non-trainable tracking-based `ROI Prediction Branch`. This allows the system to effectively handle both densely clustered and sparsely distributed objects. The tracking branch also improves robustness under occlusions and motion blur by incorporating temporal information from previous frames. Additionally, while other methods introduce scale estimation modules to handle resizing and cropping [34, 75, 157], the proposed `Detection Window Proposal Module` directly generates ROIs that match the input size of the detector, using a mix of resized and sliding-window regions when needed. This keeps the system efficient and avoids additional computational costs.

The method in CDMNet [34] also merges ROIs from two sources (density map and segmentation map), but its segmentation map is a binarized version of the density map. In the proposed approach, segmentation labels are directly derived from ground truth detection labels, and the second source is a tracking module, not another trained map. Other methods like CRENet [142] rely on clustering after initial detection to guide fine detection but cannot recover missed objects. DREN [31] attempts to address this by applying super-resolution to enhance low-quality object regions, but this significantly increases computational requirements. Instead, SegTrackDetect restores such regions using low-resolution segmentation and tracking, supporting fast detection with a lightweight detection model.

Another challenge in window-based detection is how to combine detections from overlapping regions. Most systems [34, 75, 142, 157, 169] use standard Non-Maximum Suppression (NMS), which can struggle with partial or fragmented detections. In SegTrackDetect, the limitations of standard NMS have already been mitigated by integrating the `Overlapping Box Suppression (OBS)` algorithm into the `Global Filtering Block`. However, in this chapter the evaluation isolates the `ROI Prediction Module`, so all improvements in metrics can be attributed directly to the `Predictor`. The impact of filtering methods such as OBS and OBM is analyzed separately in Chapter 6. Finally, the SegTrackDetect system can work together with recent advances in

small object detection architectures that improve feature extraction and label assignment [19, 75, 103, 153, 154]. Techniques like Switchable Atrous Convolutions, attention mechanisms, and Gaussian receptive field-based assignment can be used inside the framework to further boost performance. In this way, SegTrackDetect combines strong ROI proposal and fusion based on temporal consistency with flexibility to use state-of-the-art small object detection methods, creating a powerful and efficient solution for challenging detection problems.

The research presented in this chapter aims to validate and support Auxiliary Thesis #2, which states: "Incorporating an additional ROI source, such as an object tracker, into the ROI-based object detection system further improves detection quality, inference speed, and computational efficiency over state-of-the-art tiny object detection methods". To this end, the proposed detection framework (which combines spatial and temporal cues to select Regions of Interest (ROIs) and corresponding detection windows) is evaluated against several state-of-the-art detection approaches. The comparison is conducted in terms of both detection accuracy and computational efficiency. To ensure a fair and comprehensive evaluation, the benchmark includes general-purpose object detectors, methods specifically designed for small and tiny object detection, and video object detection approaches. While SegTrackDetect system leverages temporal information through tracking, it does not rely on sequence-based training; nevertheless, video object detection methods are included in the comparison for completeness. Based on the literature review presented in this section and Chapter 2, the following representative methods were selected for comparison with the proposed SegTrackDetect framework:

- General Object Detection: SAM [62], InternImage [140], YOLOv7 [136], YOLOv10 [135]
- Small and Tiny Object Detection: DetectoRS [103], QueryDet [156],
- Video Object Detection: DiffusionVID [113], DFF [186], FGFA [185], RDN [30], MEGA [22].

Beyond the main quantitative and qualitative evaluations, an extensive ablation study is also included to analyze the effect of the tracking-based `ROI Prediction` component on the overall detection quality, speed and computational cost compared to a purely segmentation-based `ROI Estimation` strategy. In summary, the main contributions to the state-of-the-art demonstrated in this chapter are as follows:

- a lightweight, ROI-based video object detection system that integrates standard detection models, tailored specifically for detecting small and tiny objects in high-resolution images, and achieving superior performance compared to existing methods,
- a novel `ROI Fusion Module` that combines tracking-based `ROI Prediction` with low-resolution, segmentation-based `ROI Estimation`, enabling precise localization of objects across varying scales,
- a comprehensive component-wise evaluation using three datasets (SeaDronesSee, DroneCrowd, and ZebraFish) along with detailed comparisons to multiple state-of-the-art detection methods on the challenging DroneCrowd and SeaDronesSee benchmarks.

The content of the following sections is derived from and consolidates the author's earlier publications [69, 70], which form the foundation of this part of the dissertation.

5.3 Proposed method

Figure 5.2 illustrates the overall architecture of the proposed SegTrackDetect object detection framework, tailored for detecting small and tiny objects in high-resolution video data. The design is specifically optimized for time-sensitive applications in embedded systems such as autonomous vehicles [162] and UAV-based monitoring platforms [184], where both high detection accuracy and low inference latency are essential. High-resolution imagery increases the computational load, often reducing the inference speed of standard object detectors. To address this, the system restricts inference to selected subregions within each frame, rather than performing dense sliding-window detection across the full image. The proposed approach is grounded in the premise that real-time detection of small and tiny objects is feasible using lightweight detectors - provided their attention is guided by efficient auxiliary modules. To this end, the architecture integrates two complementary mechanisms: `ROI Estimation` and `ROI Prediction`, which work together to identify the most promising regions for object detection. By narrowing the focus of the detector to these regions, the system reduces computational overhead while maintaining or improving detection accuracy.

Processing begins with the current high-resolution frame $\mathbf{I}_t$. Drawing on region proposal concepts from classical object detection methods [48], an `ROI Estimation Module` is used to produce a binary heatmap $\mathbf{R}_t^e$, highlighting areas with higher object probability. This allows the system to ignore large background areas and significantly reduce the number of regions passed to the detector. The estimation-based version of this approach, relying solely on the segmentation-derived ROI mask $\mathbf{R}_t^e$, was already presented and analyzed in Chapter 4. It was shown to be effective in improving detection quality, particularly for medium and large objects, while significantly reducing processing time compared to naive sliding-window approaches. The focus of this chapter is to extend that baseline by introducing further optimizations, including the use of temporal consistency via tracking, and to benchmark the resulting system not only against the sliding-window baseline but also against state-of-the-art full-frame object detectors. Compared to Chapter 4, the system is extended with a tracking-based `ROI Prediction Branch` and a `Fusion` mechanism that combines both sources of information. These additions aim to improve detection quality, speed, or both, by compensating for missed regions and leveraging temporal consistency.

Detecting small or partially occluded objects in a single frame remains challenging due to limited visibility and low contrast. Human perception, however, benefits from temporal consistency, using motion cues across time to identify such objects. Inspired by this mechanism, a second `ROI Prediction Branch` is incorporated to leverage temporal continuity through object tracking. This module predicts the current position of previously detected objects using past detections, resulting in a heatmap $\mathbf{R}_t^t$ that marks likely regions of interest. By incorporating this prior knowledge, detection performance is improved for objects that were visible in earlier frames but are now partially obscured or harder to distinguish due to size or contrast issues.

The outputs of the `ROI Estimation` and `ROI Prediction Modules` ($\mathbf{R}_t^e$ and $\mathbf{R}_t^t$ respectively) are merged via an element-wise OR operation to produce a Fused ROI Mask ($\mathbf{R}_t^f$). This mask is used by the `Detection Window Proposal` Module to define a set of fixed-size windows within

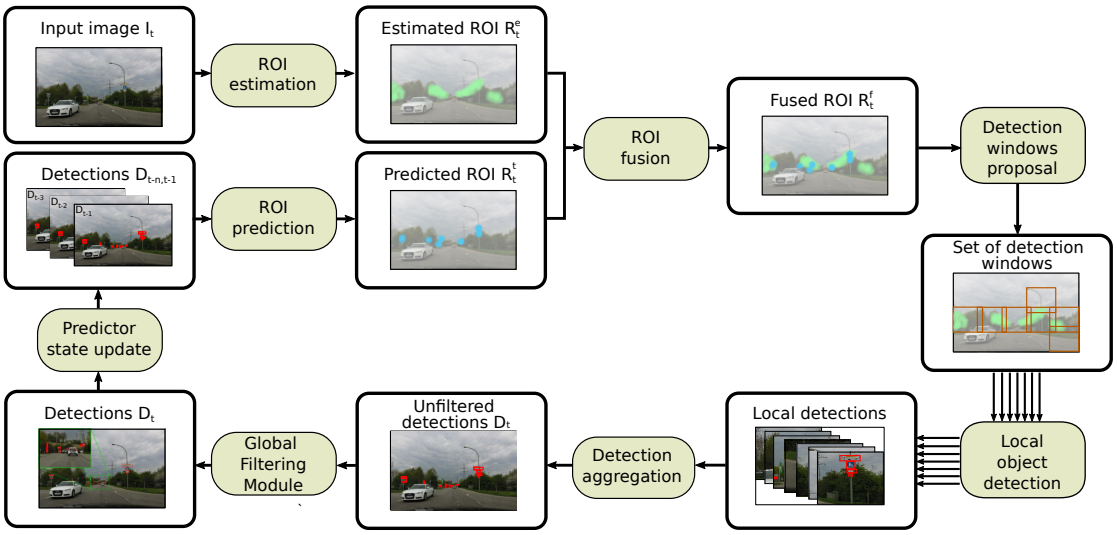


FIGURE 5.2: Architecture of the proposed system for detecting small and tiny objects. The detector is guided toward selected fixed-size regions in the high-resolution input image to improve efficiency. These regions are identified using two complementary ROI branches: (1) current-frame `Estimation` from a low-resolution binary segmentation map (R_t^e) and (2) `Prediction` from an object tracker (R_t^t). The union of these masks forms the fused ROI mask (R_t^f), which is processed by the `Detection Window Proposal Module`. A lightweight `Local Object Detector` is then applied to these subregions, and detections are projected back to the original image coordinates. Finally, the `Overlapping Box Suppression (OBS)` used in a `Global Filtering Module` filters redundant outputs before updating the object tracker, enabling predictive ROI generation in the next frame.

the high-resolution frame where detection is to be performed. In scenarios where the ROI spans a larger area than the detector's input size (e.g., dense object clusters, large objects), either downsampling or a localized sliding-window strategy is employed to ensure coverage, while maintaining context. The proposed windows are passed through a lightweight object detector, and the resulting bounding boxes are then rescaled to match their corresponding locations in the original image. To eliminate false positives and duplicate detections arising from overlapping windows, the `Overlapping Box Suppression (OBS)` algorithm is applied (`Global Filtering Module` in Fig. 5.2). Unlike traditional Non-Maximum Suppression (NMS), `OBS` operates across detections from multiple windows and is especially effective when objects are split across regions, improving detection quality in multi-scale settings. While the improvements from the `Global Filtering Block` are analyzed in Chapter 6, the focus here is exclusively on the contributions of the `ROI Prediction Module` – its impact is isolated in the ablation studies so that all observed performance gains can be directly attributed to the `Predictor`.

Subsequent sections provide descriptions of each system component, including the segmentation-based `ROI Estimation`, tracking-based `ROI Prediction`, `Window Generation` process, and the `OBS` algorithm for post-processing. Several of these modules, particularly those related to segmentation-based `ROI Estimation` and `Detection Windows Proposal`, were already introduced in the context of the estimation-based SegTrackDetect variant in Chapter 4. For these shared components, only a concise summary is provided here, emphasizing the differences and design choices motivated by the ablation studies presented in the previous chapter. In contrast, the tracking-based `ROI Prediction` and `ROI Fusion` mechanisms, which represent novel extensions in this chapter, are described in detail.

5.3.1 ROI Estimation Module

To efficiently detect small objects in high-resolution imagery, the proposed system leverages a Region of Interest (ROI) Estimation Module (Fig. 5.2), shared with the solely estimation-based version of the SegTrackDetect system introduced in Chapter 4. As in the prior system, `ROI Estimation` is formulated as a binary segmentation task, with ground-truth masks generated directly from bounding box annotations. Following the findings from Chapter 4, ground-truth masks are constructed to exactly match the bounding box geometry. This exact alignment between labels was shown to yield superior detection performance.

The `Estimation Module` is implemented using a U-Net [114] with a ResNet18 backbone and operates on input images resized to a fixed resolution. In this chapter, however, the `Estimation Network` is used in conjunction with an additional `ROI Prediction Branch` based on object tracking, which compensates for regions the segmentation-based `Estimator` might miss. As shown in the ablation study later in this chapter, this fusion allows a reduced input resolution for the `Estimation Module` compared to Chapter 4, improving the processing speed without sacrificing detection quality. Unlike Chapter 4, which evaluated the `ROI Estimation Module` in isolation, this chapter evaluates both ROI components indirectly through their impact on overall object detection performance. For the two evaluated datasets, the `Estimation Branch` operates on inputs of 448×768 for SeaDronesSee and 192×320 for DroneCrowd. Additional resolution settings for SeaDronesSee are explored and discussed in the ablation study.

5.3.2 ROI Prediction Module

In many cases, low-resolution segmentation-based `ROI Estimators` struggle to consistently localize tiny objects, especially across consecutive frames, leading to unstable detection results. To address this, an additional tracking-guided `ROI Prediction Module` is introduced (Fig. 5.2). This module serves as a complementary source of ROIs, helping to recover object regions that may be missed by the segmentation branch alone.

The prediction process begins with initializing object tracks using the detections found within the estimated ROIs of the first frame ($t = 0$). For all subsequent frames ($t > 0$), object locations are inferred using a constant velocity motion model, allowing the system to predict new ROIs based on previously observed motion patterns. The predicted locations are initially represented as bounding-box coordinates and are converted into binary mask format prior to fusion with the current segmentation-based ROI proposals. The fused ROI mask is then used to guide detection, and the resulting filtered detections are used to update the internal state of the tracker.

This module is implemented using a simplified multi-object tracking scheme inspired by the SORT algorithm [9]. Specifically, a Kalman filter is employed to maintain the internal state of each track, and frame-to-frame data association is performed by minimizing a cost matrix defined using IoU between predicted and observed bounding boxes. Each object's Kalman state vector $\mathbf{x}$ consists of geometric and kinematic properties: the bounding box center coordinates (x_c, y_c), area A, aspect ratio r, and their respective velocities ($\dot{x_c}$, $\dot{y_c}$, $\dot{A}$). The corresponding

measurement vector $\mathbf{z}$ includes only the observable elements: $[x_c, y_c, A, r]$. The system assumes constant velocity and a fixed aspect ratio, leading to the following state transition model:

$$\mathbf{x}_{t+1|t} = \mathbf{F}\mathbf{x}_t, \tag{5.1}$$

with the transition matrix $\mathbf{F}$ defined as:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.2}$$

The projection model $\mathbf{z}_t$ maps the state vector to the measurement space:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t, \tag{5.3}$$

with:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{5.4}$$

Each track is initialized using the first detection and assumes zero velocity at startup. During each prediction step, the system estimates the new state $\mathbf{x_{t|t-1}}$ and error covariance $\mathbf{P_{t|t-1}}$:

$$\mathbf{x_{t|t-1}} = \mathbf{F}\mathbf{x_{t-1|t-1}}, \tag{5.5}$$

$$\mathbf{P_{t|t-1}} = \mathbf{F}\mathbf{P_{t-1|t-1}}\mathbf{F}^T + \mathbf{Q}. \tag{5.6}$$

where $\mathbf{Q}$ is the process noise matrix.

When new detections become available, the filter performs a correction step:

$$\mathbf{x_{t|t}} = \mathbf{x_{t|t-1}} + \mathbf{K_t}(\mathbf{z_t} - \mathbf{H}\mathbf{x_{t|t-1}}), \tag{5.7}$$

$$\mathbf{P_{t|t}} = (\mathbf{I} - \mathbf{K_t}\mathbf{H})\mathbf{P_{t|t-1}}, \tag{5.8}$$

where $\mathbf{K}_t$ is the Kalman gain.

This tracking-based `ROI Prediction` approach is especially helpful for small or partially occluded objects that might be missed by the segmentation-based `Estimator` due to fast motion, low contrast, or lack of clear visual features. By using temporal information, the system can recover such objects based on their earlier positions, making the detection process more stable and reliable across frames. This improves recall for hard-to-detect objects and increases the overall robustness of the pipeline. To keep inference fast, the system uses a lightweight tracking module, instead of more complex approaches relying on deep features [90, 148]. Since the goal is not to

maintain full object tracks but only to suggest additional regions for detection, high tracking accuracy across long sequences is not required. This design choice ensures efficient motion prediction and object association without significantly increasing computational load. While the tracker can occasionally lose object identity or drift in crowded scenes, such errors have little impact on detection results because the tracker is used only to propose ROIs, not to manage object identities.

In the implemented system, the tracker is configured with a maximum age of 10 frames, meaning a track can persist for up to 10 consecutive frames without receiving a new detection before it is discarded. A new object hypothesis is confirmed after a single detection, allowing for rapid adaptation to newly appearing objects, ths improving ROI recall. For association between predicted and observed bounding boxes, a minimum Intersection over Union (IoU) of 0.3 is required. To improve prediction stability in the early part of each video, the tracking module introduces a frame delay of 3 during the first few frames. This delay helps accumulate enough detections before initializing motion-based ROI predictions, reducing the risk of early tracking errors due to sparse or noisy detections.

5.3.3 ROI Fusion Module

The `ROI Fusion Module` combines the outputs of the two ROI selection branches: the segmentation-based `Estimation Module` and the tracking-based `Prediction Module`. Each branch produces a binary heatmap indicating regions likely to contain objects. These heatmaps are aligned to the same spatial resolution and then merged into a single binary mask using an element-wise logical OR operation. This fused mask defines the final Regions of Interest and serves as input to the `Detection Windows Proposal` stage. To ensure compatibility, the bounding boxes predicted by the tracking branch are first converted into a binary mask format that matches the resolution of the segmentation output.

While the `Fusion Module` supports different configurations, here the primary focus is on the combined use of both ROI branches. Using only the `Estimation Module` (as discussed in Chapter 4) is possible and is revisited here in the ablation study for comparison. This baseline configuration helps quantify the improvements brought by incorporating the tracking branch, both in terms of detection quality and runtime efficiency. In tracking-only mode, the tracker is first initialized using standard sliding-window detection for a fixed number of frames (N). After this warm-up phase, new ROIs are predicted solely based on object motion. Although this mode is not the primary focus of the system, it offers a lightweight alternative for fast video inference, especially effective in relatively stable environments where new objects rarely appear.

In general, the fusion of both branches allows the system to maintain high detection recall while reducing redundant processing, especially in high-resolution video settings. The ablation study presented later in this chapter compares these configurations to demonstrate the benefits of the joint approach.

5.3.4 Detection Windows Proposal

The `Detection Windows Proposal Module` converts the fused ROI mask, produced by combining segmentation-based and tracking-based proposals, into a set of image subregions that are processed by the object detector. The implementation of this component remains consistent with the architecture described in Chapter 4, and is illustrated in Figs. 3.1 and 4.4. As in the earlier version, the fused ROI heatmap is first binarized to extract individual foreground regions. The bounding boxes of these regions are used to generate an initial, unfiltered list of candidate windows. These are then aggregated and filtered to eliminate redundancy (see "Detection Windows Filtered" in Fig. 4.4). The resulting refined set of windows is extracted from the original input image and forwarded to the `Local Object Detection Module`.

As discussed in Chapter 4, SegTrackDetect introduces a more flexible and adaptive strategy for constructing detection windows, designed to address the challenges of varying object scales and real-time constraints. This hybrid approach combines cropping with both resizing and sliding-window tiling when the ROI exceeds the detector input size. In contrast, when the ROI is smaller than the detector input size, a single-centered crop is extracted without resizing. Unlike TinyROI, which used fixed-size detection windows and applied gray padding when they extended beyond the image boundaries, SegTrackDetect constrains all windows to lie fully within the image area. This eliminates padding artifacts and enables the use of smaller input sizes for the detector, improving overall computational efficiency. The filtering stage follows the same naive sorting-based approach validated in Chapter 4. As shown in the ablation studies presented there, this simple method offers a trade-off between runtime and detection quality. The complete pseudocode for the window generation and filtering pipeline is provided in Algorithm 1, which remains applicable here.

5.3.5 Local Object Detection

As in the system described in Chapter 4, this component is responsible for performing object detection within the proposed subregions of the original image. By restricting inference to the selected ROIs, the system reduces computational load without compromising spatial resolution. Thanks to the adaptive detection window generation strategy introduced in SegTrackDetect, lightweight detectors such as YOLOv7 Tiny [136] can be employed effectively across diverse scenarios. For both the SeaDronesSee and DroneCrowd datasets, the model operates with an input resolution of 512×512 pixels. For the less complex ZebraFish dataset, a smaller resolution of 160×256 is used instead. These low input resolutions are feasible because large ROIs are either tiled or moderately downsampled in the `Detection Windows Proposal Block`, ensuring that all relevant regions are covered.

Although newer YOLO variants (e.g., YOLOv10 [135], YOLOv12 [127]) are available, YOLOv7 Tiny was selected for its favorable balance of inference speed, stability, and accuracy. As shown in Tables 5.1 and 5.2, even the largest YOLOv10 variant (YOLOv10x) does not consistently outperform YOLOv7-e6e in the tested scenarios, despite requiring significantly more resources.

Training and inference procedures are consistent with those outlined in Chapter 4. Detection datasets are prepared by cropping annotated full-resolution frames into subwindows and adjusting bounding boxes to local coordinates. Negative samples are included to reduce false positives, and scale augmentation is used to improve generalization across object sizes.

During inference, detections within each window are post-processed using Non-Maximum Suppression (NMS). The final aggregation step transforms local detections back to the global image space, incorporating scaling adjustments when applicable.

5.3.6 Global Filtering Module

To finalize the detection results across all sub-windows, a `Global Filtering` step is performed to remove redundant or fragmentary detections - common artifacts in systems that rely on overlapping detection windows. As demonstrated in the ablation study in Chapter 4, `Global Filtering` plays a critical role: on the MTSD dataset, TinyROI's performance measured by AP increased from 32.3% to 50.0% when global NMS was applied. This highlights the importance of an effective global post-processing step.

In SegTrackDetect, this role is fulfilled by the `Overlapping Box Suppression (OBS)` method, a dedicated filtering algorithm tailored to the window-based detection setting. Unlike standard NMS, which evaluates boxes solely based on confidence scores and spatial overlap, `OBS` also considers the position of the detection windows from which each box originates. This additional context enables it to distinguish between full-object detections and incomplete fragments caused by intersecting detection windows. By accounting for how each object was viewed across the set of overlapping regions, `OBS` provides more precise suppression decisions, reducing false positives while preserving true positives that standard NMS may incorrectly discard. This becomes especially important in dense scenes, where overlapping views are frequent. A detailed breakdown of the `OBS` algorithm and its comparison to NMS is provided in Chapter 6.

5.4 Experimental results

The experimental evaluation presented in this section begins with a quantitative comparison of the SegTrackDetect framework (featuring a dual ROI selection strategy) against several state-of-the-art detection methods. This comparison includes three main groups of detectors: (1) general-purpose object detectors, ranging from lightweight YOLO [136] architectures to transformer-based models [190]; (2) tiny object detection methods; and video object detectors (3) that incorporate temporal consistency during training. Although SegTrackDetect employs standard image-based detection models, it leverages temporal information during inference via a tracking module. Therefore, video object detectors are included in the comparison to ensure fairness.

The system is evaluated on two challenging datasets: SeaDronesSee, which features high-resolution images, multiple object classes, and a wide range of object sizes (from micro to large) in relatively simple backgrounds; and DroneCrowd, which contains lower-resolution images, more complex backgrounds, and densely packed tiny objects. This dual-dataset evaluation enables a

TABLE 5.1: Evaluation of the proposed ROI-based approach against a range of state-of-the-art object detectors on the DroneCrowd test set. The assessment follows the protocol defined in [67], with a true positive IoU threshold of 0.5 and a detection cap of 500 per image. All models were trained with their respective original configurations.

method	**params**	**AP**	**AP$_{50}$**	**AP$_{75}$**	**AP$_{vt}$**	**AP$_{t}$**	**AR**	**AR$_{vt}$**	**AR$_{t}$**
InternImageT-DINO-LWLR [140]	48.6M	11.6	32.1	5.7	32.3	27.3	40.5	40.5	42.2
InternImageL-DINO-LWLR [140]	240.9M	18.6	48.3	10.3	48.1	**66.7**	69.9	69.9	72.9
YOLOv7 [136]	36.5M	10.1	31.3	3.7	34.2	10.3	51.7	51.8	53.2
YOLOv7e6e [136]	164.9M	18.5	47.6	10.5	48.4	36.4	62.7	62.7	68.7
YOLOv10x [135]	29.5M	17.9	47.6	9.0	47.8	45.0	69.5	69.5	71.5
SAM+CLIP-ViT-H-32points [62, 105]	635.8M	0.0	0.0	0.0	1.0	0.0	0.3	0.3	0.1
SAM+CLIP-ViT-H-64points [62, 105]	635.8M	0.0	0.0	0.0	1.0	0.0	0.8	0.8	0.2
DFF-GenRCNN-R101 [186]	194.6M	0.4	1.8	0.1	1.8	4.7	14.4	14.3	19.2
FGFA-GenRCNN-R101 [185]	198.3M	0.6	2.3	0.1	2.4	3.0	15.4	15.4	18.5
RDN-GenRCNN-R101 [30]	169.5M	0.7	2.7	0.1	2.7	5.0	15.6	15.6	18.5
MEGA-GenRCNN-R101 [22]	172.5M	0.6	2.4	0.1	2.5	1.5	15.0	15.0	16.7
DiffusionVID-DiffusionDET-SwinB [113]	145.3M	8.1	27.3	2.2	31.4	5.5	43.3	43.4	42.2
DetectoRS-CascadeRCNN-R50 [103]	124.0M	10.0	27.2	5.0	27.1	39.1	35.4	35.4	39.4
SegTrackDetect [69, 70]	**20.3M**	**22.8**	**53.8**	**15.2**	**53.9**	60.0	**73.3**	**73.4**	**75.8**

comprehensive assessment of the system under different detection scenarios and demonstrates its strong performance across real-world use cases.

Following the quantitative results, qualitative detection examples are presented alongside an analysis of common detection errors and their implications. While this section focuses on benchmarking against state-of-the-art methods, Section 5.5 isolates the impact of the `ROI Prediction Module` and compares it to both ROI estimation-only and ROI prediction-only versions of the SegTrackDetect system.

All results in this section use the following configuration. `Local Object Detection` is performed using the lightweight YOLOv7 Tiny [136] architecture with an input resolution of 512×512 pixels. The `ROI Estimation Module` is based on a UNet [114] with a ResNet18 backbone, with input resolutions of 192×320 pixels for DroneCrowd and 448×768 pixels for SeaDronesSee. The `ROI Prediction Module` uses default hyperparameters as detailed in Section 5.3.2.

5.4.1 Quantitative results

The SegTrackDetect framework is evaluated on two challenging datasets: DroneCrowd and SeaDronesSee. Quantitative results are presented in Tab. 5.1 and Tab. 5.2, comparing the system against a wide range of state-of-the-art object detection methods. These include general-purpose detectors [62, 135, 136, 140, 190], methods specifically designed for tiny object detection [103, 156], and video object detectors [22, 30, 113, 185, 186] that leverage temporal information during training or inference. Although SegTrackDetect is trained solely on individual frames, the use of a tracking module during inference allows the system to utilize temporal context in a lightweight and modular way, justifying the inclusion of video methods for comparison.

Tab. 5.1 shows the results obtained on the DroneCrowd dataset, where SegTrackDetect outperforms most evaluated models across nearly all metrics. Despite relying on compact components,

TABLE 5.2: Object detection performance on the SeaDronesSee validation dataset. The proposed system is compared with state-of-the-art object detectors in terms of detection quality (AP, AR) and model complexity (parameter count). Evaluation follows the protocol defined in [67], using standard IoU thresholds from 0.5 to 0.95 (step size 0.05) and a maximum of 100 detections per image.

method	params	AP	AP_{50}	AP_{75}	AP_{vt}	AP_t	AP_s	AP_m	AP_l	AR	AR_{vt}	AR_t	AR_s	AR_m	AR_l
InternImageT-DINO-LWLR[140]	48.6M	50.5	83.8	53.1	26.4	48.1	61.9	41.4	69.4	60.0	36.3	60.3	69.9	47.4	72.7
InternImageL-DINO-LWLR[140]	240.9M	50.7	84.9	53.1	26.1	47.1	**62.5**	44.1	69.9	59.9	35.2	59.3	**70.2**	50.9	73.5
YOLOv7[136]	36.5M	45.9	82.9	45.7	23.7	44.0	57.6	47.4	**71.8**	55.3	32.2	56.6	66.1	54.4	**76.9**
YOLOv7e6e[136]	164.9M	51.7	**86.9**	54.2	29.5	50.7	62.1	46.3	67.0	60.0	35.6	62.4	68.9	53.4	76.4
YOLOv10m[135]	15.4M	43.1	73.8	44.1	11.4	40.4	59.9	43.0	58.6	51.2	18.8	51.0	68.3	49.8	63.2
YOLOv10x[135]	29.5M	47.4	82.0	49.5	24.2	44.4	59.3	39.0	51.3	57.3	33.5	62.2	67.1	46.9	54.5
SAM+CLIP-ViT-H-32points[62, 105]	635.8M	18.4	31.5	20.4	2.8	8.0	32.9	16.1	51.0	29.1	4.1	18.5	47.1	26.7	58.6
SAM+CLIP-ViT-H-64points[62, 105]	635.8M	18.7	33.3	20.0	6.1	8.5	35.7	15.6	51.8	34.1	11.3	25.5	54.3	28.3	60.7
DFF-GenRCNN-R101[186]	194.6M	15.5	40.6	9.0	1.3	5.2	26.7	17.1	50.2	21.5	2.8	10.7	36.3	24.0	54.6
FGFA-GenRCNN-R101[185]	198.3M	21.4	51.6	14.9	4.9	11.6	36.3	27.5	55.0	28.6	10.6	19.9	45.7	35.2	60.0
RDN-GenRCNN-R101[30]	169.5M	24.7	51.2	21.6	7.5	17.6	37.9	28.6	61.5	34.3	12.4	27.0	50.8	43.9	64.0
MEGA-GenRCNN-R101[22]	172.5M	26.2	61.2	19.2	8.1	21.0	42.7	32.1	52.5	35.2	11.8	31.4	52.8	39.6	54.2
DiffusionVID-DiffusionDET-R101[113]	101.4M	44.3	80.2	43.8	19.8	41.8	59.7	43.6	71.6	54.4	29.1	52.6	68.5	50.5	75.9
DiffusionVID-DiffusionDET-SwinB[113]	145.3M	46.4	83.5	45.3	24.1	42.2	59.9	**51.5**	68.5	56.3	32.8	52.9	69.5	**59.5**	**76.9**
QueryDet-RetinaNet-R50[156]	39.3M	19.8	50.0	11.0	10.6	27.9	22.2	3.7	0.0	29.3	23.5	37.7	32.4	4.7	0.0
DetectoRS-CascadeRCNN-R50[103]	124.0M	45.1	74.9	48.4	20.4	45.3	58.5	36.7	62.3	52.0	24.6	53.8	66.6	40.5	63.7
SegTrackDetect [69, 70]	**20.3M**	**53.1**	85.0	**57.7**	**34.3**	**52.1**	61.7	37.4	70.9	**62.1**	**40.8**	**62.6**	69.9	43.8	75.5

including a 6M-parameter YOLOv7 Tiny detector and a lightweight `ROI Estimator`, the system achieves AP=22.8%, AP_{50}=53.8%, AR_{75}=15.2%, AP_{vt}=53.9%, AP_t=60.0%, AR_{500}=73.3%, AR_{vt}=73.4%, AR_t=75.8%, while using significantly fewer parameters than larger models. Only the InternImage-L [140] architecture surpasses it in AP_t, but it requires over 12 times more trainable parameters (params in Tab. 5.1). Performance on the *tiny* object subset is notably affected by dataset characteristics: this class constitutes only 2% of test annotations and is concentrated in a single video sequence, where annotation granularity and object scale differ substantially from the rest of the dataset. Most DroneCrowd annotations loosely enclose entire persons from a distance, whereas the *tiny* subset includes bounding boxes around smaller regions, such as heads, in closer views. Transformer-based models like InternImage may be better suited to such variability due to their higher capacity and complexity, which likely contributes to their advantage in this narrow subset. Among the compared models, only InternImage (both base and large variants), YOLOv7 (standard and e6e), YOLOv10, DiffusionVID, and DetectoRS reach acceptable detection quality on this dataset. Other architectures fail to adequately handle the fine-grained instances characteristic of DroneCrowd. DetectoRS was evaluated with anchor settings adapted to match the dataset's object size distribution, but performance remained significantly lower than that of SegTrackDetect. SAM, which was evaluated without training, struggled due to its sparse sampling strategy. While increasing the number of sampled points did improve recall, it rendered inference impractically slow, taking hours per image in some configurations. Additional difficulties arose during training of YOLOv10m and YOLOv10x at a reduced resolution of 640×640 px. These issues likely resulted from the way the original training pipeline handles image resizing, which further diminishes the visibility of already small objects. To ensure a fair comparison, neither the training procedures nor the model architectures were altered.

Overall, the results demonstrate that SegTrackDetect delivers strong and consistent performance across a wide range of metrics, despite the extreme challenge posed by datasets dominated exclusively by tiny objects. Its ability to maintain high detection quality while keeping computational demands low highlights its suitability for real-world applications, especially in scenarios where both precision and efficiency are critical.

On the SeaDronesSee validation set (Tab. 5.2), the proposed method outperforms all compared

approaches in detecting very tiny and tiny objects, achieving the highest scores in both Average Precision ($AP_{vt} = 34.3\%$, $AP_{t} = 52.1\%$) and Average Recall ($AR_{vt} = 40.8\%$, $AR_{t} = 62.6\%$). It also delivers strong results in localization accuracy, with AP and AP_{75} values exceeding those of significantly larger models. For large objects, performance remains highly competitive, with $AP_{l} = 70.9\%$ and $AR_{l} = 75.5\%$, only marginally lower than the best-performing methods by 0.9 and 1.4 percentage points, respectively. These results confirm the effectiveness of a hybrid sliding-window and resizing strategy.

The most notable performance drop occurs in the medium-sized object category. Manual inspection reveals that this is likely due to inconsistencies in annotations, including imprecise or poorly placed bounding boxes. Furthermore, a subset of the validation set contains horizontal medium-sized human figures not present in the training data, which presents a generalization challenge for the lightweight detector. Larger models appear to generalize better in such cases, showing a smaller performance degradation. Nevertheless, all methods exhibit reduced performance for medium objects, with both AP_{m} and AR_{m} lower than the scores for smaller object categories - an unusual trend given their typically higher detectability. Interestingly, the video detection model DiffusionVID achieves the highest performance for medium objects, suggesting that leveraging temporal cues may help counteract the effects of noisy annotations.

Overall, SegTrackDetect achieves the highest Average Precision (53.1%) and Average Recall (62.1%) across all evaluated models on the multi-scale SeaDronesSee dataset, while maintaining one of the smallest model sizes (20.3M parameters). This complements earlier findings that demonstrated SegTrackDetect's strong performance specifically in tiny-object-only scenarios, highlighting its versatility across a wide range of object scales. The results demonstrate a balance between detection quality and computational efficiency, confirming the system's robustness for both tiny and multi-scale detection tasks. Such performance reinforces SegTrackDetect's suitability for deployment in resource-constrained environments.

5.4.2 Qualitative results

Detection examples from the SegTrackDetection system on the SeaDronesSee dataset are presented in Fig. 5.3. Black rectangles represent the detection windows, while each object is marked with a bounding box and a predicted class - each class is visualized in a distinct color. For visual clarity, the original class label "swimmer with life jacket" has been replaced with "swimmer+PFD", where PFD stands for Personal Flotation Device. The confidence threshold for visualization was set to 0.1. For visual clarity, the predicted and estimated ROIs are omitted in this visualization to avoid overcluttering; these are shown and discussed in the Ablation Study further in this chapter.

As illustrated by these examples, the detections are of very good quality for both tiny objects (as in sequence 001) and larger ones. There are no false positive partial detections in the overlapping regions of the detection windows, which validates the effectiveness of the `Global Filtering Block`. Redundant or unnecessary detection windows are rare; in these examples, they are present only in sequences 000 and 016. In the former, an empty detection window appears due to a still-active tracklet corresponding to an object that has moved beyond the

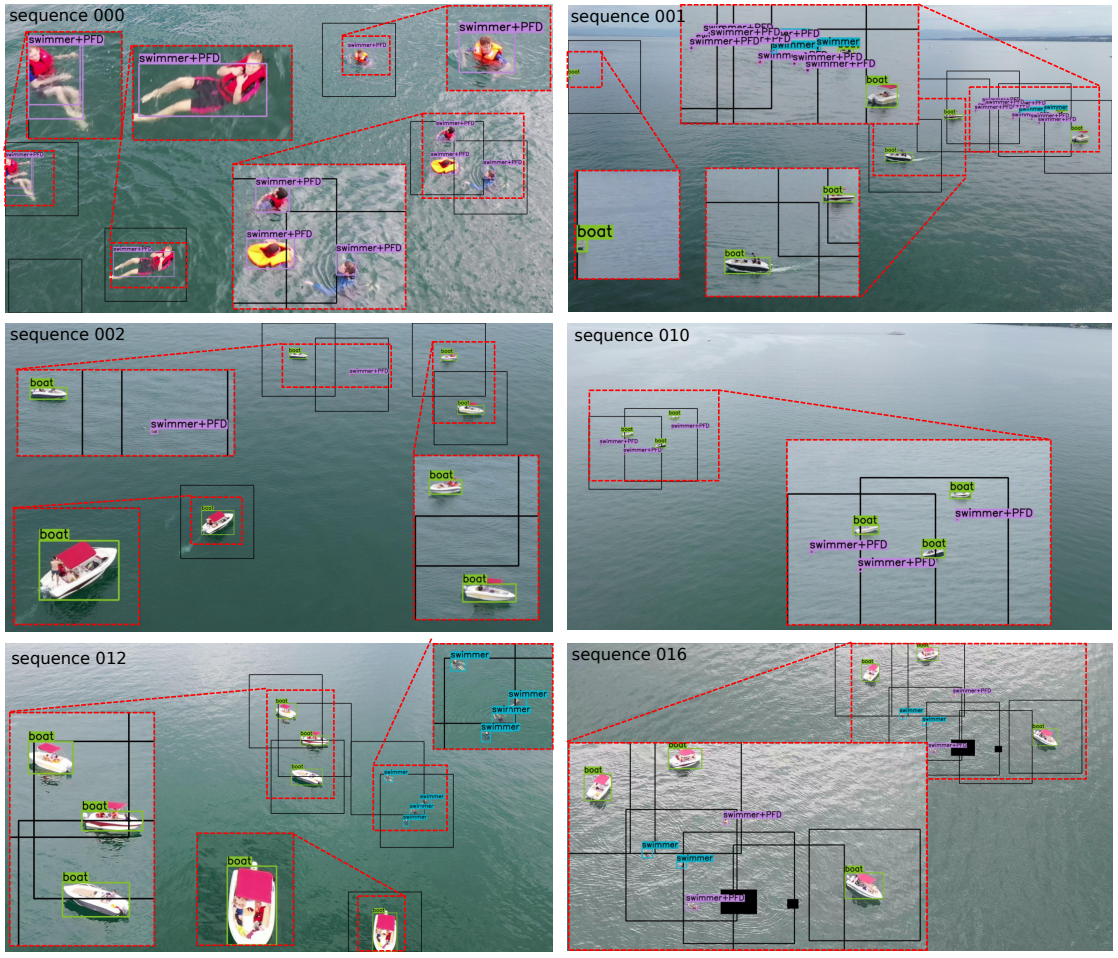


FIGURE 5.3: Examples of detections in the SeaDronesSee validation subset and SegTrackDetect with dual ROI selection strategy.

image borders. In the latter, a black rectangle, used as an anonymization tool in the original data, is mistakenly identified as foreground by the ROI selection module, resulting in a detection window being placed over it. However, the detector correctly outputs no detections within this window. These false-positive detection windows are infrequent and do not affect the overall detection quality. Their occurrence has a minimal impact on processing time, primarily because they are rare and therefore only marginally increase the number of windows processed. Since the majority of detection windows are placed based on valid ROIs, the few unnecessary windows that are occasionally introduced do not significantly alter the computational load or slow down the system.

In addition to the detection examples across different sequences, Fig. 5.4 presents selected detection errors on the SeaDronesSee dataset. To avoid obscuring the tiny object instances with additional data, the full input images are shown without any annotations, while zoomed-in regions display the ground-truth labels (on the left or top) and the corresponding detections (on the right or bottom).

In the SeaDronesSee validation subset, medium-sized swimmers present a range of detection challenges (see Fig. 5.4a). These instances are often confused with swimmers wearing personal flotation devices (PFDs), localized with inaccurate bounding boxes, frequently capturing only

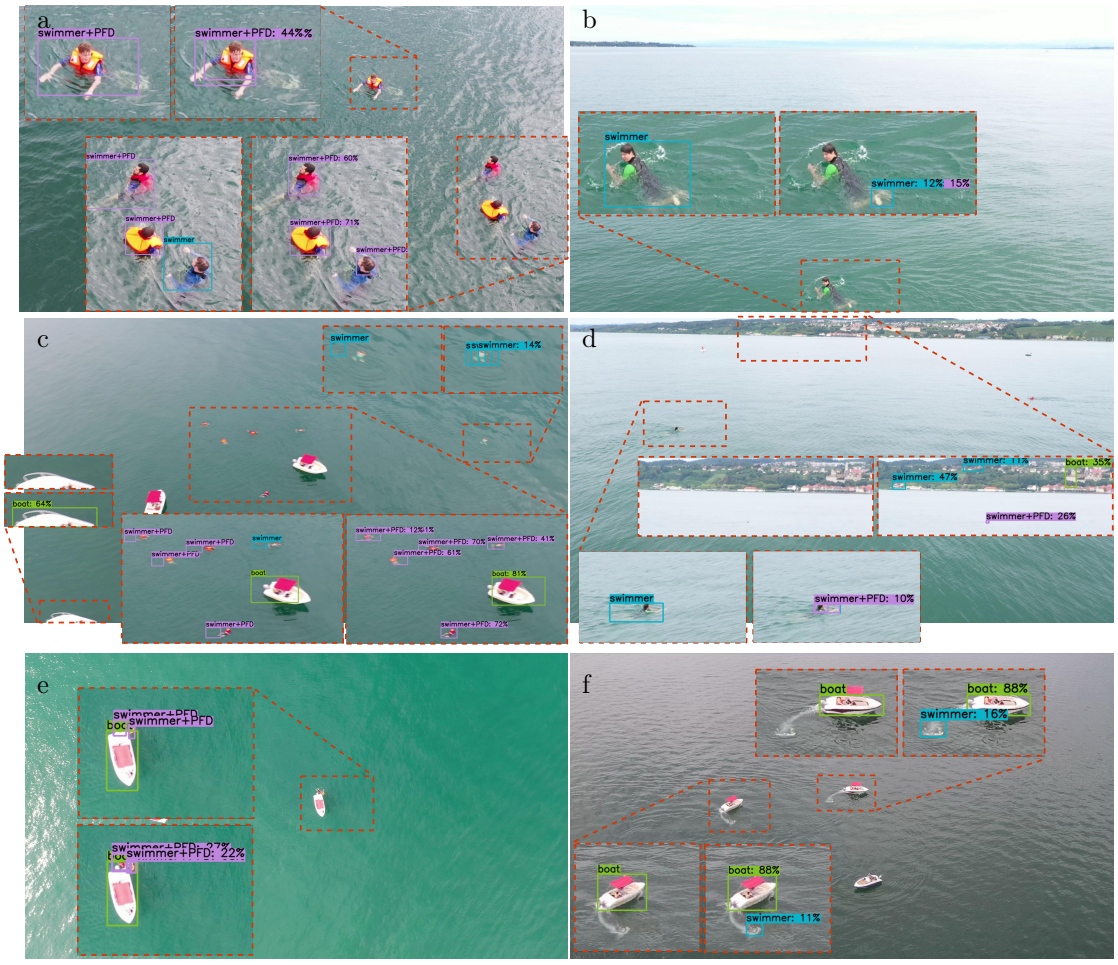


FIGURE 5.4: Illustrative examples of detection errors generated by SegTrackDetect on the SeaDronesSee [132] validation set. For each case, the original image is presented alongside zoomed-in views that display both the ground-truth and the corresponding predicted bounding boxes.

the head instead of the full body, or result in redundant detections. In Fig. 5.4b, detection is limited to a small fragment of a clearly visible swimmer. Such errors appear to be linked to an imbalance in the training data, where the vast majority of annotated swimmers are distant and barely visible, typically reduced to small head-sized regions. This distribution mismatch reduces the model's ability to generalize to nearer, full-body instances. The shallow depth of the detection network may also constrain its adaptability to such scale and perspective variations. Expanding the diversity of swimmer scales during training through targeted augmentation could help enhance robustness. Furthermore, the duplicated detection in Fig. 5.4a remains after NMS due to a low IoU between boxes. Since both detections fall within a single detection window, they bypass the `Overlapping Box Suppression (OBS)` module, which only addresses inter-window overlaps.

False positives can also result from visual patterns resembling human motion. In Fig. 5.4f, the white wake left behind a moving boat is incorrectly classified as a swimmer. This likely stems from similarities in texture and structure between wakes and areas surrounding active swimmers. A possible contributing factor is that many boats in the training set are static, limiting the model's exposure to dynamic water features. Figure 5.4c illustrates a frame captured during

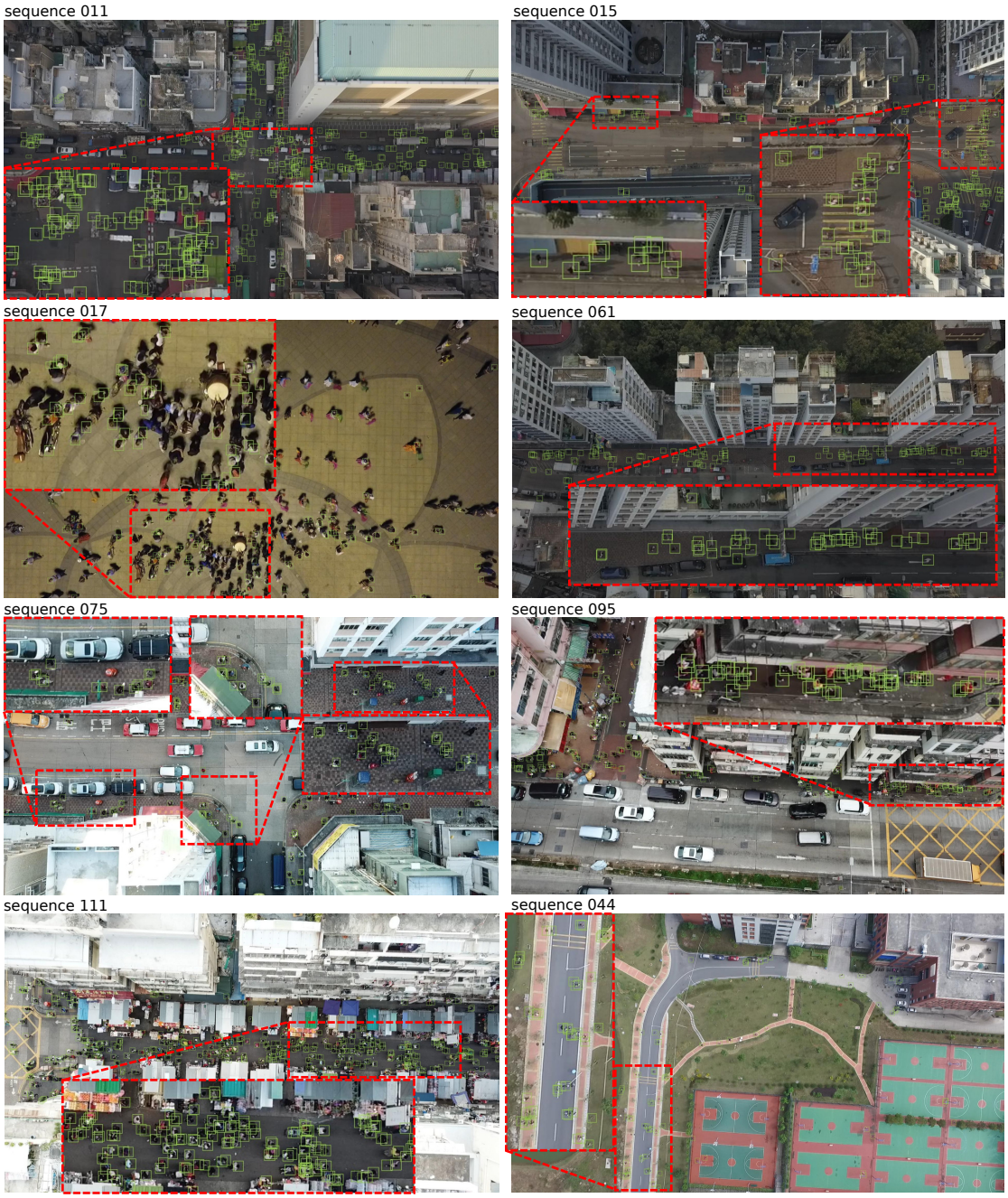


FIGURE 5.5: Examples of detections in the DroneCrowd test subset and SegTrackDetect with dual ROI selection strategy.

sudden drone acceleration, producing significant motion blur. In this instance, bounding boxes appear shifted or duplicated, while the ground-truth labels lack spatial precision. It is difficult to determine whether these anomalies are detection-related or the result of imprecise annotations. Other sequences with rapid motion exhibit similar labeling inconsistencies. In the same image, a partially visible boat is correctly detected but unlabeled; since this object is not annotated, it is not considered a detection error. Complex backgrounds further contribute to false positives, as seen in Fig. 5.4d. In such cases, the `ROI Estimation` stage misidentifies background regions as foreground, which then leads to incorrect detections. This issue may be addressed through enhanced data augmentation or loss functions that emphasize error-prone regions. Including

more background-only samples in the training data could also improve discrimination. Tiny swimmer instances in Fig. 5.4d and Fig. 5.4e result in duplicate detections that are not eliminated by NMS. A similar problem is apparent in the DroneCrowd dataset (Fig. 5.6a and Fig. 5.6c), where extremely small targets amplify the limitations of fixed-threshold suppression. These cases could benefit from adaptive IoU thresholds or the adoption of suppression methods that explicitly account for object scale.

Similarly to SeaDronesSee, qualitative examples and detection errors for the DroneCrowd dataset are analyzed in detail. The detection results are presented in Fig. 5.5, while common error cases are illustrated in Fig. 5.6. Due to the high complexity of backgrounds in this dataset, detection windows are not shown in the visualizations, and zoomed-in regions are provided only for selected areas because of the dense scenes. The overall detection quality on the DroneCrowd dataset is notably strong in urban environments featuring streets and buildings, where most people are accurately detected and false negatives are infrequent. However, detection performance tends to decline in industrial or semi-rural scenes, such as those in sequences 017 and 044, where an increased number of false positives in background regions and occasional missed detections occur. These challenges are explored further in the following discussion of detection errors. This discrepancy is likely due to the urban focus of the training dataset, which can be effectively addressed through targeted data enhancement strategies designed to enhance the model's exposure to less common landscapes, thus improving robustness in diverse environments. Despite the overall high detection quality, particularly in urban scenes, the DroneCrowd dataset presents several challenges. False positives (in terms of both ROIs and detections) are common in areas with visually complex backgrounds (Fig. 5.6b, Fig. 5.6c), where darker textures or cluttered regions are often misclassified as foreground by the `ROI Estimator` and subsequently misdetected as people. As observed in the SeaDronesSee dataset, such errors may be mitigated by emphasizing difficult samples during training and applying stronger penalization to difficult regions in both the estimation and detection models. False negatives also occur, though less frequently, and typically result from missed ROIs (Fig. 5.6a and Fig. 5.6c). However, many of these regions are later recovered through temporal association via the tracker component in the `ROI Fusion Module`. Similar to SeaDronesSee, perspective distortions and variations in object distance further complicate detection (Fig. 5.6b). In certain sequences, people appear closer to the camera and are annotated using head-only bounding boxes, whereas other sequences use full-body labels. As a result, successful detection in such cases often depends on the visibility of the head. Additional difficulties are observed under low-light conditions, particularly in night scenes (Fig. 5.6d), which are underrepresented in the training set. These limitations suggest that integrating more diverse augmentations, especially targeting perspective shifts and nighttime imagery, could significantly enhance robustness in future iterations.

In general, the SegTrackDetect framework demonstrates strong performance in diverse scenarios and datasets, effectively balancing detection accuracy and computational efficiency. Quantitative results on both DroneCrowd and SeaDronesSee highlight the system's ability to accurately detect objects at multiple scales, from tiny to large, while maintaining the smallest parameter count among compared state-of-the-art methods. Qualitative results show that the system performs well in both densely and sparsely populated environments. Most detection errors are related to

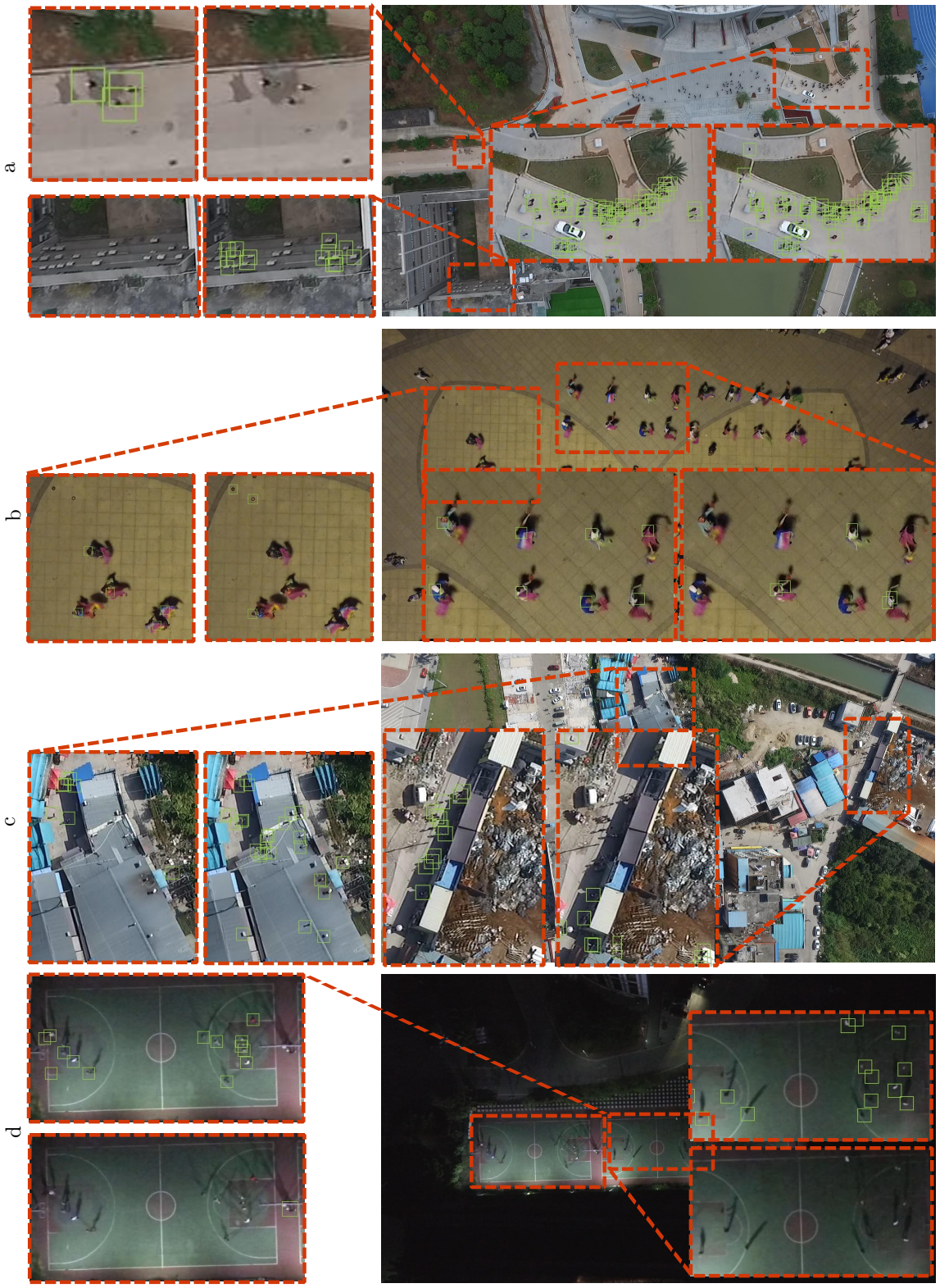


FIGURE 5.6: Examples of detection errors generated by the proposed SegTrackDetect method on the DroneCrowd [147] test set. Each sample displays the original image with zoomed-in areas that show both the ground-truth and detected bounding boxes. Class labels are omitted for clarity, as the dataset consists of only one object category.

specific dataset issues like inconsistent annotations or limited variety in training data. The following section presents a detailed ablation study isolating the impact of the `ROI Prediction Module`. This study demonstrates that adding the module speeds up inference and improves detection results, supporting its important role in the SegTrackDetect design.

5.5 Ablation Study

Several architectural choices adopted in the final version of the SegTrackDetect system were already extensively discussed in Chapter 4. These include the following:

- the design of the `ROI Estimation` training pipeline, which uses ground-truth binary masks as exact representations of bounding-box detection labels, validated to yield superior results compared to dilation, based mask generation approaches,
- the input resolution of the `ROI Estimator`, revisited here to enable an in-depth analysis of the quality-vs-speed trade-offs, particularly in relation to the addition of the tracking-based `ROI Prediction Module`,
- the sorting-based filtering method in the `Detection Windows Proposal Block`, which was shown to offer the best balance between detection quality and processing speed, as demonstrated in Chapter 4,
- the large-ROI handling strategy, which combines crop-and-resize (to preserve contextual integrity) with a sliding-window approach within ROIs (to preserve features of tiny objects),
- the `Global Filtering Block`, which ensures that the system output remains free of false positives.

While the ablation study in this chapter focuses exclusively on the impact of the `ROI Prediction` component within the SegTrackDetect framework, the remaining architectural decisions were either analyzed in Chapter 4 or are addressed in Chapter 6, which discusses the `Global Filtering Block` and evaluates the impact of the `OBS` and `OBM` algorithms on detection quality.

The first part of this section compares three `ROI Fusion` approaches: estimation-only, prediction-only, and the combined method proposed in this Chapter. This comparison includes both qualitative and quantitative evaluations utilizing the SeaDronesSee dataset. In the second part, both estimation-only and combined methods are analyzed across several input resolutions. This experiment directly demonstrates that the inclusion of the `Prediction Branch` enables a significant reduction in the input size required by the `Estimation Block`, without degrading detection quality. As a result, the overall complexity of the system is reduced, making it more efficient and better suited for deployment on resource-constrained platforms. All variants of the SegTrackDetect system (estimation-only, prediction-only, and combined estimation and prediction) use identical configurations and hyperparameters for all other modules. This ensures that any observed differences in performance can be directly attributed to the ROI selection method, and specifically highlights the benefits of incorporating the tracking-based `ROI Prediction Branch`.

This section contributes not only a direct evaluation of the proposed fusion strategy, but also provides strong empirical evidence supporting Auxiliary Thesis #2: "Incorporating an additional ROI source, such as an object tracker, into the ROI-based object detection system further improves detection quality, inference speed, and computational efficiency over state-of-the-art tiny object detection methods". The experiments clearly demonstrate that combining prediction and estimation not only improves detection quality but also enables substantial reductions in input resolution and system complexity, confirming the practical benefits of multi-source ROI generation.

5.5.1 Different ROI Selection Strategies

This section presents a comparison of various ROI selection strategies. Specifically, it evaluates the estimation-only and prediction-only approaches against the proposed combined strategy, which fuses ROI masks generated by both segmentation and tracking modules. Extensive qualitative and quantitative experiments demonstrate that combining both sources yields the highest detection performance, outperforming either method used in isolation.

Figure 5.7 illustrates a qualitative comparison between segmentation-based ROI selection (`ROI Estimation`), tracking-based ROI selection (`ROI Prediction`), and the proposed fusion approach (`ROI Fusion`). The fusion strategy consistently produces superior results. Segmentation-based methods tend to miss tiny objects due to limited sensitivity, whereas tracking-based approaches fail to detect newly appearing objects, as they depend solely on the regions of previously tracked objects initialized by a sliding-window at the beginning of the sequence and lack access to the global scene context. By integrating both ROI sources, the system successfully recovers tiny objects missed by the segmentation network and captures new objects that would otherwise be ignored by the tracker. In the visualization, the estimated ROIs from segmentation (shown in orange in the first and third columns of Fig. 5.7) remain consistent, as the same image is processed regardless of the fusion strategy. In contrast, tracking-based ROIs (blue regions) can vary over time due to differences in detection window placements, leading to differences between the second and third columns of Fig. 5.7. Notably, both the segmentation-only and fused ROI approaches use a reduced input resolution for the `ROI Estimation Module` (128×192 px), highlighting the system's efficiency at lower image scales.

Rows (a), (b), (d), and (f) in Fig. 5.7 illustrate how the proposed `ROI Fusion` strategy effectively combines the strengths of both ROI sources. In Fig. 5.7a, a group of tiny swimmers is correctly recovered thanks to tracking, while the large boat is successfully detected through `ROI Estimation`. This highlights a key limitation of the tracking-only approach, which being initialized at the start of the sequence, lacks global context and cannot detect newly appearing objects. Similarly, in Fig. 5.7b, a swimmer missed by the segmentation network is still detected due to successful tracking across frames. Rows (b) and (c) further emphasize that newly appearing objects are consistently missed by tracking-based ROI selection, but can be captured through `ROI Estimation`. For instance, in both rows, two boats are correctly identified as foreground objects via segmentation. However, in Fig. 5.7f, one of the boats is missed due to the limited input resolution used for the `Estimation Network`. This issue could be mitigated by increasing

FIGURE 5.7: Example results comparing segmentation-based detection (first column), tracking-based detection (second column), and the proposed method that integrates ROIs from both segmentation and tracking modules (third column). Blue regions indicate predicted ROIs, orange regions represent estimated ROIs, thin black boxes show detection windows, and thick black boxes denote detected objects. All examples are drawn from the SeaDronesSee [132] dataset. In the proposed architecture, the `ROI Estimation` and `Prediction Branches` operate in coordination rather than independently; therefore, the ROIs shown in the third column are not a simple union of those in the first and second columns.

the input size, though it would come at the cost of higher computational load. Figure 5.7d provides a clear example of the `ROI Fusion` method outperforming both individual components. In the first column, a small boat is missed by the segmentation network, and as it was not present at the start of the sequence, the tracker also fails to capture it. However, the fusion approach correctly detects all relevant objects by combining the partial segmentation mask with

TABLE 5.3: Detection metrics across three ROI approaches: estimation-only, prediction-only, and the proposed fusion method combining both segmentation- and tracking-based ROI masks. All experiments were conducted on the SeaDronesSee validation subset with a confidence threshold of 0.1. When applicable, the `Estimator` input size was set to 128×192.

Method	**AP**	**AP_{50}**	**AP_{75}**	**AP_{vt}**	**AP_t**	**AP_s**	**AP_m**	**AP_l**	**AR**	**AR_{vt}**	**AR_t**	**AR_s**	**AR_m**	**AR_l**
ROI Estimation	48.5	75.9	53.5	23.7	46.5	62.1	36.1	69.9	55.3	27.4	55	68.7	40.5	73.7
ROI Prediction	35.7	56.6	37.9	28.7	41.4	26.6	15.3	51.3	42.2	33.4	50.2	29.3	19.8	52.2
ROI Fusion	**51.9**	**81.9**	**57.1**	**31.7**	**50.1**	**62.1**	**36.2**	**70.0**	**59.3**	**37.1**	**59.1**	**68.7**	**40.6**	**74.8**

information retained by the tracker. For very small objects, segmentation often produces correct ROIs in some frames; tracking then ensures these regions are consistently forwarded to the detector, enabling reliable detection, illustrated by the boat in the last column of Fig. 5.7d, which is missed in both individual methods. The example in Fig. 5.7e represents a closed environment where no new objects enter the scene. In this case, the tracking-only method achieves complete detection, and the fused approach performs identically. Nevertheless, it still outperforms the segmentation-only strategy, highlighting the added robustness provided by tracking. Conversely, in Fig. 5.7f, the segmentation network accurately captures all ROIs, so the fused output matches the segmentation-only result. However, the tracking-only approach fails to detect several objects that were not present at initialization. Additionally, segmentation masks offer resilience during sudden changes in object shape or orientation, where predicted locations from the tracker may initially be inaccurate until it adjusts to the new motion trajectory.

These observations underscore the importance of combining both segmentation- and tracking-based ROI sources. The fusion approach not only enhances detection robustness and coverage but also enables the use of smaller input resolutions for the `ROI Estimator`, critical for meeting real-time processing requirements. In time-constrained applications, where fast inference is essential and computational resources are limited, this synergy allows the system to maintain high detection performance without sacrificing speed or efficiency.

A quantitative comparison of the three ROI selection strategies is presented in Tab. 5.3, with all hyperparameters kept constant across methods. In the prediction-only configuration, the tracker was initialized using detections obtained from a uniform sliding-window placement over the first three frames. Each of the three ROI strategies is evaluated using both general detection metrics (AP, AP_{50}, AR) as well as Average Precision and Average Recall scores with respect to object scales. For both, the estimation-only and fused approaches, the `ROI Estimator` operated on an input size of 128×192 pixels, and a confidence threshold of 0.1 was applied across all configurations. As shown in Tab. 5.3, the fused ROI strategy consistently outperforms both individual methods in all metrics. These results support the qualitative findings shown in Fig. 5.7, demonstrating that the combined approach leverages the strengths of both estimation- and prediction-based methods to achieve superior performance, particularly for tiny and multi-scale object detection in low-resolution `Estimator` settings. Across all object scales, the proposed method delivers the highest detection quality. It maintains performance comparable to the estimation-only approach for larger objects (see Fig. 5.8), while significantly improving AP_{vt} and AP_t by 8.0 and 3.6 percentage points, respectively, over the estimation-only baseline, and by 2.9 and 8.7 points compared to the prediction-only variant. Figure 5.8 illustrates how the

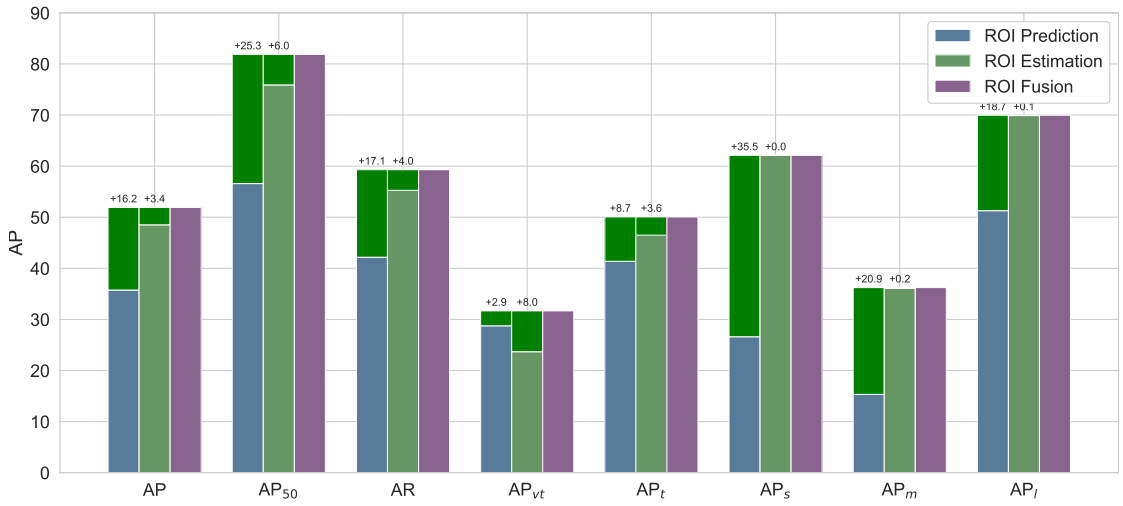


FIGURE 5.8: Comparison of selected detection metrics across three ROI approaches: estimation-only, prediction-only, and the proposed fusion method combining both segmentation- and tracking-based ROI masks. Gains from using the fused method are highlighted in green, alongside the percentage-point advantage it provides over each individual ROI approach. All experiments were conducted on the SeaDronesSee validation subset with a confidence threshold of 0.1. When applicable, the `Estimator` input size was set to 128×192.

fused method combines the accuracy of estimated ROIs for larger objects with the precision of tracking-based ROIs for the smallest objects, consistently producing the highest-quality results. While this subsection focuses exclusively on detection quality, the next subsection extends the analysis to multiple `Estimator` input resolutions, ranging from tiny to large, and examines the trade-offs between inference speed and detection performance. There, it is shown that the fused ROI method also enables substantial reductions in input resolution without compromising detection quality.

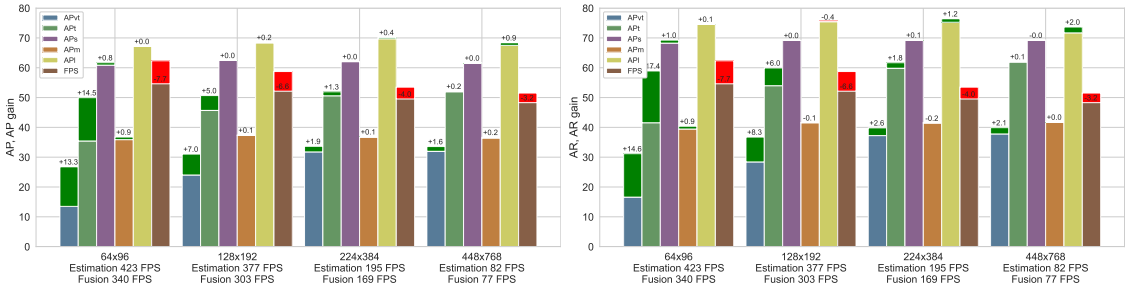
5.5.2 Impact of the ROI Fusion on Detection Quality and Processing Speed

The proposed `ROI Fusion` strategy combines information from two complementary sources: a segmentation-based `ROI Estimation Network` and a tracking-based `ROI Prediction Module`. This hybrid approach enhances the detection of objects of various sizes in dynamic scenes while maintaining high processing efficiency. To evaluate its impact, experiments were conducted using different input resolutions of the `ROI Estimation Network`, analyzing both detection quality and processing speed. Although increasing resolution improves segmentation accuracy, it also introduces significant inference overhead. By incorporating tracking-based `Prediction`, the system maintains high detection quality even when segmentation masks are coarse, enabling the use of smaller input resolutions with minimal accuracy loss. All experiments were performed using an NVIDIA RTX 4090.

Table 5.4 presents a comparative analysis for four input resolutions (64×96, 128×192, 224×384, and 448×768 px) of the segmentation network. The UNet architecture with a ResNet18 backbone was used for `ROI Estimation`. The metrics include AP, AR, and FPS for both segmentation-only (superscript e) and fusion-based (superscript f) ROIs. The inclusion of the tracking-based

TABLE 5.4: Impact of adding the `ROI Prediction Module` to the `ROI Estimation Network` on object detection metrics for the SeaDronesSee dataset. AP, AR, and FPS refer to Average Precision, Average Recall, and frames per second (for both the whole system and the ROI selection process). Superscripts e and f denote metrics using only the segmentation-based ROIs and the fused ROIs (segmentation + tracking), respectively. Values in parentheses indicate the performance gain due to the `Prediction Module`. Experiments run on GPU (NVIDIA RTX 4090), confidence threshold of 0.1.

size	AP^e	AR^e	FPS^e_{sys}	FPS^e_{roi}	AP^f	AR^f	FPS^f_{sys}	FPS^f_{roi}
64x96	41.5	47.0	60.9	423	51.4 (+9.9)	59.2 (+12.2)	50.0 (-10.9)	340 (-83)
128x192	48.5	55.5	54.9	377	52.4 (+3.9)	60.2 (+4.7)	47.7 (-7.2)	303 (-74)
224x384	52.3	60.5	49.6	195	53.3 (+1.0)	61.9 (+1.5)	43.3 (-6.3)	169 (-26)
448x768	52.9	61.5	44.4	82	53.3 (+0.4)	62.0 (+0.5)	40.8 (-3.6)	77 (-5)



(A) Average Precision (B) Average Recall

FIGURE 5.9: Effect of including the `ROI Prediction Module` on Average Precision (a), Average Recall (b) across object sizes, and processing speed measured by FPS (a, b), compared to segmentation-only ROIs, evaluated at different input resolutions.

`ROI Prediction Module` consistently improves detection performance, especially at lower input resolutions and for smaller object sizes, where segmentation masks are less accurate.

Although the tracking module introduces a minor computational overhead, its impact is significantly smaller than that of increasing the input resolution. All configurations maintain real-time performance, with the fastest speeds reaching 60.9 FPS for the `Estimator` alone and 50.0 FPS for the full fused system. The increase in total processing time for lower-resolution configurations is not primarily due to slower ROI selection (FPS_{roi}) caused by the inclusion of the tracking component, but rather to the increased number of detection windows resulting from more complete ROI coverage. This enhanced coverage leads to a higher number of correctly localized tiny objects, thereby improving overall detection quality at the cost of slightly reduced processing speed. This is evident from the differences between ROI selection speed (FPS_{roi}) and overall system speed (FPS_{sys}). The modest impact of the tracking component on FPS_{roi} confirms its efficient implementation, while the larger drop in FPS_{sys} reflects the processing cost of correctly handling a greater number of detections.

The ROI fusion approach achieves performance comparable to the high-resolution (448×768) segmentation-only baseline while operating at a significantly lower resolution (128×192) and achieving nearly four times higher processing speed (FPS_{roi}). Specifically, it yields 52.4% AP, 60.2% AR, and 303 FPS, compared to 52.9% AP, 61.5% AR, and 82 FPS for the high-resolution segmentation-only method. Further downscaling to 64×96 input resolution increases the ROI selection speed to 340 FPS, with only a modest decline in detection quality (51.4% AP, 59.2%

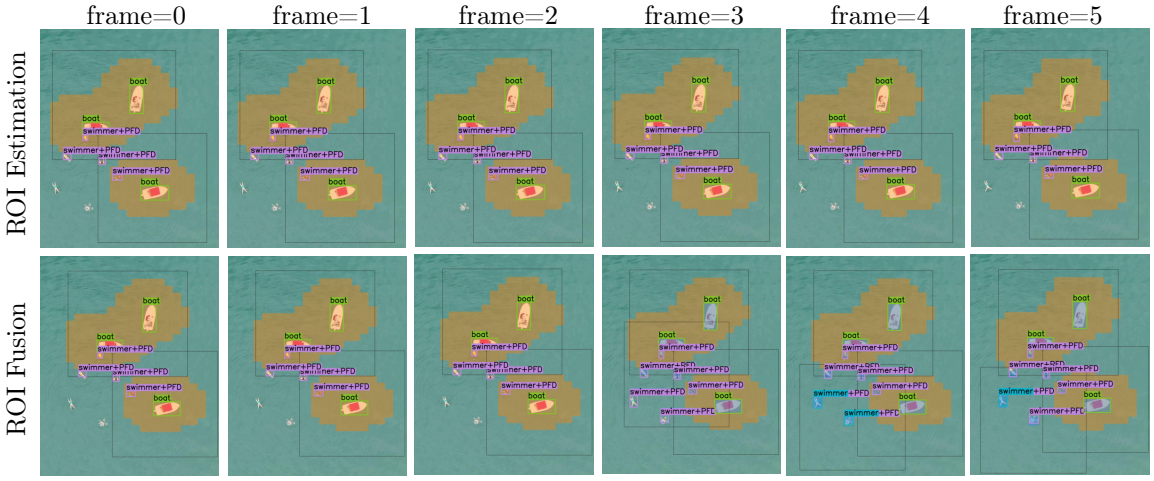


FIGURE 5.10: Qualitative comparison between segmentation-only and fusion-based ROI selection, illustrating the recovery of missed tiny objects. Orange regions indicate ROIs generated by the segmentation module, while blue regions represent predicted ROIs from the tracking component. The fusion approach enables the system to identify and retain previously omitted targets across consecutive frames.

AR). At the system level, overall processing speed improves from 44.4 FPS for the segmentation-only baseline to 47.7 FPS and 50.0 FPS for the fused system at 128×192 and 64×96 input resolutions, respectively.

As shown in Fig. 5.9, the tracking-based `ROI Prediction Module` is particularly effective in restoring detection performance for very tiny and tiny objects. While segmentation quality for smaller objects degrades significantly at lower input resolutions, the fusion approach recovers both AP and AR across all object sizes. Importantly, the proposed method does not degrade detection quality under any configuration. Its impact is consistently beneficial, introducing only a slight increase in ROI selection time, most notably at lower resolutions where segmentation masks are less precise. At higher resolutions, where the initial masks are already of high quality, the added overhead is minimal. This slight increase in processing time is primarily due to the greater number of correctly identified regions that require further processing, rather than inefficiencies in the fusion pipeline itself. The modest difference between ROI-level and system-level FPS further confirms the efficiency of the implementation. The ROI fusion method achieves an effective balance between detection quality, computational efficiency, and processing speed, substantially improving performance for small objects while maintaining strong detection for medium and large ones. By integrating the `ROI Prediction Module`, the ROI selection module operates at 303 FPS using low-resolution inputs (128×192), yet delivers detection quality comparable to the highest-performing segmentation-only configuration, which achieves similar accuracy, but runs at just 82 FPS.

Figure 5.10 illustrates how the tracking-based `ROI Prediction Module` complements the segmentation-based `ROI Estimation` by recovering regions missed in low-resolution masks. The figure shows six consecutive frames from a SeaDronesSee sequence. The segmentation-only method consistently fails to capture the smallest objects, particularly those near the edges of detection windows. In contrast, the fusion-based approach incorporates predicted ROIs (blue) that gradually expand coverage to include previously missed objects. As these peripheral regions

are identified and tracked, the system introduces new detection windows that persist across subsequent frames, allowing reliable identification of small targets. This process demonstrates the temporal consistency introduced by tracking, which is critical for detecting objects that are otherwise lost due to segmentation inaccuracies.

The experiments presented in this chapter demonstrate that combining segmentation-based `ROI Estimation` with tracking-based `Prediction` significantly enhances small object detection in high-resolution video streams, while allowing for smaller computational demand and faster inference speeds compared to high resolution estimation-only baselines. The proposed `ROI Fusion Module` improves detection quality across all object sizes while maintaining real-time performance, even for larger segmentation inputs. Through both quantitative and qualitative evaluation, it was shown that the fusion strategy mitigates the limitations of coarse segmentation masks without imposing a substantial computational burden. Ablation studies confirm that the added tracking component is most beneficial at low resolutions and for very small objects, while having negligible impact on performance at higher resolutions. This trade-off between speed and accuracy makes the approach well-suited for real-time applications with constrained resources. The experiments also directly demonstrate that integrating the `ROI Prediction Module` improves processing speed and computational efficiency by enabling the use of lower-resolution `Estimators`, achieving better-than-state-of-the-art performance, and providing strong empirical support for Auxiliary Thesis #2.

5.6 Conclusions

Real-time detection of small objects in high-resolution video is particularly challenging in UAV-based robotics applications, where both accuracy and efficiency are critical under limited computational resources. To address this, the ROI estimation-based detection system introduced in Chapter 4 is extended by incorporating a lightweight tracking component. This addition enables the use of lower-resolution inputs to the `Estimator Network`, significantly reducing computational demand and improving inference speed while maintaining high detection quality. By combining low-resolution semantic segmentation with tracking-driven `ROI Prediction`, the system efficiently identifies relevant regions and supports accurate multi-scale detection. Extensive evaluation demonstrates that the proposed approach outperforms several state-of-the-art detectors, achieving higher accuracy with significantly fewer parameters and delivering robust performance across different object sizes and datasets.

Compared to other approaches in the literature, the proposed dual-ROI strategy leverages data-driven region selection, which better preserves the detection context than uniform tiling strategies, such as the one proposed in [151], which tend to fragment larger objects. The proposed method effectively handles both small and large objects by using a hybrid large ROI handling technique: sliding-windows within each ROI capture tiny objects, while a crop-and-resize strategy maintains the integrity of larger objects. As a result, the SegTrackDetect system delivers high-quality performance in both tiny-only and multi-scale object detection scenarios. The method has been validated across varied detection settings and proved effective in both densely populated datasets, such as DroneCrowd, and sparse environments like SeaDronesSee. Notably, the

latter scenario is often overlooked by ROI selection methods that rely solely on clustering or density-based heuristics [34, 75, 157], which focus detection on regions containing multiple objects. These methods also typically require annotations that are not straightforward to derive from standard detection labels, for example, cluster-level or density-based annotations, which introduces a degree of subjectivity. This challenge is mitigated in the proposed approach by relying on segmentation masks generated directly from detection labels and a non-trainable tracking component. Finally, while some related methods rely on additional networks for scale estimation, the proposed method eliminates this need. Thanks to the dual large ROI handling strategy, tiny objects are detected through sliding-window operations, and larger objects, often fragmented by such windows, can still be detected accurately in resized detection windows.

The proposed approach outperforms several state-of-the-art detectors on the DroneCrowd dataset, achieving superior results with the smallest number of model parameters (20.3M). In the multi-scale detection scenario of the SeaDronesSee dataset, it demonstrates strong performance for very tiny and tiny objects, while maintaining competitive accuracy for larger objects. These results highlight the effectiveness of the dual large ROI handling strategy in addressing common challenges faced by window-based detection systems. Overall, the method emphasizes the critical role of object tracking in enhancing the detection of small and tiny objects. By integrating Regions of Interest (ROIs) derived from both tracking and segmentation, the system achieves higher detection accuracy than other approaches [62, 103, 113, 136, 140, 156], while maintaining computational efficiency. Its robustness in multi-scale scenarios is particularly valuable in dynamic, real-world environments, where rapid changes in object scale due to sensor or object movement are common. The SegTrackDetect system is designed to handle such variability effectively. Moreover, its lightweight architecture and reduced parameter count make it especially well-suited for deployment in resource-constrained applications, such as mobile robotics.

These findings validate the third thesis of this work, demonstrating that incorporating an additional ROI source, specifically object tracking, not only enhances detection quality but also improves the computational efficiency beyond what is achievable with existing state-of-the-art methods in tiny object detection. To directly attribute these improvements to the tracking component, an ablation study was conducted. This study focused on isolating the impact of tracking on both detection quality and inference speed. The SeaDronesSee dataset was used as a representative example to validate these effects in a realistic, multi-scale detection scenario.

Several architectural choices in SegTrackDetect were previously validated in the preceding chapter, which focused exclusively on `ROI Estimation`. That analysis covered the training pipeline, input resolutions for the segmentation-based ROI component, filtering methods for ROI proposals, the large ROI handling strategy, and the rationale for introducing a `Global Filtering Block` tailored specifically to window-based detection. In this chapter, the emphasis shifts to evaluating the impact of the `ROI Prediction` component.

The proposed `ROI Fusion Module` was extensively evaluated in comparison to purely segmentation-based and purely tracking-based ROI selection strategies. Notably, the `ROI Estimator` was used with a reduced input resolution of 128×192 in both the estimation-only and fusion-based configurations. This setup enabled a direct analysis of how the inclusion of tracking contributes to recovering regions containing tiny objects that are often missed by the low-resolution `Estimator`

alone. The results, both qualitatively and quantitatively, confirm that the fused strategy is the most robust. It outperforms both individual modules and, in some cases, even surpasses naive aggregation of their independent outputs. This is particularly evident in sequences where the segmentation-based `Estimator` detects regions with small objects inconsistently; in such cases, the tracking module fills in the missing areas by leveraging temporal continuity. In contrast, the `Estimation Module` proves to be more effective in identifying newly appearing objects that a tracking-only system would miss. These findings demonstrate that the strengths of both modules are complementary, and their integration significantly enhances detection reliability across object sizes and motion patterns. These findings also demonstrate that reduced detection performance for tiny objects caused by lower-resolution segmentation can be effectively mitigated by incorporating the `ROI Prediction Module`. This directly supports the conclusion that tracking not only reduces computational complexity (by enabling lower-resolution input) but also improves detection accuracy for the smallest objects. These results provide strong validation for the second Auxiliary Thesis, which states that incorporating an additional ROI source, such as object tracking, enhances detection quality, inference speed, and computational efficiency beyond what is achievable using segmentation-based `Estimation` alone. While inference speed was not directly measured in this part of the study, it is addressed in the following section of the ablation study, which compares both `ROI Fusion` and `ROI Estimation` strategies across four different input resolution settings, offering a comprehensive analysis of detection quality and processing speed under each configuration.

To further assess the effectiveness of the `ROI Prediction Module`, an extended ablation study was conducted using four different input resolutions (64×96, 128×192, 224×384, and 448×768 pixels) on the SeaDronesSee dataset (Tab. 5.4). This analysis focused on how varying the resolution of the `ROI Estimation` input affects both detection quality, ROI selection speed, and inference speed, particularly when fused with the tracking-based `Prediction` component. The results show that integrating the tracking module with low-resolution segmentation allows the system to match the detection quality of high-resolution segmentation while significantly increasing the processing speed. For example, using the combined ROI strategy at 128×192 resolution, the system achieves 52.4% AP, 60.2% AR and an FPS_{roi} of 303 FPS on a NVIDIA RTX 4090. In contrast, a segmentation-only baseline using 448×768 resolution reaches a comparable 52.9% AP and 61.5% AR but operates at only 82 FPS. Importantly, the more pronounced decrease in system-level throughput (FPS_{sys}) in lower-resolution configurations does not stem from inefficiencies in the ROI selection process itself. Rather, it reflects the increased number of detection windows required to recover missed regions, particularly those containing tiny objects, which are more frequently overlooked when segmentation is performed at lower resolutions. The `ROI Prediction Module` effectively compensates for these omissions, ensuring more complete coverage and higher detection quality. At the same time, the relatively small drop in ROI module speed (FPS_{roi}) demonstrates that the tracking-based `Prediction Branch` introduces minimal overhead. This highlights its efficiency compared to simply increasing the input resolution of the segmentation network. By enabling accurate ROI recovery with minimal computational cost, the `Predictor` proves to be a lightweight yet powerful addition to the system, significantly boosting detection performance for small and very tiny objects without incurring the high cost of large input sizes. These findings demonstrate that tracking enhances the quality of tiny object

detection, especially when operating at reduced input resolutions, recovering regions missed by segmentation, and maintaining strong performance across all object sizes (see Fig. 5.9, Fig. 5.10).

This chapter introduced a lightweight, window-based detection system that integrates segmentation- and tracking-driven ROI selection. By fusing these complementary modules, the system achieves efficient detection of both large and tiny objects without increasing overall computational load. Experimental results confirm that this fusion strategy outperforms segmentation-only or tracking-only variants while maintaining real-time processing. These findings strongly support the second Auxiliary Thesis of this dissertation: that incorporating an additional ROI source not only improves detection quality, but also enhances computational efficiency, and processing speed. Tracking, in particular, enables the use of lower-resolution inputs without sacrificing accuracy, making the approach practical for high-resolution video streams in resource-constrained applications.

Future research may explore optimizing system complexity by experimenting with various detector backbones to find configurations that best balance detection accuracy and computational efficiency for tiny object detection. The modular design, motivated by deployment requirements, facilitates flexibility and adaptability in the system architecture. Independently, integrating the backbones of the `ROI Estimation Network` and the object detector could be pursued to reduce trainable parameters and accelerate both training and inference. Incorporating more advanced motion models, such as non-linear trajectory estimations, could further improve robustness in scenarios involving rapid camera or object movements common in drone-based sensing systems. Additionally, refinement of the `Global Filtering Block` through the application of `Overlapping Box Merging` is addressed in the following chapter.

Chapter 6

Detection Filtering Methods

6.1 Introduction

Numerous methods have been developed specifically for detecting small and tiny objects [24], with many relying on external guidance to focus the detector on small, relevant regions cropped from the original, typically high-resolution image. These focus-and-detect approaches [24] increase the relative size of target objects, making their features more prominent and thus easier to detect. While window-based methods are commonly used for small object detection, they may also introduce additional challenges. Partially visible objects and overlapping detection windows often lead to false positives and reduced detection quality, particularly for larger objects. Common strategy in object detection is to apply Non-Maximum Suppression (NMS) to eliminate redundant bounding boxes. However, in window-based settings, partial detections frequently exhibit low Intersection over Union (IoU) with full detections, allowing them to bypass NMS filtering. Furthermore, since NMS retains only the highest-confidence detection, it can mistakenly discard the most complete detection if it has a lower confidence score than a fragmentary detection. Additionally, if the object of interest is not fully visible in any detection window, either because it is larger than the window or due to inaccuracies in the retrieved Region of Interest (ROI), both false positive and false negative partial detections may be introduced.

Fragmented detections in window-based systems give rise to several challenges. Figure 6.1c illustrates a situation where partial detections appear alongside a correct full detection. Because these fragments have low overlap (IoU) with the full detection, standard NMS fails to suppress them, resulting in false positives. Figure 6.1d shows another common problem, where an object is only partially visible in several overlapping windows, and no single window captures it fully. In such cases, merging the fragmentary detections can lead to a more complete and accurate result. Another error scenario is shown in Fig. 6.1a, where an object is only partially detected within a single window. Since all fragments originate from the same window, merging across windows is not possible, and the failure is due to the detector itself missing parts of the object. This type of error cannot be corrected through post-processing. Subplot Fig. 6.1b depicts a correct detection for reference.

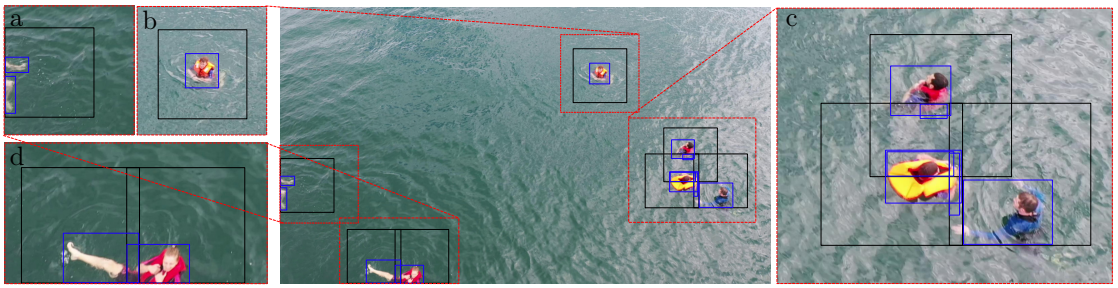


FIGURE 6.1: Examples of detection cases discussed in this work: (a) fragmentary detections within a single detection window that cannot be merged or filtered; (b) a correct detection; (c) partial detections coexisting with a full detection, handled by the `Overlapping Box Suppression (OBS)` method; and (d) partial detections without any corresponding full detection, addressed by the proposed `Overlapping Box Merging (OBM)` method. Black rectangles represent the detection windows, while detected objects are shown in blue.

To address these challenges (Fig. 6.1c and Fig. 6.1d), two complementary post-processing algorithms are proposed: `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)` [64, 65, 69]. Most existing window-based object detection systems rely on NMS as the default post-processing step to filter redundant detections [34, 75, 142, 157, 169]. However, NMS is not designed to handle inter-window inconsistencies, often resulting in either missed detections or the persistence of fragmented boxes (see Fig. 6.2b). To overcome these limitations, `OBS` and `OBM` were developed as specialized alternatives. `OBS` eliminates partial detections when a more complete detection is available (Fig. 6.2c), while `OBM` merges fragmented detections across overlapping windows when no full detection exists (Fig. 6.2d). Together, they provide a robust replacement for NMS in the context of window-based detection, effectively addressing fragmentation between detection windows. The two methods are integrated into a unified `Global Filtering Module`. They are architecture-agnostic and can be applied to any window- or ROI-based detection framework, such as [43], as long as detection results and window coordinates are accessible. However, it is important to note that `OBS` and `OBM` are specifically designed to resolve inter-window detection errors and do not address errors occurring within a single window (such as Fig. 6.1a).

The effectiveness of the proposed `Filtering Module` is evaluated on the SeaDronesSee dataset [132], using the video object detection framework introduced in [70], which builds on the architecture detailed in Chapter 5. Experimental results, both qualitative and quantitative, indicate that the proposed method consistently surpasses traditional Non-Maximum Suppression (NMS) in terms of precision and recall. Illustrative examples in Fig. 6.2 highlight how the two components, `OBS` and `OBM`, operate in practice. `OBS` assumes that at least one sub-window is likely to capture the full object and uses both the detection results and the coordinates of detection windows to eliminate redundant partial detections. In contrast, `OBM` targets situations where no single window contains the complete object. It progressively merges detection fragments from overlapping windows, improving the final output by addressing both missed detections and false positives.

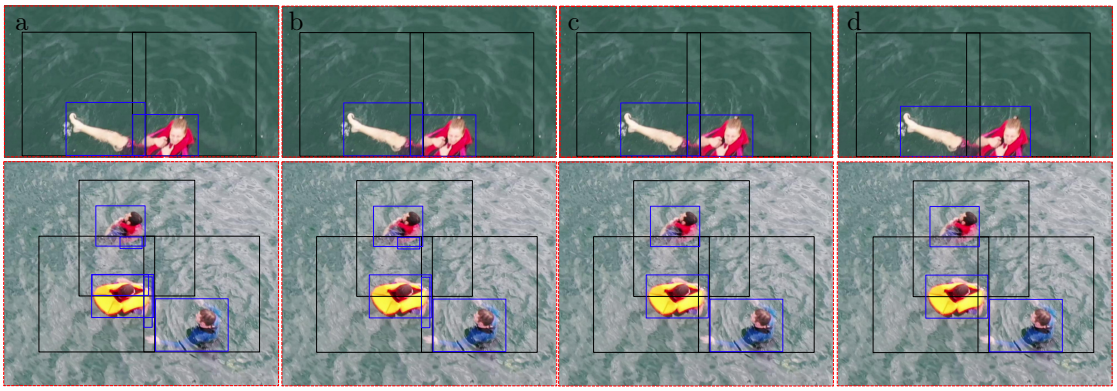


FIGURE 6.2: Example application of the `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)` algorithms. When combining detections from multiple sub-windows, intersecting detection windows and large objects can introduce false positives. Without global filtering, this results in poor detection quality (a). While NMS removes some redundant detections (b, bottom row), it struggles with small partial detections. `OBS` effectively removes all partial detections (c, bottom row), but objects that are partially visible across all windows lack a complete detection. `OBM` addresses this by merging such partial detections into a single, complete detection (d), delivering a false-positive-free output from the detection system.

6.2 Contribution

Traditional Non-Maximum Suppression (NMS) [115] aims to eliminate redundant detections by retaining only the bounding box with the highest confidence score when multiple boxes overlap significantly. While effective, this strategy can lead to the suppression of valid detections, especially in cluttered scenes. To address this limitation, Soft-NMS[13] was introduced as a refinement. Instead of removing overlapping boxes outright, Soft-NMS progressively lowers their confidence scores, reducing the likelihood of suppressing relevant but overlapping instances, particularly for foreground objects that dominate the visual field. Further advancements include the use of learning-based approaches. For instance, a deep reinforcement learning framework was proposed in [101], where an attention mechanism is optimized to dynamically select region proposals. This learned policy replaces the traditional greedy approach by enabling better placement of detection windows, thereby improving localization and object coverage. The time required for these filtering steps has also been questioned. Shapira et al. [119] reveal how inefficient NMS configurations can be exploited, resulting in excessive processing delays. To address this, parallelized versions like the one in [171] have been developed, which speed up the filtering and can fully replace NMS at all stages of the detection pipeline. In contrast, this work proposes postprocessing methods that are independent of the detection approach used, making them compatible with any object detection backbone or proposal generation method. The focus is on tackling a key challenge specific to window-based detection pipelines: fragmented and redundant detections caused by overlapping sub-windows. NMS-like methods struggle in window-based detection settings because fragmentary detections often have low IoU with full detections. As a result, these partial detections are not effectively suppressed, leading to redundant or incomplete predictions remaining in the output.

Several alternative techniques to standard NMS have emerged in the literature. The Confluence method [120], for instance, introduces a proximity-based strategy that selects the most central detection within a group and suppresses nearby redundant boxes based on a Manhattan distance-like metric. In contrast, Gilg et al. [41] present a probabilistic framework that directly estimates the likelihood of duplication for each detection and adjusts confidence scores accordingly, avoiding explicit suppression. Some methods attempt to compensate for the suppression of occluded objects. The Suppression and Enhancement (SE) algorithm [98] increases the scores of detections partially hidden from view, preserving them in the final output. Feature-based filtering, such as FeatureNMS [117], leverages deep features to identify duplicates by comparing learned embeddings. Additionally, the Weighted Boxes Fusion algorithm [122] takes a different route: instead of discarding any detections, it merges them into a single prediction by calculating a weighted average based on confidence scores. This strategy has shown better results than both NMS and Soft-NMS on popular datasets such as COCO and Open Images. However, it was originally designed to merge outputs from multiple detection models rather than handling partial detections within window-based approaches.

This work presents a unified filtering system that both removes false positives caused by overlapping detection windows and merges detections of the same object appearing in multiple windows. While many recent window-based methods [34, 75, 142, 157, 169] use some form of NMS to filter detections from different windows, they often do not properly handle the problem of fragmentary detections. A conceptually related strategy was introduced in [71], which shares similarities with `Overlapping Box Suppression (OBS)`. However, their approach depends solely on Intersection over Union (IoU) as a suppression criterion. The proposed `OBS` method extends this by integrating IoU with the detection confidence and the bounding box area to prioritize the removal of partial, uncertain detections. Unfortunately, due to the lack of an official implementation of [71], a direct comparison could not be conducted.

This chapter presents experimental results supporting Auxiliary Thesis #3, which states that the proposed postprocessing techniques effectively mitigate a key limitation of window-based detection systems like fragmentary detections that degrade overall detection performance. The main contributions discussed in this section are as follows:

- introduction of the `Overlapping Box Suppression (OBS)` algorithm, which filters partial false-positive detections more effectively than traditional Non-Maximum Suppression (NMS), achieving higher precision and recall in multi-scale window-based object detection,
- development of the `Overlapping Box Merging (OBM)` algorithm, designed to merge partial detections when no complete detection is available, thereby enhancing the `Global Filtering` stage,
- a detailed evaluation comparing `OBS` to both NMS and unfiltered detection outputs, highlighting its superior performance and including an ablation study on key `OBS` hyperparameters,
- an in-depth analysis of `OBM`, demonstrating consistent improvements in both precision and recall, independent of the global filtering method or specific parameter settings.

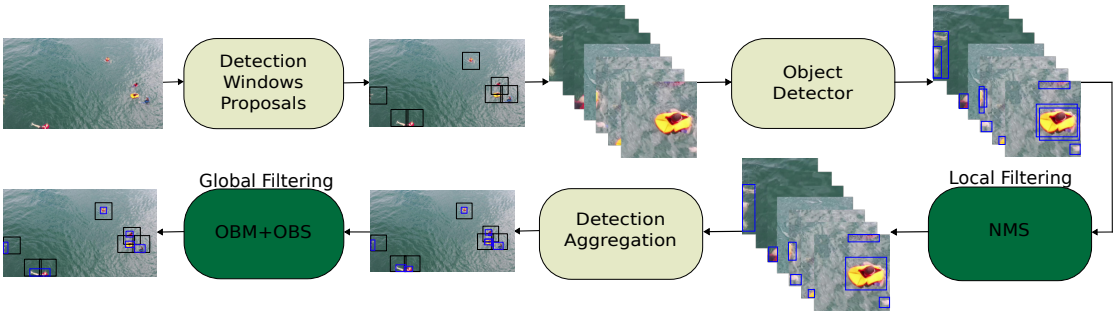


FIGURE 6.3: Simplified window-based detection pipeline. Detection occurs within independent windows, with detections filtered twice - first with NMS-style local filtering at the window level, then with `Global Filtering` at the image level. This study compares NMS and `OBS` for `Global Filtering` and examines the impact of the `OBM` algorithm on detection quality. NMS is always applied as postprocessing within separate subwindows.

6.3 Proposed method

Figure 6.3 presents a high-level overview of a typical window-based object detection pipeline. These systems generally begin with a pre-processing step that identifies candidate regions and discards background areas. Detection is then performed on multiple cropped sub-windows derived from the original image. The window proposals are often generated using a segmentation model that predicts Regions of Interest (ROIs), which are then resized or transformed to match the input dimensions required by the detector. Each sub-window is processed independently, with local Non-Maximum Suppression (NMS) applied within each window to eliminate redundant detections. The resulting outputs are then projected back to the coordinate space of the full image. In standard pipelines, a second round of NMS is typically used as a global filtering step to resolve duplicate detections originating from overlapping windows. However, this approach is often suboptimal due to the low IoU between fragmented and complete detections. To address this issue, the proposed `Global Filtering Block` introduces two complementary steps: `Overlapping Box Merging (OBM)`, which fuses partial detections into complete bounding boxes, and `Overlapping Box Suppression (OBS)`, a more robust alternative to NMS designed to suppress false positives arising from window overlaps.

The detection framework used for experiments [69] incorporates a Region of Interest (ROI) selection step into a high-resolution detection pipeline. Full-resolution detections are performed only within sub-windows chosen based on two sources: a binary segmentation network that identifies foreground regions in the current frame, and a SORT-based [9] object tracker that extrapolates object locations from previous frames. The detections from these independent windows are then aggregated and mapped back to the original image coordinates. Earlier versions of the system [67] employed Non-Maximum Suppression (NMS) alone for global filtering, to eliminate redundant detections arising from overlapping sub-windows. In this approach, the global NMS stage is replaced with `Overlapping Box Suppression (OBS)`, which additionally leverages window location information to suppress false positives more effectively, a common challenge in window-based detection. `Overlapping Box Merging (OBM)` is also incorporated to consolidate fragmented detections across window boundaries. Although the experimental setup builds on the system described in [69], the proposed `OBS` and `OBM` methods are model-

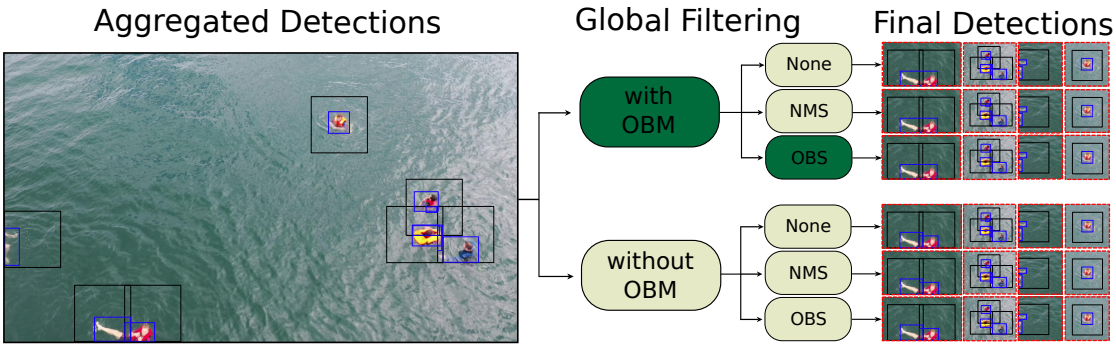


FIGURE 6.4: Detailed architecture of the experimental setup for the `Global Filtering Block` used in this study. The proposed approach, combining `OBM` and `OBS`, is highlighted in green. In total, six global filtering strategies are evaluated: merging with `OBM` followed by one of three options: no filtering (None), `OBS`, or NMS, as well as the same three filtering methods applied without merging. Zoomed-in regions from the final detection outputs are shown to clearly illustrate the differences between methods (full images are omitted for clarity).

and framework-agnostic and can be integrated into any window-based detection pipeline, as illustrated in the generalized architecture in Fig. 6.3. The exact implementation of the system used in the experiments is detailed in Chapter 5.

Figure 6.4 illustrates the experimental setup used to evaluate the proposed `Global Filtering` strategy. The primary method introduced in this study, combining `Overlapping Box Merging (OBM)` with `Overlapping Box Suppression (OBS)`, is highlighted in green. To isolate the effect of `OBM` independently of the filtering method, three configurations are included in which `OBM` is applied before one of three filtering options: None (no filtering), `OBS`, or NMS. These are compared against the same three configurations applied without `OBM`, resulting in a total of six global filtering configurations. This structure enables a detailed comparison of detection quality across all combinations, allowing to assess the individual and combined benefits of `OBM` and `OBS`. The zoomed-in output samples included in the figure visually demonstrate how each method handles partial and overlapping detections, providing insight into both the qualitative and quantitative differences observed in the experiments. The following sections present extensive experiments across multiple confidence and IoU thresholds, as well as varying hyperparameters for NMS, `OBS`, and `OBM`.

6.3.1 Overlapping Box Suppression Algorithm

Window-based detection systems are particularly well-suited for small and tiny object detection, as they increase the relative object size within each cropped region. However, overlapping detection windows frequently lead to duplicate and fragmented detections, which in turn raise the number of false positives and degrade overall precision. Traditional Non-Maximum Suppression (NMS) is effective at removing duplicate detections when objects are fully visible, but it fails to handle fragmented detections arising from partial views. When an object is only partially visible within a detection window, its resulting detection may have a low Intersection over Union (IoU) with the full detection of that object in another window, allowing it to bypass traditional Non-Maximum Suppression (NMS). Lowering the IoU threshold to catch these partial detections risks removing valid detections of nearby objects, especially in crowded scenes where objects are close

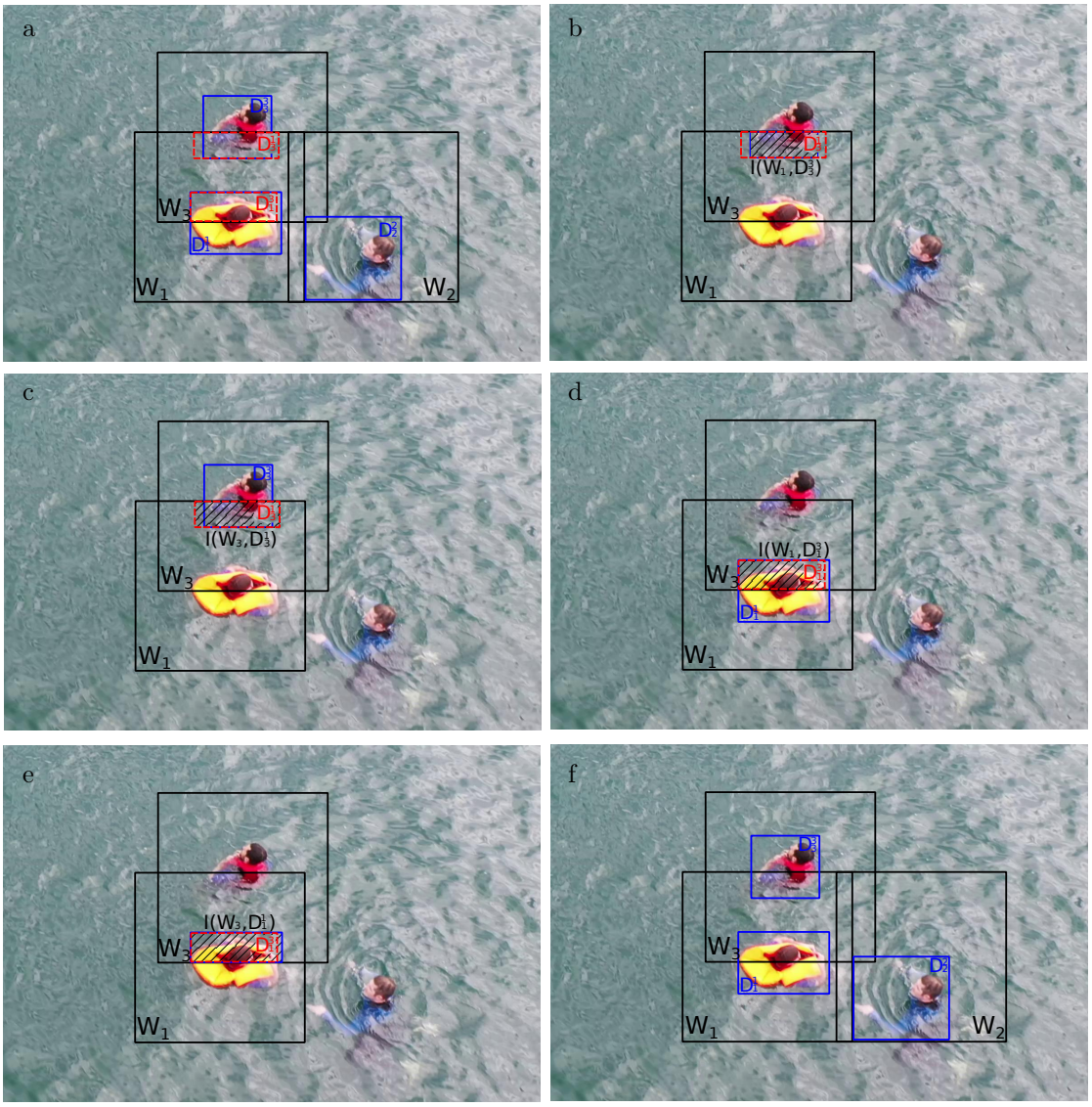


FIGURE 6.5: Simplified illustration of the `OBS` algorithm. Detection windows W_1 and W_3 contain partially visible objects, while W_2 contains only a fully visible object (a). Aggregating detections from windows W_1, W_2, and W_3 produces the correct object detections D_1^1, D_2^2, and D_3^3, along with partially visible detections D_1^3 and D_3^1. The intersection $I(W_j, D_i^k)$ between sub-window W_j and detected bounding box D_i^k (b,c,d,e) is calculated to eliminate false positives (f). Among all found intersections ($I(W_1, D_3^3)$, $I(W_3, D_3^1)$, $I(W_1, D_1^3)$, and $I(W_3, D_1^1)$), the intersections $I(W_1, D_3^3)$ and $I(W_3, D_1^1)$ exhibit high IoU with detections D_3^1 and D_1^3, respectively; thus, detections D_3^1 and D_1^3 are suppressed.

to each other, thereby increasing false negatives. To address these challenges, the `Overlapping Box Suppression (OBS)` algorithm uses the coordinates of detection windows to identify and remove partial detections. For each detected bounding box, `OBS` first calculates its intersection with every detection window, essentially determining the visible portion of that detection from the perspective of each window. Then, it computes the Intersection over Union (IoU) between these visible parts and the full detections found. Partial detections typically exhibit a high IoU with these visible parts, enabling `OBS` to recognize them as redundant. By filtering out detections with high overlap to visible fragments, the algorithm effectively suppresses fragmented detections while preserving the most complete and accurate bounding boxes. Moreover, partially visible

Algorithm 2 Overlapping Box Suppression

Require: $detections \in \mathbb{R}^{N\times 6}$ ▷ Detections (xmin, ymin, xmax, ymax, score, class)
Require: $windows \in \mathbb{R}^{N\times 4}$ ▷ Detection windows (xmin, ymin, xmax, ymax)
Require: $th \in (0, 1)$ ▷ IoU threshold for suppression (default $th = 0.6$)
Require: $filters \in$ {'all', 'iou', 'area', 'score'} ▷ Criteria for suppression
Require: $class\_agnostic \in \{True, False\}$ ▷ Apply filtering across all classes or within same class
Ensure: $filtered\_detections \in \mathbb{R}^{L\times 6}$ ▷ Filtered detections after suppression
Ensure: $filtered\_windows \in \mathbb{R}^{L\times 4}$ ▷ Filtered detection windows

Step 1: Compute Intersections Between Detections and Detection Windows
$W_u \leftarrow$ unique($windows$) ▷ Find unique detection windows $W_u \in \mathbb{R}^{M\times 4}$
$I \leftarrow$ intersect($detections[:, :4], W_u$) ▷ Find intersection $I \in \mathbb{R}^{N\times M\times 4}$

Step 2: Compute IoU Matrix Between Intersections and Full Detections
$ious \leftarrow$ empty tensor ($ious \in \mathbb{R}^{N\times M\times N}$) ▷ Initialize IoU tensor
for each $w \in W_u$ index i **do**
    $ious[:, i, :] \leftarrow$ box_iou($I[:, i, :], detections[:, :4]$) ▷ Compute IoU for each intersection
end for
if not class_agnostic **then**
    $same\_class \leftarrow (detections[:, 5] == detections[:, 5]^\top)$ ▷ Same-class mask, shape: (N, M, N)
    $ious \leftarrow ious \times same\_class$ ▷ Filter only within same class
end if
$ious \leftarrow [ious > th]$ ▷ Filter to keep only IoUs greater than threshold th
$to\_del \leftarrow$ find where $ious > th$ ▷ Find detections candidates for removal $to\_del \in \mathbb{R}^{K\times 3}$
if no IoU greater than th **then**
    return $windows, detections$ ▷ Return original windows and detections if no overlaps
end if

Step 3: Create Cost Vector for Deletion Candidates
$costs \leftarrow$ empty vector of length K ▷ Initialize cost vector
for each detection candidate i in $to\_del[:, 2]$ **do**
    $conf \leftarrow 1 -$ normalize($detections[i, 4]$) ▷ Normalize confidence score
    $area \leftarrow 1 -$ normalize(area of detection[i]) ▷ Normalize area of the detection
    $iou \leftarrow$ normalize($ious[i]$) ▷ Normalize IoU for this detection
    cost_map $\leftarrow$ {'all' : mean($[conf, area, iou]$), 'iou' : iou, 'area' : $area$, 'score' : $conf$}
    $costs[i] \leftarrow$ cost_map[$filters$] ▷ Select the appropriate cost
end for

Step 4: Get Detections for Removal
$to\_del \leftarrow (to\_del, \text{cost})$ ▷ Append cost to deletion candidates
$to\_del\_ids \leftarrow$ unique sorted indices from to_del ▷ Remove detections based on $costs$

Step 5: Filter Windows and Detections
$filtered\_windows \leftarrow windows$[not in to_del_ids] ▷ Filter out detections to delete
$filtered\_detections \leftarrow detections$[not in to_del_ids] ▷ Filter out corresponding detections
return $filtered\_windows, filtered\_detections$ ▷ Return filtered windows and detections

objects appearing in multiple overlapping windows can produce several high-confidence detections. NMS typically retains only the highest-scoring detection, which is not always the most complete representation of the object. The `OBS` algorithm addresses this by using a combination of confidences, areas, and IoUs as filtering criteria to preserve the most complete detection.

The `OBS` algorithm is illustrated in Fig. 6.5, with its pseudocode detailed in Alg. 2. `OBS` functions by examining the intersections between detected bounding boxes and the detection windows and then calculating the Intersection over Union (IoU) for every detection-intersection region. Detections exceeding a specified high IoU threshold are suppressed, as they are likely redundant fragments of the same object. Specifically, consider a detection D_i^j located in sub-window W_i corresponding to object j. If this detection has a substantial overlap with the intersection $I(W_i, D_k^j)$, where D_k^j is another detection of the same object from a different sub-window W_k,

then D_i^j is classified as a false positive. This significant overlap indicates that both detections refer to the same object, and thus one is redundant. For example, in Fig. 6.5, the detections D_1^3 and D_3^1 show a high overlap with intersections $I(W_1, D_3^3)$ and $I(W_3, D_1^1)$, respectively, leading to their removal and leaving only the complete detections D_1^1, D_2^2 and D_3^3.

Unlike traditional NMS, which mainly depends on confidence scores, `OBS` incorporates a combined cost metric involving IoUs, bounding box areas, and confidence scores. It prioritizes removal of detections that are smaller, have lower confidence, and exhibit high IoU with partial views, thus ensuring that the most complete detection of each object is retained. This is done by computing a normalized cost vector for each detection that determines the order in which detections are suppressed. The full details of the algorithm can be found in Alg. 2.

This study explores different configurations of the cost vector and demonstrates that combining confidence, area, and IoU metrics yields better performance than using any single metric alone. A class-agnostic version of `OBS` proves more effective in filtering fragmentary detections, as partial detections tend to be misclassified. Furthermore, the effect of varying IoU thresholds on detection quality is analyzed, alongside a detailed comparison between `OBS` and traditional NMS filtering methods.

6.3.2 Overlapping Box Merging Algorithm

The `Overlapping Box Suppression (OBS)` method efficiently filters out false positives caused by overlapping detection windows and partial object views within those windows. However, `OBS` relies on the presence of a complete detection. In certain situations, only partial detections exist, either due to imprecise window proposals or because objects are larger than a single detection window and span multiple windows. To address this, the `Overlapping Box Merging (OBM)` algorithm was designed to combine partial detections coming from several sub-windows when a full detection is not available. This often happens when the Region of Interest (ROI) is bigger than the input size of the detector, a common scenario in dynamic environments where objects move quickly relative to the sensor. One common solution is to crop a large area and resize it to the detector's input size. While this works well for large objects, it may reduce the detail of smaller objects within the ROI. For regions with small objects, applying a sliding-window approach inside the ROI is preferable, as it preserves more detail. However, this introduces the problem of merging partial detections of larger objects spread across multiple windows, which `OBM` is specifically designed to solve. Moreover, `OBM` also handles cases where imprecise ROIs cause objects to appear partially in multiple windows without any complete detection in a single window. Although less frequent, such situations also benefit from the merging capabilities of `OBM`. To validate `OBM`'s utility, two approaches are examined for processing ROIs larger than the detector's input size: cropping and resizing versus using a sliding-window on the ROI. The sliding-window method increases the need for `OBM` to combine partial detections. By integrating `OBM` with `OBS` into a unified `Global Filtering` step, the system can maintain the fine details necessary for detecting tiny objects while effectively merging partial detections of larger objects. This combination provides a robust solution for multi-scale detection in high-resolution images.

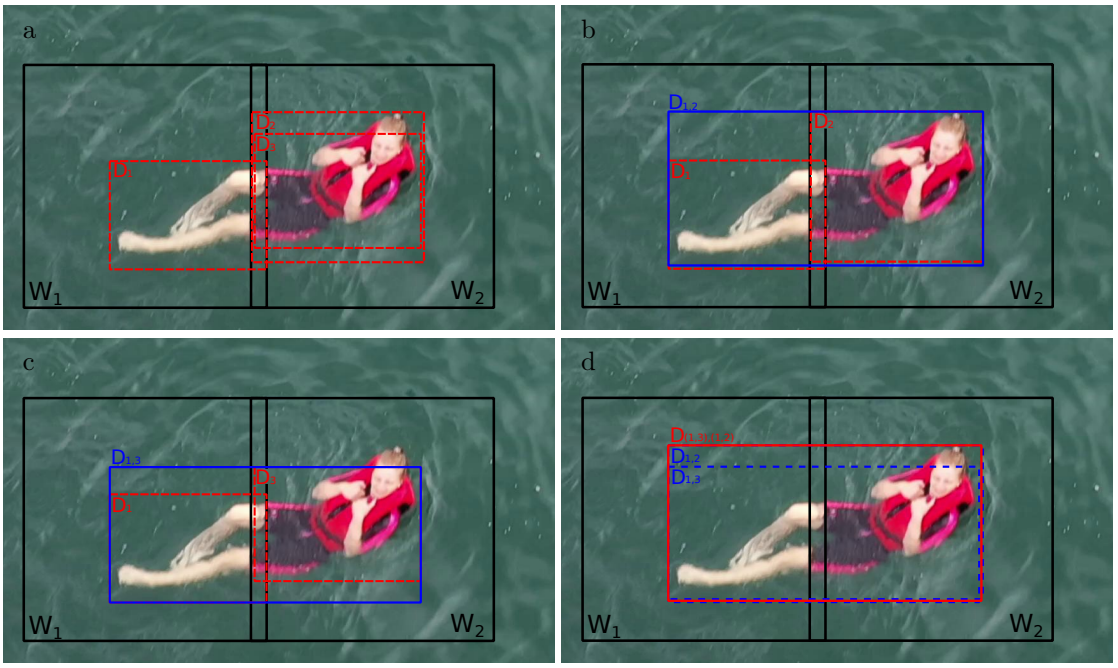


FIGURE 6.6: A simplified visualization of the `OBM` algorithm. Detection windows W_1 and W_2 include partial detections D_1, D_2, and D_3 (a). The algorithm iteratively merges detection pairs that overlap near the shared window boundaries (b,c,d), ultimately producing a single consolidated detection $D_{(1,3),(1,2)}$ (d).

The `OBM` algorithm operates by first identifying groups of overlapping detection windows, then locating clusters of overlapping detections within those window groups. It merges detection pairs if both are positioned near their respective window boundaries inside the overlapping region. This merging process happens iteratively, combining pairs step-by-step until a single consolidated detection remains. The final class label for the merged detection is computed as a weighted mean of the class probabilities and confidence scores from the individual partial detections. A tolerance of 10 pixels is used to decide whether a detection lies close enough to its window edge to be considered for merging. Figure 6.6 illustrates the `OBM` workflow. In the example, three partial detections, D_1, D_2, and D_3, appear in two overlapping detection windows, W_1 and W_2, without a full detection present. Within the cluster of overlapping windows W_1 and W_2, two separate groups of intersecting detections exist: one containing D_1 and D_2, and another containing D_1 and D_3. These pairs are merged into intermediate detections $D_{1,2}$ and $D_{1,3}$, which are then further merged to form the final detection $D_{(1,3),(1,2)}$. A detailed step-by-step description of the `OBM` algorithm can be found in Alg. 3.

In the next sections, the `OBM`'s performance is evaluated under three different global merging strategies: Non-Maximum Suppression (NMS), `Overlapping Box Suppression (OBS)`, and no global filtering (denoted as NONE). Both qualitative and quantitative analyses demonstrate `OBM`'s effectiveness in recovering fragmented detections, whether the input ROIs are processed by cropping and resizing or through a sliding-window approach. For detailed experimental setting of the `Global Filtering Block`, please refer to Fig. 6.4.

Algorithm 3 Overlapping Box Merging

Require: $detections \in \mathbb{R}^{N\times 6}$ ▷ Detections (xmin, ymin, xmax, ymax, score, class)
Require: $windows \in \mathbb{R}^{N\times 4}$ ▷ Detection windows (xmin, ymin, xmax, ymax)
Require: $th \in \mathbb{R}$ ▷ Tolerance for ending at the boundary (default $th = 10$)
Ensure: $merged\_detections \in \mathbb{R}^{L\times 6}$ ▷ Filtered detections after merging
Ensure: $merged\_windows \in \mathbb{R}^{L\times 4}$ ▷ Filtered detection windows

Step 1: Find clusters of overlapping windows
$md, mw \leftarrow \emptyset, \emptyset$ ▷ Initialize merged detections and windows
$W_u \leftarrow \text{unique}(windows)$ ▷ Find unique detection windows
$C_w \leftarrow \text{cluster}(W_u)$ ▷ Find clusters of intersecting detection windows

Step 2: Find clusters of overlapping detections within each $cw \in C_w$
$W_{cl} \leftarrow W_u[cw]$ ▷ Detection windows within the cluster
if $|\text{unique}(W_{cl})| = 1$ **then**
    continue ▷ Skip processing if only one window within cluster
end if
$D_{cl} \leftarrow detections[cw]$ ▷ Detections within the cluster
$C_d \leftarrow \text{cluster}(D_{cl})$ ▷ Clusters of intersecting detections

Step 3: Find merge candidates within each $cd \in C_d$
$D_{dcl} \leftarrow D_{cl}[cd]$ ▷ Detections within the detection cluster
$W_{dcl} \leftarrow W_{cl}[cd]$ ▷ Corresponding detection windows
if $|D_{dcl}| = 1$ **then**
    continue ▷ Skip processing if only one detection
end if
$I_w \leftarrow \text{intersection\_coords}(W_{dcl})$ ▷ Intersecting regions of detection windows
$merged \leftarrow \emptyset$ ▷ Initialize empty vector for merged coordinates
$indices \leftarrow \emptyset$ ▷ Initialize empty vector for merged indices
for $i = 1$ to $|I_w|$ **do**
    $idx_1, idx_2 \leftarrow indices[i]$ ▷ Get indices of intersecting detections
    $det_1, det_2 \leftarrow D_{dcl}[idx_1], D_{dcl}[idx_2]$ ▷ Retrieve detections
    if $ends\_on\_boundary(det_1[:4], I_w[i], th) \wedge ends\_on\_boundary(det_2[:4], I_w[i], th)$ **then**
        $merged\_coords \leftarrow \text{get\_coodinates}([det_1, det_2])$
        $merged \leftarrow merged \cup \{merged\_coords\}$ ▷ Add merged detection coordinates
        $indices \leftarrow indices \cup \{idx1, idx2\}$ ▷ Add merged detection indices
    end if
end for
$D_{dcl} \leftarrow D_{dcl}[indices]$ ▷ Detections to be merged
$W_{dcl} \leftarrow W_{dcl}[indices]$ ▷ Detection windows to be merged
$merged\_cluster\_coords \leftarrow \text{get\_coodinates}(merged)$ ▷ Compute coordinates of final merged detection within cd
$merged\_cls, merged\_conf \leftarrow \text{get\_weighted\_mean}(D_{dcl}[:,5], D_{dcl}[:,4])$ ▷ Compute final class and confidence as a weighted mean
$merged\_detection \leftarrow (merged\_cluster\_coords \cup [merged\_cls, merged\_conf])$ ▷ Get merged detection
$md \leftarrow md \cup merged\_detection$ ▷ Append to already merged detections
$merged\_window \leftarrow [min(W_{dcl}[:,0]), min(W_{dcl}[:,1]), max(W_{dcl}[:,2]), max(W_{dcl}[:,3])]$ ▷ Get merged window coordinates
$mw \leftarrow mw \cup (merged\_cluster\_coords \cup [merged\_cls, merged\_conf])$ ▷ Append to already merged windows

Step 4: Finalize results
if $\sim md$ **then**
    return *detections*, *windows* ▷ No further objects to be merged
end if
$detections \leftarrow md \cup nmd$ ▷ Combine merged detections and non merged detections
$windows \leftarrow mw \cup nmw$ ▷ Combine merged windows and non merged windows
Repeat merging process from Step 1 until no objects can be merged

TABLE 6.1: Object detection results on SeaDronesSee sequences using two different strategies for handling large ROIs. AP - Average Precision; TP - true positives, FP - false positives, FN - false negatives; P - precision; R - recall. All metrics are computed with a fixed IoU threshold of 0.5 and confidence threshold of 0.1. The ⋆ symbol indicates use of the `OBM` algorithm in the `Global Filtering Block`. Local NMS used an IoU threshold of 0.3.

(A) Cropping and resizing for large ROIs.

method	**AP**	**AP$_{50}$**	**TP**	**FP**	**FN**	**P**	**R**	**F1**
None	46.7	74.7	**42846**	23196	**4832**	64.9	**89.9**	75.4
None⋆	48.1	76.6	42813	19141	4865	69.1	89.8	78.1
NMS	48.5	80.8	41602	9275	6076	81.8	87.3	84.4
NMS⋆	49.2	80.8	41655	9085	6023	82.1	87.4	84.6
OBS	**52.3**	**83.4**	42652	9042	5026	82.5	89.5	85.8
OBS⋆	**52.3**	**83.4**	42682	**8825**	4996	**82.9**	89.5	**86.1**

(B) Sliding window for large ROIs.

method	**AP**	**AP$_{50}$**	**TP**	**FP**	**FN**	**P**	**R**	**F1**
None	42.6	69.6	40618	26289	7060	60.7	85.2	70.9
None⋆	47.8	76.1	**42536**	19905	**5142**	68.1	**89.2**	77.3
NMS	44.2	75.1	39281	12205	8397	76.3	82.4	79.2
NMS⋆	49.1	80.6	41439	9315	6239	81.6	86.9	84.2
OBS	47.5	77.5	40425	11832	7253	77.4	84.8	80.9
OBS⋆	**51.7**	**82.6**	42273	**9049**	5405	**82.4**	88.7	**85.4**

6.4 Experimental results

To evaluate the effect of the proposed `Global Filtering Block` and isolate the individual contributions of `OBS` and `OBM`, a series of experiments was conducted as outlined in Fig. 6.4. The object detection system used in this study is based on the prior work [69], previously described in detail in Chapter 5. All experiments were performed on validation sequences from the SeaDronesSee dataset, using NMS as the local filtering method. Local NMS was applied in a class-aware manner with an IoU threshold of 0.3 and a confidence threshold of 0.1.

To investigate the impact of different `Global Filtering` strategies, six configurations are considered: three without `OBM` (NMS, `OBS`, and no global filtering), and three with `OBM` combined with each of these global strategies (`OBM`+NMS, `OBM`+`OBS`, and `OBM`+None). Global NMS and `OBS` were applied in a class-agnostic manner with an IoU threshold of 0.1. For `OBS`, the cost matrix was calculated using normalized confidence scores, IoU values, and bounding-box areas. The metrics reported in Tab. 6.1a and Tab. 6.1b reflect the best-performing configurations for each strategy, selected based on the F1 score. These experimental results are further analyzed and discussed in the following section. Two strategies were evaluated for processing large ROIs that exceed the detector's input size: crop-and-resize (Tab. 6.1a) and a sliding-window approach (Tab. 6.1b). The sliding-window setting introduces more fragmented detections, providing a suitable context to assess `OBM`'s ability to merge partial detections. In both tables, the ⋆ symbol indicates configurations where `OBM` was used within the `Global Filtering Block`. Each ROI handling strategy was tested in six configurations (see Fig. 6.4), totaling 12 experiments.

Regardless of the method used to handle large ROIs, both NMS and `OBS` generally improve

detection performance across most metrics compared to no global filtering (None), with the exception of true positives (TP), false negatives (FN), and recall (R). When aggregating detections from multiple independent subwindows, overlapping regions can lead to partial false-positive detections. As a result, both NMS and `OBS` yield substantial improvements in AP, AP_{50}, false positives (FP), precision (P), and F1 score. As presented in Tab. 6.1a and Tab. 6.1b, both methods consistently increase precision by effectively reducing false positives, although their effect on recall is less uniform. In particular, `OBS` reduces false positives by 61% and 55% in Tab. 6.1a and Tab. 6.1b, respectively, while causing only a slight increase in false negatives (4% and 3%) and a negligible reduction in true positives (less than 1%). In contrast, NMS introduces a higher number of false negatives and demonstrates a more limited ability to suppress false positives compared to `OBS`. Incorporating the `OBM` algorithm into the `Global Filtering Block` (denoted by *) leads to further gains in true positives, false positives, and false negatives.

As shown in Tab. 6.1a and Tab. 6.1b, `OBS` consistently achieves higher F1 scores than NMS by effectively reducing false positives while preserving true positives. Without `OBM`, `OBS` improves the F1 score by approximately 10 percentage points compared to the absence of global filtering, and by approximately 8% when combined with `OBM` (denoted by * in the tables). Moreover, `OBS` has minimal impact on recall, underscoring its advantage over NMS in preserving true positives. Its superior ability to suppress high-confidence partial false positives is reflected in consistently higher precision. These results confirm that while both `OBS` and NMS enhance detection performance by reducing false positives, `OBS` delivers a better balance of precision and recall, regardless of the large ROI handling strategy used.

The benefits of the `OBM` algorithm are especially apparent when large ROIs are processed using a sliding-window approach (Tab. 6.1b), as opposed to the crop-and-resize method (Tab. 6.1a). This is likely because the crop-and-resize strategy already yields high-quality ROIs, resulting in fewer fragmented detections that require merging. Despite this, `OBM` consistently enhances both precision and recall in different global filtering techniques and large ROI handling strategies, demonstrating its robustness in addressing fragmented predictions. Unlike `OBS`, which contributes mainly by reducing the number of false positives, `OBM` improves the overall detection accuracy by also reducing false negatives. It achieves this by merging fragmented detections into more complete boxes, thereby increasing their overlap with ground-truth annotations and improving the likelihood of true positive matches.

In this dataset, the crop-and-resize method generally produces slightly better results than the sliding-window approach. This is likely due to the high sparsity of objects in the SeaDronesSee validation subset and the fact that small and tiny objects seldom coexist in the same detection windows as larger objects. These observations reflect overall trends related to ROI handling strategies within this particular dataset, rather than the performance of the `Global Filtering Block` itself. In particular, the difference in effectiveness between the two ROI handling methods decreases when `OBM` is applied, underscoring the strong potential of `OBM` to improve multiscale object detection.

Figure 6.7 presents a qualitative comparison of global filtering strategies, None, NMS, and `OBS`, across four representative detection scenarios. Without any global filtering (None), numerous partial false-positive detections appear in rows (a), (b), and (c). NMS is able to suppress some of

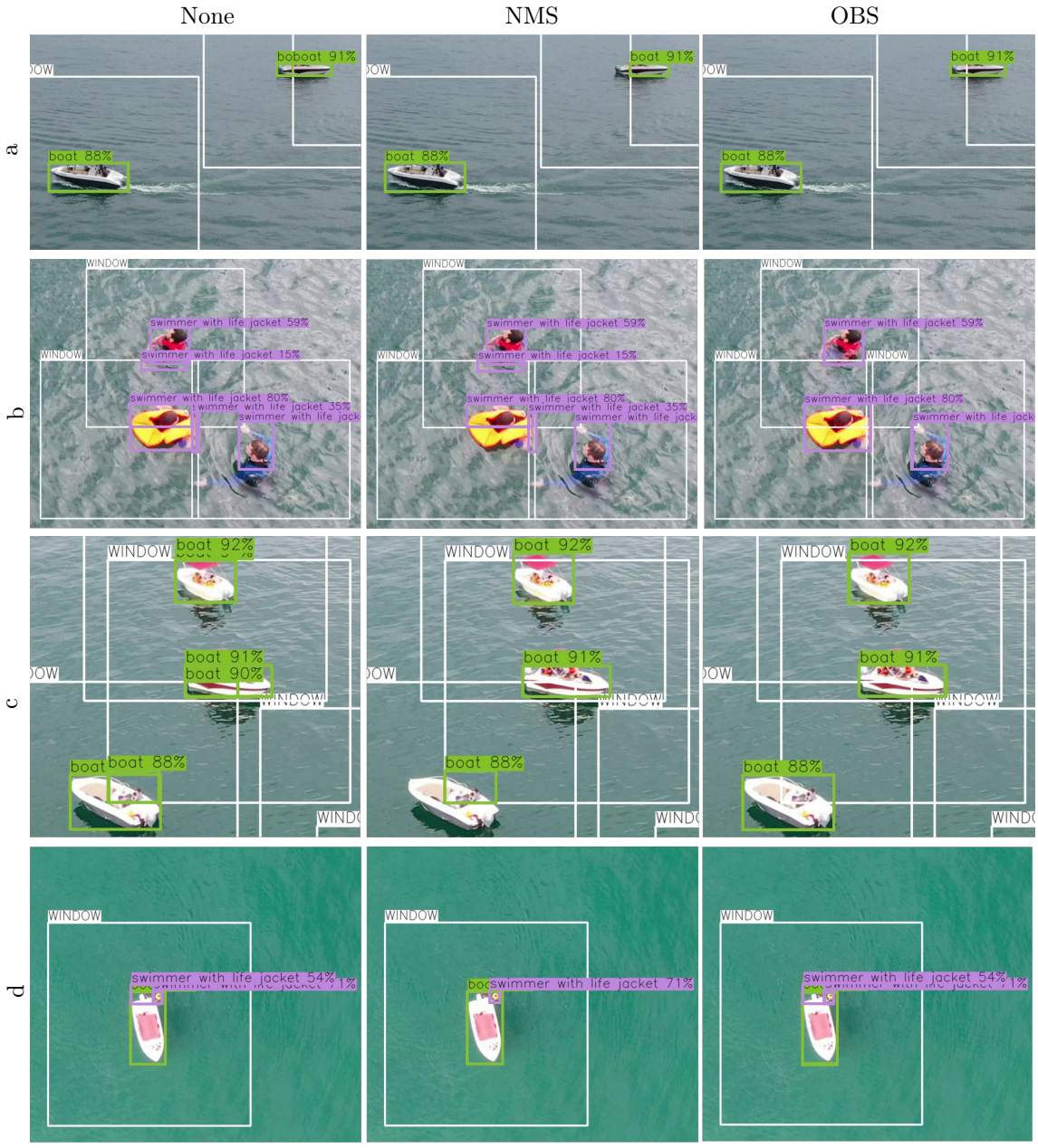


FIGURE 6.7: Qualitative comparison of no global filtering (None), NMS, and `OBS`. Without global filtering, many false positives remain. NMS retains a high-confidence partial detection, incorrectly suppresses a full detection, and fails to remove all fragmentary boxes. In contrast, `OBS` effectively eliminates all false positives without causing false negatives, preserving all correct detections. Additionally, because of the low IoU threshold, NMS suppresses a correct detection that `OBS` correctly retains by leveraging the detection window coordinates.

these (as in row c), but due to its reliance on confidence-based filtering, it often removes correct full detections while retaining partial ones with higher confidence scores (rows a-c). For instance, in row (b), three false-positive partial detections are shown: NMS fails to suppress two of them due to low IoU with full detections and retains the third because of its higher confidence, even though it is a fragment. Similar cases appear in rows (a) and (c), where fragmentary detections are retained instead of the full detections. The relatively low IoU threshold of 0.1 used in NMS helps suppress some fragmentary boxes (as seen in row c), but also increases the risk of removing

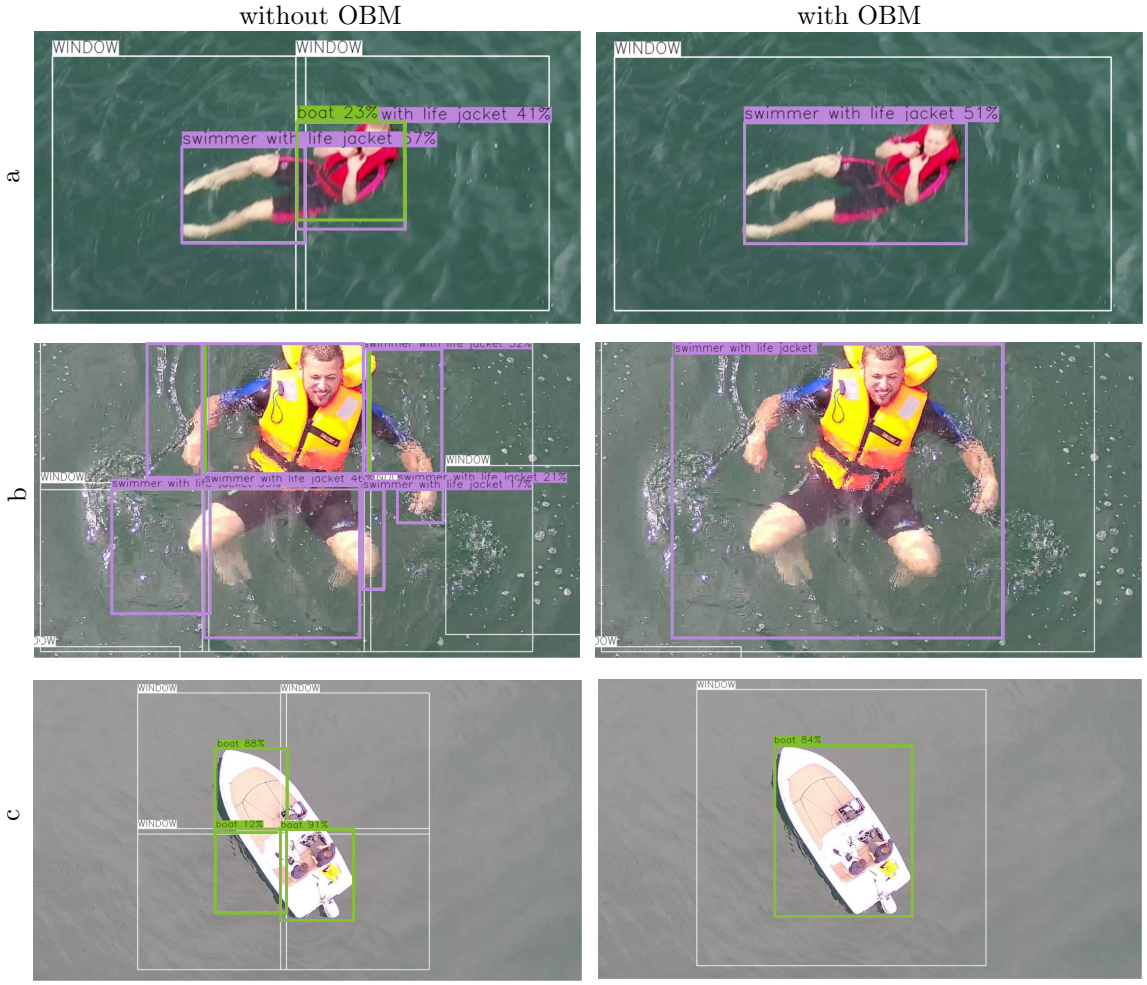


FIGURE 6.8: Qualitative comparison of the system with and without the OBM algorithm using a sliding-window strategy for large ROIs. OBM demonstrates its ability to merge fragmented detections across multiple windows (b), correct misclassified detections by assigning the correct class to the final detection (a), and partially recover missed objects (c), resulting in more complete and accurate outputs.

true detections, as illustrated in row (d). In that example, NMS incorrectly eliminates a valid detection due to its high overlap with another object. In contrast, OBS leverages the visible portion of the object when computing IoU, which allows it to retain correct detections even at very low IoU thresholds. Additionally, OBS applies a joint cost matrix based on confidence, area, and IoU, which enables better suppression of overlapping boxes. As shown consistently across all rows, OBS eliminates all partial false positives while preserving full detections, delivering a cleaner and more accurate final output. While the negative impact of the low IoU threshold used in NMS on detection quality may not be fully reflected in metrics, due to the sparse object distribution in the dataset, it would likely become more pronounced in scenarios with denser object layouts. Overall, this comparison highlights the robustness of OBS in scenarios where NMS fails.

The results of the OBM algorithm are illustrated in Fig. 6.8. In the first example (a), OBM produces a full bounding box with the correct class label, despite the presence of a false positive in the adjacent detection window, thanks to its weighted class estimation. In the second example

(b), OBM merges fragmented detections that span multiple windows, even when the object fully occupies one or more detection windows. Finally, in the third example (c), although one partial detection is missed (false negative), the OBM still successfully reconstructs the complete bounding box for the boat. These examples highlight the robustness of OBM in resolving fragmented detections, correcting misclassifications, and recovering missed objects.

Considering both qualitative and quantitative findings, it is evident that OBS outperforms NMS as a global filtering strategy in window-based detection scenarios by more effectively reducing false positives without compromising recall. The addition of OBM further refines detection outcomes by resolving fragmented, overlapping, and misclassified predictions, thus recovering missed objects and improving class consistency. Together, OBS and OBM form a robust and complementary `Global Filtering Block` that significantly boosts detection accuracy. These results strongly support the third Auxiliary Thesis, demonstrating that the proposed filtering mechanisms improves the quality of window-based tiny object detection in high-resolution images.

6.5 Ablation Study

To better understand the design choices and parameter configurations of the proposed OBS and OBM algorithms, a series of ablation experiments was conducted. While previous results focused on comparing NMS and OBS using their optimal settings, the following sections delve deeper into the influence of specific design aspects. The effect of varying IoU thresholds is analyzed, different formulations of the OBS cost vector are explored, and class-agnostic versus class-aware filtering strategies are compared. Additionally, the contribution of the OBM algorithm to overall detection performance is investigated in greater detail. To broaden the evaluation and facilitate clearer analysis of the `Global Filtering Block`, results from two additional datasets are included: DroneCrowd, which focuses on detecting tiny pedestrians in dense scenes, and ZebraFish, a simpler dataset with only a few objects per image, providing improved visual clarity for illustrating the behavior of the proposed methods.

The analysis begins with an evaluation of the performance of NMS and OBS under varying confidence and IoU thresholds, with a focus on their impact on precision and recall. This is followed by a comparison of NMS and OBS in terms of F1 score across three datasets, assessing the robustness of OBS in diverse scenarios. The design decisions presented in earlier sections are then justified through experiments evaluating both methods under different hyperparameter settings. Finally, the influence of the OBM algorithm on precision, recall, and F1 score is examined when combined with each of the three global filtering strategies: NMS, OBS, and no filtering (None).

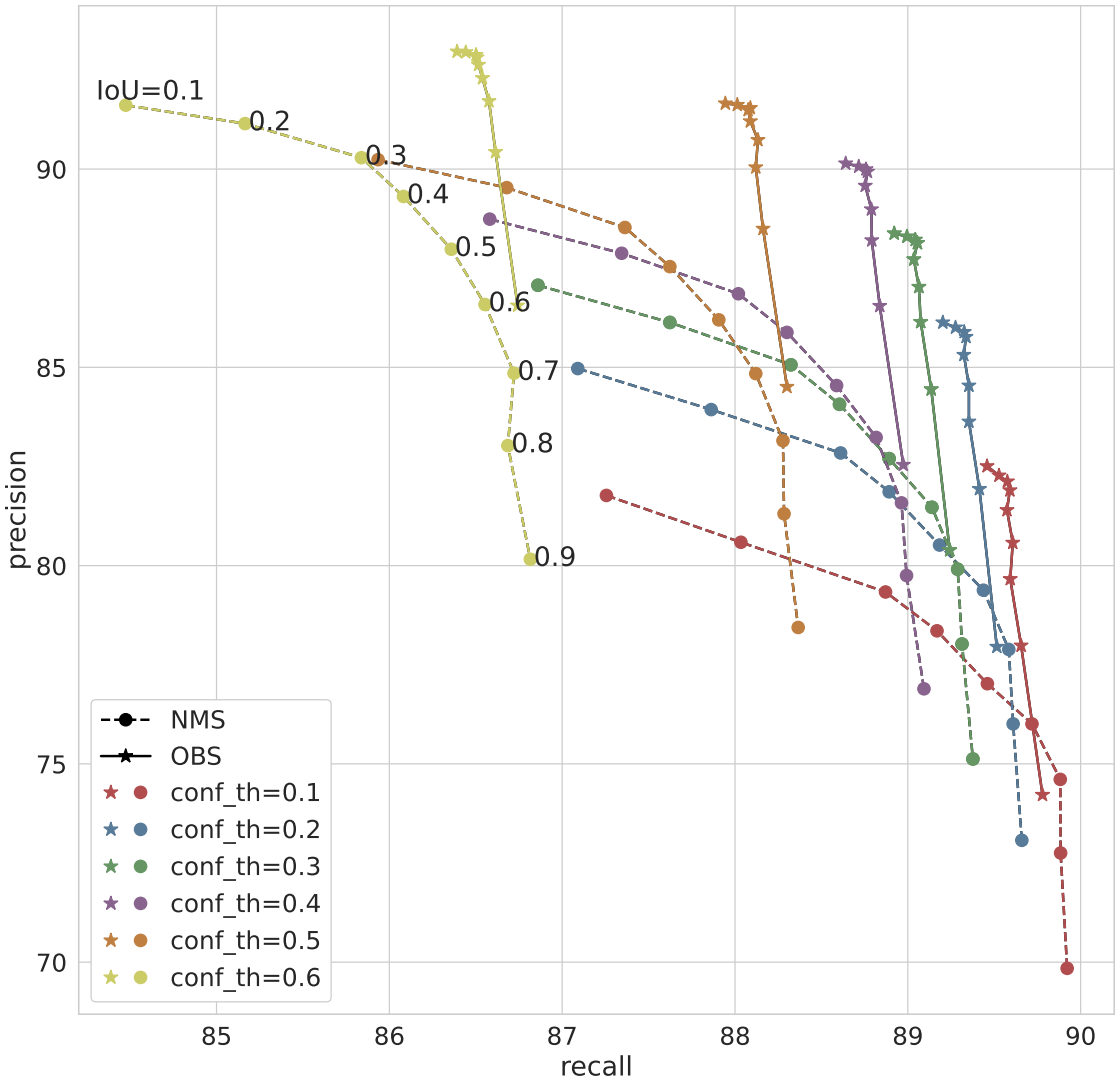


FIGURE 6.9: Comparison of NMS and the proposed `OBS` algorithm as global filtering methods in window-based object detection. Precision and recall are reported across varying confidence and IoU thresholds on the validation subset of the SeaDronesSee dataset.

6.5.1 Overlapping Box Suppression

The `Overlapping Box Suppression (OBS)` method was specifically designed to address key limitations of traditional Non-Maximum Suppression (NMS) in window-based object detection pipelines, particularly when detections span multiple overlapping sub-windows. NMS struggles in such settings because the Intersection-over-Union (IoU) between full and partial detections is often low and insufficient to trigger suppression even when both boxes refer to the same object. As a result, NMS frequently fails to remove redundant partial detections and may even suppress correct full detections due to the confidence score filtering condition.

To highlight the benefits of `OBS`, an ablation study using the SeaDronesSee dataset was conducted. The goal was to evaluate how different IoU and confidence threshold settings influence precision and recall for both NMS and `OBS` (Fig. 6.9). In the case of NMS, lowering the IoU

threshold leads to a drop in recall, as the method tends to suppress nearby true positive detections that slightly overlap. `OBS`, on the other hand, leverages the spatial layout of sub-windows and a more complex cost matrix (including confidence scores, IoU values, and bounding box areas) to distinguish between valid detections and redundant partials. As shown in Fig. 6.9, `OBS` consistently outperforms NMS in preserving recall while achieving superior precision. These results confirm that by taking the window context into account, `OBS` offers a more reliable filtering mechanism for high-resolution, window-based detection systems. This makes `OBS` especially well-suited for aerial imagery and small object detection tasks, where maintaining high precision without sacrificing recall is critical. `OBS` consistently delivers superior precision and recall compared to NMS, regardless of the chosen IoU and confidence thresholds. The performance gap, especially in recall, becomes more evident at lower confidence thresholds, demonstrating that `OBS` is more effective at retaining true positives while eliminating false positives. The strong performance of both filtering methods at low IoU thresholds is likely due to the sparse distribution of objects in the SeaDronesSee dataset, which minimizes the likelihood of accidentally suppressing valid overlapping detections.

On the densely packed DroneCrowd dataset, the optimal F1 score is reached with an IoU threshold of 0.4 for NMS and 0.5 for `OBS` (Fig. 6.10), further supporting the idea that lower IoU thresholds are preferable when objects are spread out. Figure 6.10 illustrates the variations in the F1 score for NMS and `OBS` under different IoU settings in three datasets (ZebraFish, SeaDronesSee and DroneCrowd) using a fixed confidence threshold of 0.1 and a true positive IoU threshold of 0.5. `OBS` is particularly effective in multi-scale, window-based detection pipelines. While such methods improve small object detection by enabling full-resolution analysis, they often fragment larger objects across multiple detection windows. `OBS` mitigates this issue by effectively suppressing partial and redundant boxes, significantly boosting performance on datasets like ZebraFish and SeaDronesSee. In contrast, on DroneCrowd, where partial detections are rare even with overlapping windows, NMS performs competitively or slightly better in reducing duplicate detections, though it tends to introduce more false negatives than `OBS`. Despite being designed to suppress partial detections, `OBS` also performs reliably when filtering full, redundant predictions, demonstrating its robustness across various detection scenarios.

The following paragraphs present a detailed analysis of the design decisions and parameter configurations used in both NMS and `OBS`. Since partial detections are frequently misclassified, filtering is recommended in a class-agnostic manner when addressing such cases. Figures 6.11 and 6.12 illustrate the comparison between class-based and class-agnostic filtering strategies in `OBS` and NMS across different IoU thresholds, and, in the case of `OBS`, under various filter cost vector settings. Across all tested configurations, class-agnostic filtering consistently outperforms the class-based approach. This strategy effectively reduces the number of false positives without significantly increasing the false negative rate in either method. The improvement in precision is particularly pronounced at lower IoU thresholds, likely because higher thresholds limit the pool of candidates available for filtering.

Traditional NMS often fails to preserve the most complete detections because it selects only the detection with the highest confidence, disregarding other relevant factors. To overcome this limitation, the `OBS` method introduces a multi-criteria filtering strategy that incorporates not only

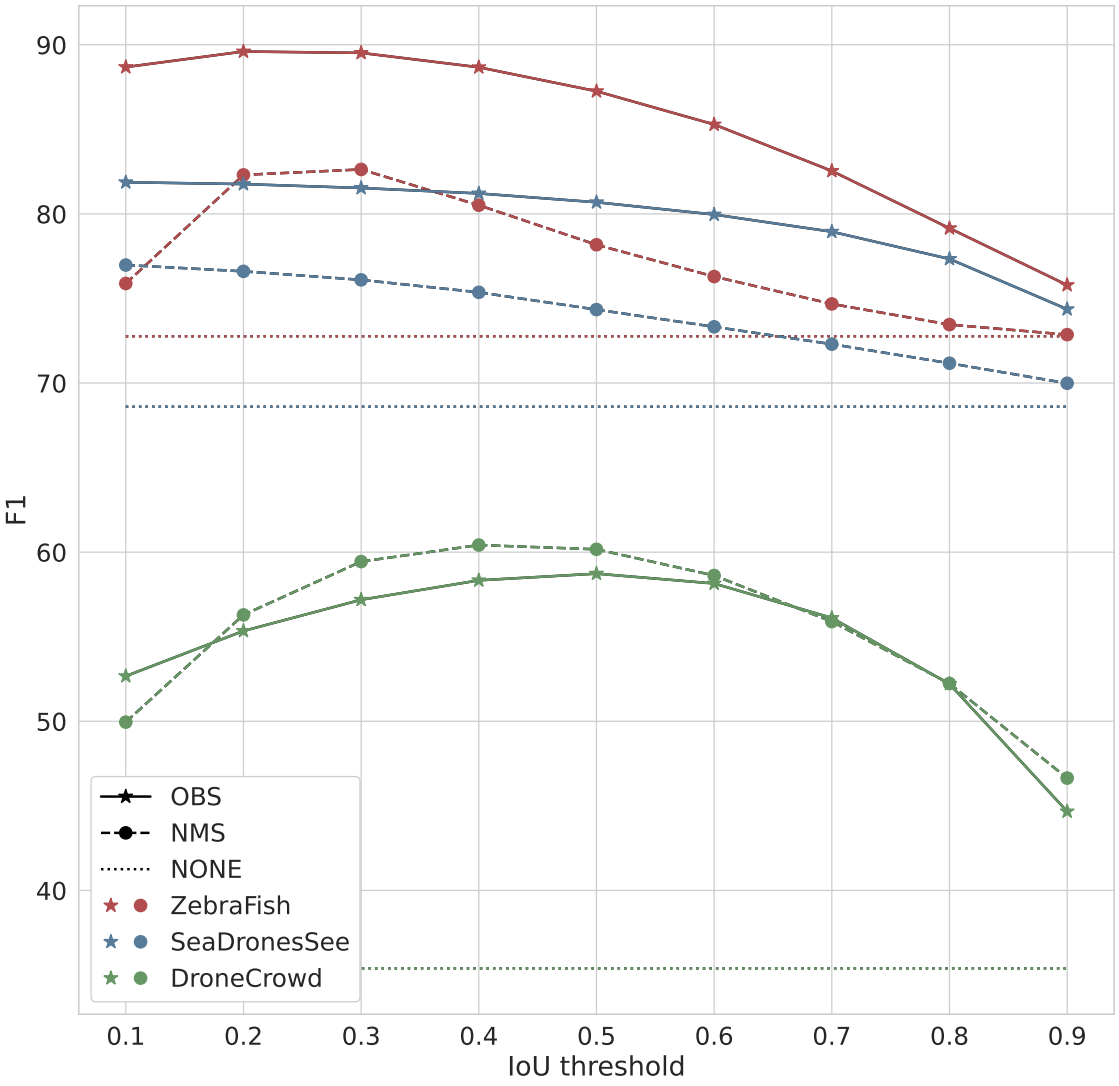


FIGURE 6.10: F1 score comparison of NMS and the proposed `OBS` algorithm across varying filtering IoU thresholds, relative to no filtering (NONE), on the ZebraFish, SeaDronesSee, and DroneCrowd datasets. Evaluations were conducted using a confidence threshold of 0.1 and a true-positive IoU threshold of 0.5.

confidence scores but also Intersection-over-Union (IoU) and bounding box area. This enables the removal of the smallest, least confident and most overlapping detections, those most likely to be partial. As shown in Fig. 6.11, this combined approach delivers better `OBS` results compared to confidence-based filtering alone. Both Fig. 6.11 (for `OBS`) and Fig. 6.12 (for NMS) show that relying solely on confidence scores results in a higher false positive rate, particularly at lower IoU thresholds. This is because multiple partial detections can have similar confidence values, making it difficult to distinguish between complete and fragmentary predictions. The proposed method, which jointly considers confidence, IoU, and area, proves to be more effective across a range of IoU thresholds and filtering modes (class-based and class-agnostic), as illustrated in Fig. 6.11. By normalizing these three metrics during filtering, it is possible to improve precision with minimal compromise on recall.

`OBS` addresses the challenge posed by low IoU between partial and full detections by leveraging detection window coordinates to estimate the visible portion of each object from the perspective

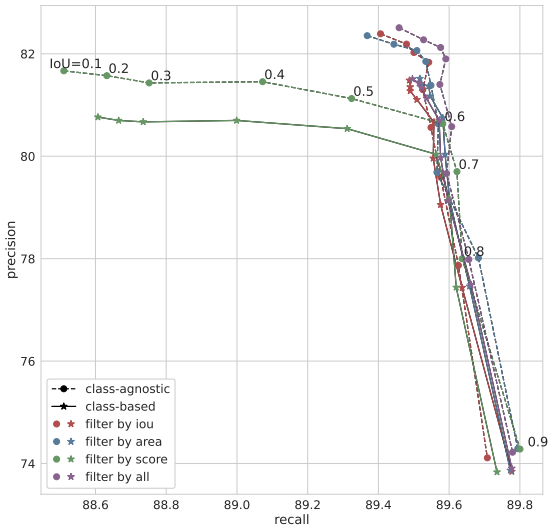


FIGURE 6.11: Evaluation of class-based versus class-agnostic filtering approaches in OBS across different IoU thresholds and filtering criteria. The results demonstrate that class-agnostic filtering consistently improves precision and recall.

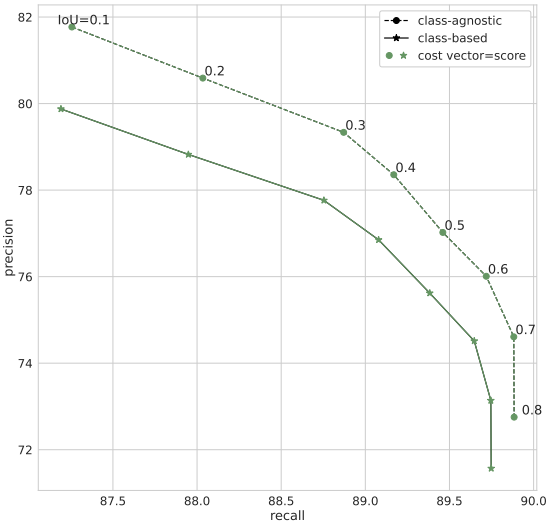


FIGURE 6.12: Assessment of class-based and class-agnostic filtering in NMS at various IoU thresholds, highlighting how class-agnostic filtering better manages the trade-off between precision and recall compared to class-based methods.

of individual windows. This approach allows OBS to more accurately identify and filter redundant partial detections. As illustrated by the discussed figures, OBS consistently reduces false positives more effectively than NMS, while having a considerably smaller negative impact on recall, regardless of the specific hyperparameter configuration. The difference in performance between the two methods is especially pronounced at lower IoU thresholds, where NMS often suppresses nearby true positives due to overlapping bounding boxes. In contrast, OBS preserves these detections by applying a more informed filtering criterion. Across a wide range of confidence and IoU thresholds, OBS achieves higher F1 scores, confirming its ability to provide more reliable and accurate filtering in window-based detection scenarios.

6.5.2 Overlapping Box Merging

Overlapping Box Merging (OBM) algorithm was introduced to effectively handle cases where multiple partial detections originate from overlapping sub-windows, particularly in situations where no full detection is present, thus rendering the Overlapping Box Suppression (OBS) algorithm inapplicable. OBM is designed to be highly adaptable, allowing seamless integration into any window-based detection pipeline, including naive sliding-window approaches and more advanced ROI-based methods. To demonstrate the broad applicability and impact of OBM, its effect on detection quality was evaluated across a wide range of global filtering configurations. Detection performance was assessed in terms of precision, recall, and F1 score, using a true positive IoU threshold of 0.5 and a confidence threshold of 0.1. Large ROIs were processed via a sliding-window strategy. Experiments were conducted for each global filtering method, including OBS with all previously tested filtering criteria (IoU, area, confidence, and their combinations), using multiple IoU thresholds from 0.1 to 0.9 in increments of 0.1 for both NMS and OBS, and for both class-agnostic and class-based filtering schemes. In total, 182 experiments were performed:

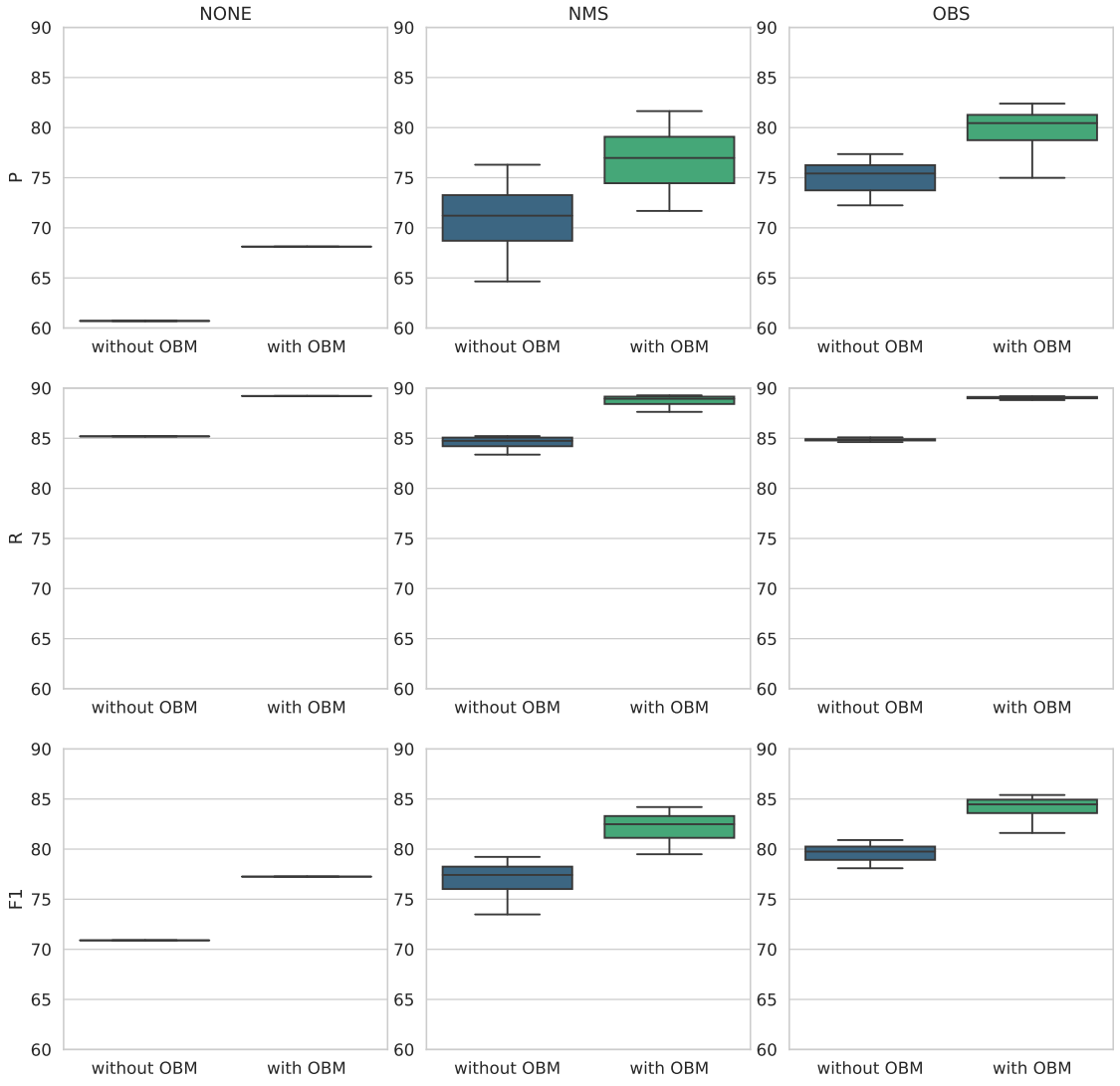


FIGURE 6.13: Effect of the `OBM` algorithm on detection performance, measured by precision, recall, and F1 score across three global filtering strategies: NONE, NMS, and `OBS`. Large ROIs were processed using a sliding-window method with a true positive IoU threshold of 0.5 and a confidence threshold of 0.1. The evaluation includes multiple IoU thresholds (0.1:0.9:0.1), both class-agnostic and class-based filtering, and all `OBS` filtering variants.

2 without global filtering, 36 with NMS, and 144 with `OBS`. The disparity in experiment counts reflects the greater number of hyperparameters in `OBS`.

As shown in Fig. 6.13, applying `OBM` (marked as True) consistently improved both precision and recall across all filtering methods. The greatest gains in precision were observed when no global filtering was applied, while the recall improvements were stable across the NONE, NMS, and `OBS` approaches. These results highlight the strong capability of `OBM` to simultaneously reduce false positives and recover missed detections by effectively merging fragmented partial boxes. Crucially, `OBM` enhances detection quality in a robust manner that is largely insensitive to the choice of filtering hyperparameters. In summary, `OBM` complements existing filtering strategies by addressing their common blind spots, fragmented detections spread across windows, and thus plays a vital role in improving the overall robustness and accuracy of window-based object detection systems.

6.6 Conclusions

This thesis addresses a fundamental challenge in window-based object detection systems: improving detection quality in the presence of intersecting windows and partial detections. Such systems are frequently employed to enhance the visibility of small and tiny objects by increasing their relative size within the input image. However, the overlapping nature of detection windows introduces a range of issues, including redundant partial detections and false positives that may not be adequately filtered by standard techniques such as Non-Maximum Suppression (NMS). Notably, NMS often fails to suppress partial false positives when their overlap with a full detection is low and tends to discard complete detections based solely on confidence scores, ignoring other informative cues such as object size. In many practical scenarios, no complete detection is available at all, only multiple fragmented ones, leading to both false positives and false negatives. These limitations motivate the global filtering strategy proposed in this chapter, which builds upon earlier research [64, 65, 69] and introduces two key contributions: the `Overlapping Box Suppression (OBS)` algorithm for suppressing redundant partial detections, and the `Overlapping Box Merging (OBM)` algorithm for consolidating multiple fragments into a single object when no full detection exists.

A comprehensive experimental comparison of `OBS` and NMS against a baseline with no global filtering demonstrates that `OBS` consistently outperforms NMS in reducing false positives while better preserving true positives. This advantage is reflected in higher precision and a less pronounced negative effect on recall, even at lower IoU thresholds where NMS typically struggles. Qualitative analysis further reveals `OBS`'s ability to prioritize the most complete detections while discarding small and ambiguous fragments that often bypass confidence-based filtering.

In-depth ablation studies highlight the performance of `OBS` in a variety of configurations, including different IoU thresholds, filtering criteria (confidence, area, and IoU), and class-based versus class-agnostic filtering. The results confirm that a combined filtering scheme that leverages all three parameters, normalized and integrated into a cost matrix, yields the most robust performance. This method notably improves recall under low IoU conditions, where confidence-based filtering alone proves insufficient. Furthermore, class-agnostic filtering consistently outperforms class-specific variants in both `OBS` and NMS, likely due to the high rate of misclassifications observed in partially visible objects.

The `OBM` algorithm further extends the robustness of window-based systems by merging fragmented detections across intersecting windows. Its application significantly boosts detection quality by recovering objects that would otherwise be missed or counted as false positives. `OBM` was shown to improve precision, recall, and F1 score in all three global filtering baselines (None, NMS, and `OBS`) especially when applied to systems using a sliding-window strategy to process large ROIs. Although it also contributes positively in crop-and-resize scenarios, the gains are more modest due to the lower frequency of partial detections. The extensive experimental analysis demonstrates `OBM`'s consistent improvements regardless of the underlying filtering configuration, underscoring its effectiveness and versatility.

When `OBS` and `OBM` are applied together, the overall detection quality improves significantly. In the SeaDronesSee validation dataset, their combined use increases the F1 score from 75.4% to 86.1% when using a crop-and-resize ROI strategy, and from 70.9% to 85.0% with a sliding-window approach, surpassing NMS by 1.7 and 6.2 percentage points, respectively. These results provide strong empirical support for Auxiliary Thesis #3, which states that the common problem of false positive partial detections in window-based systems can be mitigated through post-processing techniques such as `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`. The improvements achieved across both ROI strategies clearly demonstrate the advantages of the proposed methods over conventional NMS in handling overlapping regions and fragmented detections.

In the future, more work is planned to extend the applicability of `OBS` and `OBM` to additional detection frameworks, including naive sliding-window systems and various domain-specific datasets. Exploration of class-aware merging strategies for `OBM` may also yield additional gains in scenarios with highly imbalanced or visually similar object classes. Furthermore, the development of a learned suppression and merging mechanism, using `OBS` and `OBM` output as a foundation, presents a promising direction for further enhancing detection quality. These future efforts aim to extend the capabilities of window-based systems by addressing their most persistent weaknesses: partial detections resulting from intersecting detection windows.

Together, `OBS` and `OBM` offer a practical and extensible framework for improving precision and recall in object detection tasks characterized by overlapping regions and multi-scale objects. Their introduction lays a strong foundation for continued innovation in this space, where accurate detection remains a persistent challenge.

Chapter 7

Implementation Details

7.1 Introduction

Sliding-window methods [96, 151, 178] enhance tiny object detection by increasing the relative object scale but come with high computational cost. Region-focused approaches [34, 67, 75, 155, 157], on the other hand, improve efficiency by localizing and processing only selected Regions of Interest (ROIs), thereby achieving a better trade-off between accuracy and speed. While these methods reduce computational load and retain high-resolution processing, there is a noticeable lack of open-source frameworks that simultaneously prioritize detection accuracy, computational efficiency, and inference speed. Existing approaches often lack official implementations, rely on fixed architectures with limited configurability, or are not optimized for real-time applications such as robotics, navigation, or surveillance.

The proposed SegTrackDetect framework fills this gap by offering a highly modular and customizable pipeline. The system combines segmentation-based `ROI Estimation` and tracking-based `ROI Prediction` to identify regions for full-resolution inference. To handle partial detections from multiple sub-windows, it introduces the `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)` algorithms, which help eliminate redundancies and improve results. By combining `OBS` and `OBM` with large ROI handling strategies, the system efficiently handles both tiny-only and multi-scale detection scenarios in high-resolution images, making it suitable for deployment in resource-constrained environments. Designed with real-world applications in mind, it supports both image and video input while balancing detection accuracy with computational efficiency.

Most of the experiments presented in this thesis were conducted using the SegTrackDetect framework, with precise architectural components and configurations described in detail in their respective chapters. This chapter focuses on the system-level implementation, outlining the framework's capabilities, supported tasks, and available customization options. The first part of the chapter describes the desktop implementation, in which all models are deployed in native TorchScript format. This version of the system has been released as open-source software [70], enabling

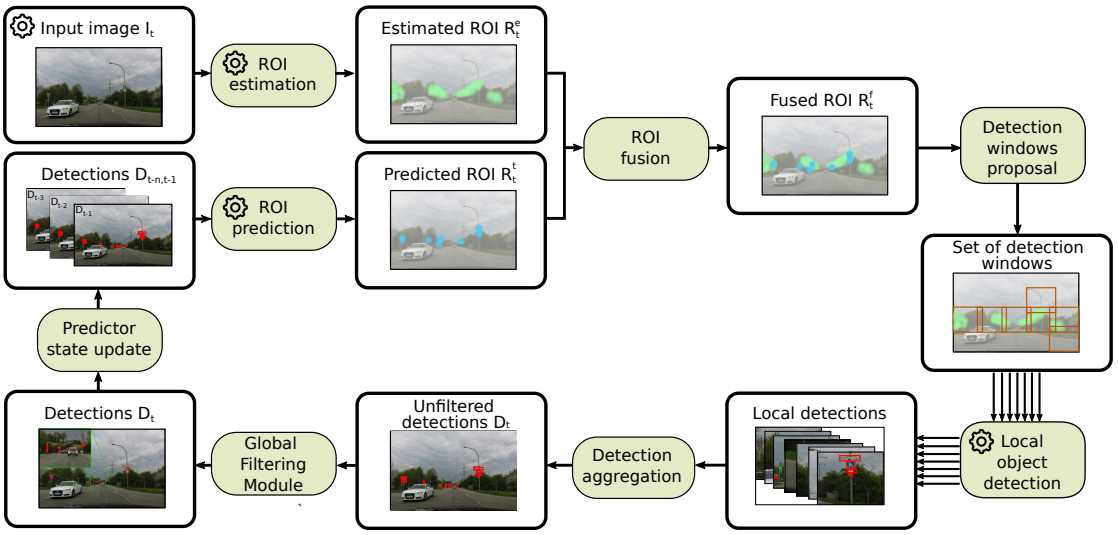


FIGURE 7.1: High-level architecture of the SegTrackDetect framework. Modules that support user-defined customization are marked with the "cog" icon. All other components can also be configured via arguments passed to the inference script (Python CLI).

broad accessibility and reproducibility. The second part discusses the embedded deployment using TensorRT, providing a discussion of performance gains and trade-offs in quality and inference time. This part reflects the applied nature of the research conducted within the framework of an industrial PhD program. Due to commercial considerations, only the desktop implementation has been made publicly available, while the embedded version remains proprietary.

7.2 SegTrackDetect

SegTrackDetect [70] is an open-source framework for efficient small and tiny object detection in high-resolution images. It performs inference only within selected regions of interest (ROIs), substantially reducing the computational load compared to exhaustive sliding-window approaches. Within these selected regions, full-resolution inference is preserved, enabling the use of lightweight detectors without compromising accuracy. The system employs two complementary ROI selection mechanisms. The `ROI Estimation Module` produces binary masks of potential object locations via semantic segmentation, while the `ROI Prediction Module` uses an object tracker to predict ROIs based on previous frame detections. This dual approach enables compatibility with both image-based and sequential detection tasks. Detections obtained from individual windows are aggregated and post-processed via `Global Filtering Block` implementing both `Overlapping Box Suppression` and `Overlapping Box Merging` to remove redundant results and produce high-quality outputs. Detailed descriptions of all system components are provided in Chapter 3, with additional extensions and validations discussed in Chapters 4, 5, and 6. The focus here shifts from architectural design and experimental validation to practical usability and application from the end-user perspective.

SegTrackDetect is designed with modularity and extensibility in mind. All core modules (`ROI Estimation`, `ROI Prediction`, and `Object Detection`) can be replaced with user-defined models, as long as they are exported to TorchScript and follow the required pre- and post-processing

structure. This flexibility allows users to adapt the system to different performance constraints and application requirements, whether prioritizing speed, accuracy, or model complexity. The framework automatically determines whether input data is image- or video-based and supports both CPU and GPU execution (with GPU recommended for faster inference). A Dockerfile and accompanying scripts are provided to streamline setup, and the Python-based interface allows configuration via command-line arguments or configuration files. Users can customize pre- and post-processing pipelines, and integrate any dataset by following the required structure outlined in the manual. The overall architecture, closely aligned with the design presented in Chapter 5, is illustrated in Fig. 7.1. Components that support user-defined customization are marked with a "cog" icon. These include the `ROI Estimation Module`, `ROI Prediction Module`, `Object Detection Module`, and the dataset interface (indicated at the input image stage), as the framework is designed to facilitate seamless integration of custom datasets and models. The remaining components also offer a degree of configurability, which can be adjusted via command-line arguments.

SegTrackDetect supports four datasets for benchmarking:

- Mapillary Traffic Sign Dataset (MTSD) [35] for image-based traffic sign detection,
- SeaDronesSee [132] for drone-based person and boat detection in video sequences,
- DroneCrowd [147] for video aerial crowd detection,
- ZebraFish [100] for sequential fish detection in laboratory tanks.

Example configurations include pre-trained models such as YOLOv4 [12], YOLOv7 [136], U-Net [114], and U2-Net [104]. The SORT-based tracker [9] is used in the default implementation of the `ROI Prediction Module`, but users are encouraged to incorporate their own architectures into any part of the pipeline. SegTrackDetect facilitates rapid experimentation by allowing researchers and developers to focus on architectural design rather than framework engineering. This is, to the best of current knowledge, the first window-based detection framework offering this degree of modularity and configurability for small and tiny object detection. The system has been used in previous work focused on UAV-based detection pipelines [69] and was widely used throughout the experiments discussed earlier in this work.

The main contributions of SegTrackDetect are as follows:

- a complete and modular system for real-time small and tiny object detection in high-resolution imagery,
- configurable modules for `ROI Estimation`, `ROI Prediction`, and `Object Detection`, supporting both pre-trained and custom models,
- integrated support for four publicly available datasets: SeaDronesSee, DroneCrowd, ZebraFish, and Mapillary Traffic Sign Dataset (MTSD).

The code is publicly available at `https://github.com/deepdrivepl/SegTrackDetect`.

7.2.1 Software Functionalities and Default Settings

The SegTrackDetect framework is organized around a modular pipeline designed to support efficient detection of small and tiny objects in high-resolution images. Its architecture includes three core components: the `ROI Estimation Module`, the `ROI Prediction Module`, and the `Detection Module`. These are supported by auxiliary components such as the `ROI Fusion Module`, `Detection Windows Proposal`, and the `Global Filtering Block`. The `ROI Estimation Module` extracts coarse binary masks from input images, reducing the search space for the detector. When operating in video mode, the `ROI Prediction Module` complements this process by predicting object locations based on temporal continuity using a tracker. The outputs from both modules are combined by the `ROI Fusion Module` to then form a set of detection subwindows. Optionally, the framework supports a sliding-window mode in which the fused mask is replaced with the all-foreground output. Given the binary mask, the `Detection Windows Proposal Module` creates the batch of detection subwindows for the `Local Object Detector` to process at high resolution. Detected boxes are projected back to the original image coordinate system and filtered to eliminate duplicates. If video mode is active, filtered detections are used to update the tracker state. Users may replace the default models and logic for `ROI Estimation`, `ROI Prediction`, and `Object Detection`, as well as modify pre- and post-processing functions. These advanced customization procedures are discussed in detail in the next section. Additional configuration is supported through the Python CLI. Modules that enable model-level customization are highlighted with a "cog" icon in Fig. 7.1, while other settings can be adjusted via script arguments.

Pre-trained components and example configurations are provided for several tasks:

- traffic sign detection in image mode using the Mapillary Traffic Sign Dataset (MTSD) [35], with an integrated YOLOv4 detector and a U2-Net-based `ROI Estimator`,
- maritime object detection in drone video sequences using the SeaDronesSee dataset [132], employing a trained YOLOv7 detector and U-Net-based `Estimators` with various input resolutions,
- crowd detection in drone video footage using the DroneCrowd dataset [147], integrating YOLOv7 with multiple U-Net-based `ROI Estimators`,
- fish detection in laboratory recordings using the ZebraFish dataset [100], combining a YOLOv7 detector with a U-Net-based `ROI Estimator`.

All three video-based use cases employ the lightweight SORT-based tracker integrated into the SegTrackDetect framework. To ensure ease of use, SegTrackDetect provides a Docker-based environment that manages all dependencies. Installation requires Docker Engine and NVIDIA Container Toolkit for GPU execution. Scripts are provided to build the image, run the pipeline, and download both pre-trained models and demo datasets. The main inference script, `inference.py`, supports both the image and the video modes, with the appropriate mode automatically inferred from the structure of the dataset. Its behavior can be extensively customized via a set of command-line arguments.

TABLE 7.1: Command-line arguments supported by the inference script `inference.py`.

Argument	Type	Description
`--roi_model`	`str`	Specifies the ROI model to use. All available ROI models: `MTSD`, `ZeF20`, `SDS_tiny`, `SDS_small`, `SDS_medium`, `SDS_large`, `DC_tiny`, `DC_small`, `DC_medium`
`--det_model`	`str`	Specifies the detection model to use. All available detectors: `MTSD`, `ZeF20`, `SDS`, `DC`
`--tracker`	`str`	Specifies the tracker to use. All available trackers: `sort`.
`--data_root`	`str`	Path to the dataset directory (e.g., `/SegTrackDetect/data/MTSD`).
`--split`	`str`	Data split to use (e.g., `val`). If present, the script saves detections using COCO image IDs defined in `val.json`.
`--flist`	`str`	Alternative to `--split`; specifies a file containing absolute paths to input images.
`--name`	`str`	Name for the `flist` input. A COCO-style `name.json` file will be saved in the dataset root directory.
`--bbox_type`	`str`	Type of detection window filtering algorithm: `all` (no filtering), `naive`, or `sorted`.
`--allow_resize`	`flag`	Enables resizing of cropped detection windows. Otherwise, sliding windows are used within large ROIs.
`--obs_iou_th`	`float`	IoU threshold for Overlapping Box Suppression (default: 0.7).
`--cpu`	`flag`	Forces CPU execution. If not set, CUDA is used.
`--out_dir`	`str`	Directory to save output results (e.g., `results`).
`--debug`	`flag`	Enables saving visualizations in `out_dir`.
`--vis_conf_th`	`float`	Confidence threshold for visualized detections (default: 0.3).

Through the Python CLI, users can control key components of the framework. In image mode, frames are processed independently using only the `ROI Estimation Module`. In contrast, video mode treats inputs as a continuous sequence, enabling `ROI Prediction` through tracking. All detections are saved in the output directory as `results-<split>.json` or `results-<name>.json`, depending on the evaluation split or provided name. The complete list of command-line arguments with detailed explanations is provided in Tab. 7.1. For step-by-step usage instructions, please refer to the manual at `https://github.com/deepdrivepl/SegTrackDetect/README.md`.

Figure 7.2 illustrates a sample output from the system. The orange regions represent the binary mask produced by the `ROI Estimation Branch`, while the black rectangles mark the detection windows. The detected objects are labeled with their class names and confidence scores. Aside from the zoomed-in insets shown for clarity, the image is an unaltered debug output generated by the framework. All detections are saved in the COCO-compatible JSON format.

7.2.2 Advanced Customization Options

The SegTrackDetect framework has been designed with a strong emphasis on modularity and extensibility, facilitating advanced customization of its core components. The system architecture supports seamless substitution and modification of the `ROI Estimation`, `ROI Prediction`, and `Object Detection Modules`, making it suitable for both academic research and industrial applications. Users can integrate custom models, datasets, and processing pipelines with minimal effort, while the internal behavior of each component can be configured via dedicated configuration dictionaries.

FIGURE 7.2: Sample debug output from the SegTrackDetect framework using the Mapillary Traffic Sign Dataset.

The following paragraphs outlines the available customization options, including parameter tuning for integrated models and the integration of new architectures. A more detailed step-by-step explanation is available in the project manual.

ROI Estimation Models For integrated ROI estimation models, users can adjust the behavior of the module by modifying its configuration file. This includes changes to pre- and post-processing routines, as well as to specific hyperparameters such as thresholds or morphological dilation settings. New models may be incorporated by adhering to the required dictionary format (Listing 7.1).

```
CustomModel = dict(
    weights = 'weights/my_custom_model_weights.pt',
    in_size = (h,w),
    postprocess = postprocess_function,
    postprocess_args = dict(),
    preprocess = preprocess_function,
    preprocess_args = dict()
)
```

LISTING 7.1: Example configuration dictionary for a custom `ROI Estimator`.

Custom preprocessing and postprocessing functions must comply with the expected input and output formats, which are documented in detail in the accompanying user manual at `https://github.com/deepdrivepl/SegTrackDetect`.

ROI Prediction The default implementation of the `ROI Prediction Module` relies on a SORT-based object tracker, which can be customized by editing its configuration. Adjustable parameters include `max_age`, `min_hits`, `iou_threshold`, `min_confidence`, and `frame_delay`. To integrate a custom tracking algorithm, users need to implement a tracker class with separate prediction and update stages. Once implemented, the tracker can be incorporated into the system via the corresponding configuration entry (Listing 7.2).

```
CustomTracker = dict(
    module_name = 'rois.predictor.CustomTracker',
    class_name = 'CustomTracker',
    args = dict(),
    frame_delay = 3,
)
PREDICTOR_MODELS = {
    'sort': sort,
    'custom': CustomTracker,
}
```

Listing 7.2: Configuration dictionary for a custom `ROI Predictor`.

Object Detection The `Object Detection Module` can also be fully customized. Similarly to the `ROI Estimator`, new models must be registered via a configuration dictionary that specifies the model weights, input resolution, preprocessing and postprocessing functions, and class definitions (Listing 7.3).

```
CustomModel = dict(
    weights = '/SegTrackDetect/weights/my_custom_model_weights.pt',
    in_size = (h,w),
    preprocess = preprocess_function,
    preprocess_args = dict(),
    postprocess = postprocess_function,
    postprocess_args = dict(),
    classes = ['class_a', 'class_b', 'class_c', ...],
    colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), ...]
)
```

Listing 7.3: Configuration dictionary for a custom `Object Detector`.

This structure allows integration of detection networks with arbitrary input requirements and classes, provided that the appropriate IO interface is respected.

New Datasets SegTrackDetect supports datasets in both image-based and video-based formats. To integrate a new dataset, users must follow the required directory structure and annotation format, as shown in Listing 7.4.

```
SegTrackDetect/data/YourVideoDataset/
|-- images/
|   |-- seq1/              # Sequence 1 with A~images
|   |-- seq2/              # Sequence 2 with B images
```

```
|   |-- seq3/              # Sequence 3 with C images
|   |-- seq4/              # Sequence 4 with D images
|   +-- ...                # Additional sequences as needed
|-- split_x.json           # Annotations in COCO format
|-- split_y.json           # Annotations in COCO format
+-- split_z.json           # Annotations in COCO format

SegTrackDetect/data/YourImageDataset/
|-- images/                # All images should be placed directly in this directory
|   |-- image1.jpg         # Image file 1
|   |-- image2.jpg         # Image file 2
|   |-- image3.png         # Image file 3
|   +-- ...                # Additional image files as needed
|-- split_x.json           # Annotations in COCO format
|-- split_y.json           # Annotations in COCO format
+-- split_z.json           # Annotations in COCO format
```

Listing 7.4: Required directory structure for new datasets.

Annotations should follow the COCO format, and the `file_name` field in each annotation entry must contain an absolute path to the corresponding image. Once the structure and annotation files are correctly configured, the dataset can be used seamlessly within the framework for inference or evaluation.

7.2.3 Trade-off Analysis of SegTrackDetect Configurations

This section presents example results that illustrate the trade-offs associated with different configurations of the SegTrackDetect framework. In Table 7.2, three operating modes are considered: (1) video mode, which combines both `ROI Estimation` and `ROI Prediction`; (2) image mode, which uses only `ROI Estimation`; and (3) a sliding-window mode, which disables both ROI modules and simulates a full uniform tiling across the image. To contextualize performance, the SAHI sliding-window object detection framework [2] is included as a baseline for comparison.

Evaluation is performed using the protocol from [67], tailored to high-resolution tiny object detection with relative size thresholds. The protocol extends standard COCO metrics with additional object size categories, enabling finer-grained performance analysis across scales. The results report Average Precision (AP) for micro to large object categories and inference speed in frames per second (FPS). The same detection models (YOLOv7 Tiny, YOLOv7, and YOLOv7 e6e) are used in all experiments. SegTrackDetect is configured with a UNet-based `ROI Estimator` (with ResNet18 backbone and input size of 448×768) and a SORT-based `ROI Predictor`, while SAHI is tested with identical subwindow sizes, overlap ratios, and post-processing parameters as the sliding-window mode in SegTrackDetect. Across all detection models, SegTrackDetect in video mode delivers the highest detection accuracy by leveraging both `Estimation` and `Prediction`. The addition of the tracking-based `ROI Predictor` improves recall, especially for small and discontinuously visible objects (as thoroughly discussed in Chaper 5), at a modest computational cost approximately 7–11% reduction in speed depending on model size. Despite this, real-time performance is preserved, with frame rates between 18.4 and 33.5 FPS for video mode, and between 20.3 and 36.3 FPS in image mode. In image mode, where only `ROI Estimation` is used, a small drop in detection quality is observed along with a moderate improvement in processing

TABLE 7.2: Quantitative results comparing SegTrackDetect (STD) in three modes (video, image, and sliding-window) with SAHI [2] using the SeaDronesSee validation dataset. All experiments run on NVIDIA RTX 4090 and use the evaluation protocol from [67].

Framework	**Estimator**	**Predictor**	**Detector**	**AP**	**AP$_{50}$**	**AP$_{vt}$**	**AP$_t$**	**AP$_s$**	**AP$_m$**	**AP$_l$**	**FPS**
STD	UNet	Sort	yolov7tiny	**53.2**	**84.7**	**33.8**	**52.4**	**61.9**	**37.6**	**69.0**	33.5
STD	UNet	—	yolov7tiny	52.9	83.9	32.2	52.2	**61.9**	37.3	68.1	**36.3**
STD	—	—	yolov7tiny	47.1	79.3	33.2	46.3	52.8	29.2	30.6	18.1
SAHI [2]	—	—	yolov7tiny	45.0	80.4	33.4	49.3	54.5	19.7	16.7	1.2
STD	UNet	Sort	yolov7	**53.6**	**86.0**	**35.8**	**51.5**	61.7	**41.6**	**68.4**	26.4
STD	UNet	—	yolov7	53.5	85.2	34.5	**51.5**	**61.8**	**41.6**	67.4	**29.7**
STD	—	—	yolov7	44.7	78.7	33.0	44.2	50.7	27.3	37.0	7.9
SAHI [2]	—	—	yolov7	39.2	76.4	32.1	42.6	48.9	21.8	17.9	1.1
STD	UNet	Sort	yolov7e6e	**52.3**	**86.3**	**34.3**	**51.8**	61.3	**45.6**	**69.4**	18.4
STD	UNet	—	yolov7e6e	52.1	85.6	32.3	**51.8**	**61.4**	**45.6**	67.7	**20.3**
STD	—	—	yolov7e6e	45.3	80.2	32.6	46.4	51.1	34.8	48.5	4.9
SAHI [2]	—	—	yolov7e6e	40.3	77.7	31.8	46.8	51.5	25.6	17.2	0.8

speed. This configuration may be preferable in scenarios where inference speed is the primary constraint. Alternatively, the video mode, leveraging both `ROI Estimation` and `Prediction`, can be used with a reduced input resolution for the `Estimation Module` to achieve a more favorable balance between detection quality, computational efficiency, and runtime performance. Finally, the SegTrackDetect implementation of the sliding-window approach demonstrates that it can match or surpass the accuracy of SAHI while offering significantly faster inference due to its more efficient window management and internal optimizations. The greatest performance advantage is observed for medium and large objects, which benefit from the framework's large-ROI handling strategy and the `Global Filtering Block`, two key contributions discussed earlier in this thesis that effectively consolidate redundant detections and improve overall detection quality.

Increasing detector size (e.g., moving from YOLOv7 tiny to e6e) leads to longer inference times but does not consistently improve detection quality. This effect is likely due to the fixed input resolution across models and the dataset-specific class imbalance. The greatest improvements are observed in the underrepresented medium-size category, where the capacity of the model may contribute more significantly to generalization. These results underscore SegTrackDetect's flexibility: By selecting appropriate operating modes and detectors, users can tailor the system to meet specific speed and accuracy constraints in a variety of deployment contexts.

7.2.4 Conclusions

SegTrackDetect introduces a flexible, modular, and efficient solution for small and tiny object detection in high-resolution imagery that do not compromise the quality for larger objects. The framework integrates `ROI Estimation`, `ROI Prediction`, and `Detection Modules` into a unified pipeline that enables real-time performance without sacrificing detection accuracy. By focusing inference only on relevant sub-regions of the image, SegTrackDetect reduces computational cost while maintaining high precision, particularly for small-scale objects. Unlike prior

window-based approaches, which often rely on rigid architectures or lack available implementations [34, 67, 75, 155, 157], SegTrackDetect emphasizes adaptability and reproducibility. Its architecture supports easy replacement or modification of individual components, such as using a different estimators, trackers, or detectors, while maintaining providing standardized interfaces. This is further supported by Docker-based deployment, which streamlines setup and ensures reproducibility across platforms. A key strength of the system lies in its `Global Filtering Block`, which employs the `OBS` and `OBM` algorithms to eliminate redundant detections across overlapping windows. This mechanism directly addresses one of the main challenges in window-based detection and plays a central role in maintaining high precision in the final outputs.

Beyond its core capabilities, SegTrackDetect serves as an extensible research and prototyping platform. Its support for COCO-format outputs, detailed documentation, and customizable modules lowers the entry barrier for new users while enabling advanced experimentation by experienced researchers. The system has already supported the research [69], demonstrating its practical utility in developing new detection algorithms.

The experimental results confirm the efficiency and robustness of the framework in multiple configurations. SegTrackDetect achieves superior accuracy compared to SAHI [2] while running significantly faster, even when operating in sliding-window mode. When used with the video-mode, it achieves over 25 FPS with YOLOv7 and over 30 FPS with its lightweight variant, making it suitable for real-time applications. In summary, SegTrackDetect is a comprehensive publicly available solution for customizable ROI-based detection pipelines. It provides both a practical tool for real-world deployment and a research platform for advancing the field of small object detection.

7.3 Embedded Device Implementation

Optimization for embedded devices is a critical practical consideration for robotic computer vision systems. A brief analysis of the quality-speed trade-offs is presented here. Since this aspect relates primarily to implementation rather than the core scientific contributions of the dissertation, the discussion is kept concise. Furthermore, the evaluation is performed on the initial TinyROI system rather than the full SegTrackDetect framework, as the latter has been optimized for embedded platforms in a proprietary version maintained for commercial use.

The TinyROI system was deployed and evaluated on an embedded device with limited memory and computational resources. The experiments were conducted on a Jetson AGX Orin 64GB platform. To reduce memory requirements, the neural network models used in the system were optimized for the device's hardware architecture. The impact of reduced model weight precision on inference time and detection quality was also examined. For this purpose, TensorRT was employed, and three levels of weight precision were evaluated: FP32, FP16, and INT8. Both trained components of the system were subject to optimization, namely the UNet [114] semantic segmentation model used by the `ROI Estimator` and the YOLOv7 Tiny [136] employed for `Local Object Detection`. During performance measurements (FPS), the inference times of both models were included. Reported values were determined as the average inference time per

TABLE 7.3: Object detection metrics on the SeaDronesSee validation set for different formats and weight precisions. TS – TorchScript, TRT – TensorRT. Measurements conducted on the Jetson AGX Orin 64GB.

format	precision	FPS	AP	AP_{50}	AP_{75}	AP_{vt}	AP_t	AP_s	AP_m	AP_l	AR	AR_{vt}	AR_t	AR_s	AR_m	AR_l
TS	FP32	12.2	**53.1**	**85.0**	**57.7**	**34.3**	52.1	**61.7**	37.4	70.8	**62.1**	40.8	62.6	**69.9**	43.7	75.6
TRT	FP32	28.2	**53.1**	**85.0**	**57.7**	**34.3**	**52.2**	**61.7**	**37.5**	**71.2**	**62.1**	**40.9**	**62.7**	**69.9**	**43.8**	**76.1**
TRT	FP16	35.1	**53.1**	**85.0**	**57.7**	**34.4**	52.1	**61.7**	**37.5**	71.0	**62.1**	**40.9**	62.6	**69.9**	**43.8**	75.3
TRT	INT8	**39.4**	28.6	56.1	27.1	25.9	31.1	25.7	18.8	49.5	45.1	34.7	46.0	46.3	30.4	50.9

image across the entire SeaDronesSee [132] validation set. The experimental results are presented in Tab. 7.3. As a reference for both detection quality and inference speed, TorchScript-based models with FP32 precision weights were used, as prepared and described in [69]. The batch size for both the segmentation model and the detector was set to 1.

Optimization of the trained models with TensorRT enabled more than a twofold increase in inference speed – from 12.2 FPS to 28.2 FPS. Furthermore, reducing weight precision from FP32 to FP16 provided an additional speedup to 35.1 FPS without any loss in detection quality. INT8 weights offered even greater acceleration, though at the cost of a noticeable degradation in quality. These results highlight the practical trade-off between speed and accuracy, and confirm that reducing the model weights and optimizing the model for a specific hardware allows efficient deployment on resource-constrained systems.

Chapter 8

Conclusions

8.1 Summary

The primary goal of this dissertation was to design a modular object detection system based on deep neural networks, tailored for detecting tiny and multi-scale objects in high-resolution images. The system was developed with an emphasis on achieving not only high detection accuracy but also low computational cost and fast inference, enabling deployment in resource-constrained environments such as mobile robots. The proposed system comprises several components working together to balance accuracy and efficiency:

- a low-resolution, segmentation-based `ROI Estimator` that selects Regions of Interest (ROIs) from each frame; analyzed in detail in Chapter 4,
- a `ROI Prediction Module` that leverages object tracking in video mode to restore regions potentially missed by the `Estimator`; described in Chapter 5,
- a `ROI Fusion Module` that combines the masks generated by both the `Estimation` and `Prediction` branches into a single unified ROI mask; presented in Chapter 5,
- a `Detection Window Proposal Block` responsible for converting the fused ROI mask into a set of detection windows; discussed in Chapter 4,
- a lightweight `Local Object Detector` applied independently to each cropped window; implementation details are provided in both Chapter 4, and Chapter 5,
- a `Global Filtering Module`, incorporating the proposed `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)` algorithms, designed specifically to address redundancy and fragmentation in window-based detection; described and evaluated in Chapter 6.

Together, these components form a complete detection framework optimized for small object detection in large-scale images, while remaining suitable for real-time deployment on embedded systems. The proposed system is evaluated against several baselines throughout this work.

The main experimental results in Chapter 4 compare both TinyROI (an initial version of the framework) and SegTrackDetect (a final, more robust version) with sliding-window implementations integrated into each system. For broader context, the comparison also includes SAHI [2], a widely used open-source framework for sliding-window detection. In addition, the ablation studies presented in Chapter 4 analyze three distinct detection strategies (single-shot detection on downsampled full images, the sliding-window approach, and the estimation-based TinyROI) highlighting the strengths and limitations of each. In Chapter 5, the complete SegTrackDetect framework, incorporating both the `Estimator` and `Predictor`, is benchmarked against state-of-the-art object detection methods across three categories: general-purpose detectors, tiny object detectors, and video object detectors, providing a broad quantitative comparison. Finally, Chapter 6 evaluates the proposed `Global Filtering Block`, which includes both `OBS` and `OBM`, against the baseline Non-Maximum Suppression (NMS) approach.

While modern general-purpose object detection methods perform well on medium and large objects in benchmark datasets such as COCO, their performance on small objects remains significantly lower. Furthermore, such benchmarks typically include relatively low-resolution images and do not require downsampling of the input data. When these general-purpose methods are applied to high-resolution inputs, both processing time and computational resource requirements increase dramatically. This is especially problematic for transformer-based architectures, which often require several gigabytes of GPU memory to process even moderately sized images. Increasing the input resolution, which is essential for detecting tiny objects, therefore leads to a steep rise in memory and computation demands. Conversely, reducing the input resolution causes the features of small objects to vanish, making them difficult or impossible to detect. Additionally, most modern detectors include downsampling layers in their feature extractors, which further degrade the visibility of small-scale objects. A straightforward way to address this is to apply a small detection window over the high-resolution image using a regular sliding grid. This increases the relative size of tiny objects within each window and improves recall. However, this approach is computationally expensive and unsuitable for real-time applications, particularly when detections are sparse and most regions contain only background. Additionally, in multi-scale detection, larger objects are more likely to be split across multiple windows when using a regular grid. The proposed system employs lightweight, shallow, general-purpose detectors in the `Local Object Detection Block`. Their shallow design mitigates the effects of downsampling in backbone networks while keeping the system efficient and fast.

To overcome these limitations, a class of methods known as window-based or focus-and-detect approaches has emerged. These methods aim to select detection windows in a more data-driven approach, rather than relying on exhaustive sliding-window scanning. Typically, they include a learned preprocessing step that identifies informative regions. In the proposed system, a lightweight, low-resolution binary segmentation network is used to estimate Regions of Interest (ROIs), which are then used to select areas for full-resolution detection. Unlike other methods that rely on clustering or density estimation, the proposed approach generates precise ROIs in both dense and sparse scenes. This is achieved by directly training the segmentation network on detection labels, without requiring additional annotations or assumptions about object

density. Moreover, the proposed system eliminates the need for separate scale estimation mechanisms, which are often found in related methods. By combining sliding-window and crop-and-resize strategies within large ROIs, the system can detect objects across a wide range of scales. The complete design and evaluation of the ROI Estimation-based system are discussed in detail in Chapter 4. In addition to comparing TinyROI and SegTrackDetect against single-shot detectors and sliding-window methods, this chapter includes extensive ablation studies that guided the development of the final SegTrackDetect pipeline. These studies cover the training pipeline for the `ROI Estimator` (including input resolution choices and label generation methods), the filtering strategies within the `Detection Window Proposal Block`, and the large ROI handling strategy within the same module. Furthermore, this chapter demonstrated the initial need for a `Global Filtering Block`, which was later extended through the development of the novel `OBS` and `OBM` algorithms, described in Chapter 6. Overall, the analyses presented in Chapter 4 support Auxiliary Thesis #1, which states: "Estimating ROIs using a deep neural network model enhances both detection quality and inference speed compared to the naive sliding-window approach".

In Chapter 5, the proposed SegTrackDetect system is extended with a second ROI source derived from a SORT-like object tracking module. This addition was inspired by how humans often detect small or partially occluded objects by leveraging motion cues. Since many mobile robotics applications operate on temporally coherent video streams rather than independent images, incorporating a lightweight tracking component becomes feasible and, when implemented effectively, can improve detection quality without introducing significant computational overhead. Alternatively, it enables a trade-off: maintaining comparable detection quality while reducing computational demands by lowering the input resolution required for the `ROI Estimator`. This design enhances the system's adaptability to embedded, resource-constrained environments and demonstrates that tracking-based `ROI Prediction` serves as an effective complement to segmentation-based `Estimation`. Extensive experiments described in Chapter 5 demonstrate that incorporating an additional ROI source from the object tracker makes it possible to improve detection quality, inference speed, and computational efficiency over state-of-the-art tiny object detection methods. First, the final SegTrackDetect system was compared to several state-of-the-art detectors and consistently delivered competitive results in terms of detection quality, while significantly reducing the number of trainable parameters. Later, ablation studies showed that these improvements in computational efficiency and processing speed, along with strong detection performance, were directly enabled by the inclusion of the `ROI Prediction Module`. Specifically, thanks to object tracking, the `Estimator` can operate at lower input resolutions without sacrificing detection quality, resulting in faster inference and reduced resource usage. Together, these findings validate Auxiliary Thesis #2.

While window-based approaches, such as the one proposed in this thesis, significantly improve recall for tiny objects by increasing their relative size and preserving the original detail, they can also introduce new challenges. In particular, these methods often produce partial detections near the edges of detection windows, where objects are only partially visible. Due to the limited overlap between such fragments, standard Non-Maximum Suppression (NMS) struggles to filter them effectively, often leading to clusters of redundant false positives. To address this,

the proposed `Overlapping Box Suppression (OBS)` algorithm incorporates the detection window coordinates into the filtering process, allowing it to better identify and suppress partial detections from the perspective of each window. By increasing the effective overlap with fragmentary detections, `OBS` improves filtering performance even at lower IoU thresholds, reducing false positives without eliminating true positives. In addition, the `Overlapping Box Merging (OBM)` algorithm is introduced to merge partial detections in cases where no complete detection is present, a common issue in sliding-window pipelines and multi-scale object scenarios. Together, these two algorithms are described in detail in Chapter 6. There, `OBS` and `OBM` are extensively evaluated against two baselines: a system using only standard NMS and a variant with no global filtering. Experiments conducted on the SeaDronesSee dataset demonstrate that "The common issue of false positive partial detections in window-based systems can be mitigated through post-processing techniques such as `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`", directly supporting Auxiliary Thesis #3.

The final part of this thesis, presented in Chapter 7, focuses on the implementation of this work, conducted as part of the "Doktorat Wdrożeniowy" program. The proposed SegTrackDetect system is released as an open-source, highly customizable framework that not only provides the precise implementations described in Chapters 4 and 5, but also serves as a flexible foundation for developing window-based detection methods. All major components, such as the `ROI Estimator`, `ROI Predictor`, and `Local Detector`, are designed to be easily customizable. Users can modify pre- and post-processing functions or even integrate entirely new models, encouraging extensive experimentation. To the best of the author's knowledge, this is the first high-level window-based detection framework that offers such a degree of customization. Furthermore, Chapter 7 presents a brief analysis of deploying the system on embedded devices. Due to proprietary restrictions on the full SegTrackDetect implementation, the experiments focus on the initial TinyROI framework, examining the quality-speed trade-offs on resource-constrained embedded hardware.

Together, these contributions establish SegTrackDetect as a customizable framework that not only advances window-based tiny object detection through novel algorithmic improvements but also supports practical deployment, addressing real-world constraints encountered in mobile robotics applications.

8.2 Conclusions

This work addressed the problem of efficient tiny and multi-scale object detection in high-resolution video, with a particular emphasis on robotics and UAV-based applications. To balance accuracy and computational efficiency, a modular detection system SegTrackDetect was proposed, combining segmentation-based `ROI Estimation`, tracking-based `ROI Prediction`, and a custom `Global Filtering Module` designed to handle partial false positive detections common in window-based systems. Together, these components enable high detection performance with reduced model sizes, supporting deployment in real-time and resource-constrained environments.

The first core contribution, a segmentation-based `ROI Estimation Module`, discussed in detail in Chapter 4, enables context-aware selection of detection windows. Unlike uniform tiling strategies, this approach uses coarse segmentation masks to guide window placement, aligning it with meaningful image features and preserving detection context more effectively than sliding-window methods. The initial version of the framework, TinyROI, was evaluated against two baselines, single-shot detection with downsampling and a uniform sliding-window approach, all implemented within the same codebase to ensure fair comparison. Experiments on the challenging Mapillary Traffic Sign Dataset (MTSD) showed that TinyROI achieved comparable detection quality to the sliding-window baseline while reducing processing time by nearly a factor of eight. Both window-based methods substantially improved detection of small objects and maintained performance for medium-sized ones, though they initially underperformed on large objects. Further analysis attributed this drop in quality to the training procedure of the detector, which was kept identical across all evaluated methods, single-shot, sliding-window, and TinyROI, and trained on a mix of cropped and downsampled images to maintain consistency. Although this ensured fair comparison, it also introduced a compromise: the detector could not be fully optimized for the window-based approaches. In subsequent experiments with SegTrackDetect, the training strategy was adjusted to include only cropped images. This change led to noticeable improvements in detecting larger objects. Additional improvements included a large ROI handling strategy, which allowed lowering the detector input size, while ensuring that exceptionally large ROIs were still accurately processed. This enhanced computational efficiency and reduced inference time, as confirmed by extensive evaluation on three datasets: MTSD, SeaDronesSee, and ZebraFish (Tab. 4.1). SegTrackDetect consistently outperformed both TinyROI and all three sliding-window variants in terms of detection quality and speed. On SeaDronesSee and ZebraFish, the system delivered robust performance across all object sizes, achieving 44.5 FPS and 78.0 FPS, respectively. In MTSD, while SegTrackDetect was the fastest, it did not yield a quality improvement, again likely due to the previously discussed limitations of the detector's training setup. Compared to naive sliding-window strategies, the learned `ROI Estimator` in SegTrackDetect reduced redundant computation and improved detection quality by better preserving spatial context. These findings support the first Auxiliary Thesis of this work: "estimating ROIs using a deep neural network model enhances both detection quality and inference speed compared to the naive sliding-window approach".

The second major contribution is the integration of object tracking into ROI selection via a lightweight `ROI Prediction Module`, as detailed in Chapter 5. By leveraging temporal continuity, this module effectively recovers regions missed by the low-resolution segmentation branch. Extensive quantitative and qualitative evaluations demonstrate that the proposed dual-ROI strategy outperforms several state-of-the-art object detectors on the challenging SeaDronesSee and DroneCrowd datasets. On both datasets, the SegTrackDetect system achieves the highest detection accuracy for the smallest objects, while also delivering competitive performance for larger objects in the multi-scale detection scenario represented by SeaDronesSee. Notably, this high detection quality is achieved with the smallest number of parameters among all compared methods, highlighting the lightweight nature of the proposed system.

An ablation study confirms that the combination of high accuracy, low computational complexity, and fast inference speeds is a direct result of integrating the `ROI Prediction Module`. On

SeaDronesSee, the fusion-based configuration with a 128×192 input resolution achieves 52.4% AP, 60.2% AR, and an ROI selection speed of 303 FPS, closely matching the 52.9% AP and 61.5% AR of a 448×768 segmentation-only baseline, which operates at only 82 FPS. These results show that tracking enables lower-resolution processing without degrading detection quality. Qualitative analysis further supports these findings: the segmentation branch is most effective at detecting newly appearing objects, while the tracking branch compensates for inconsistently segmented regions, especially in the case of very tiny objects. Together, these results validate the second Auxiliary Thesis: "incorporating an additional ROI source, such as an object tracker, into the ROI-based object detection system further improves detection quality, inference speed, and computational efficiency over state-of-the-art tiny object detection methods".

The third major contribution of this thesis is the introduction of a dedicated `Global Filtering Module`, designed to address a key limitation of window-based detection systems: the presence of partial detections in overlapping regions between detection windows. This module introduces two complementary algorithms, `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`. `OBS` improves upon standard Non-Maximum Suppression (NMS) by leveraging a cost-based filtering scheme that integrates object size, confidence, and spatial overlap. Unlike NMS, which relies solely on IoU and often discards complete detections due to low overlap or suboptimal confidence, `OBS` utilizes the coordinates of detection windows to infer partial views of full objects from each window's perspective. Combined with a unified cost matrix, this approach enables high overlap between partial detections and partial object views, effectively prioritizing the most complete instances. `OBM` complements `OBS` by merging fragmented detections across intersecting windows, allowing recovery of objects that lack a single complete detection. Extensive experiments on the SeaDronesSee dataset confirm the effectiveness of both algorithms: `OBS` consistently outperforms NMS in terms of precision and recall, especially under low-IoU conditions, while `OBM` provides additional gains by consolidating fragments into full detections. When applied together, `OBS` and `OBM` improve F1 scores by more than 10 percentage points in some settings. These findings validate the third Auxiliary Thesis: "false positive partial detections in window-based systems can be reduced through post-processing techniques such as `Overlapping Box Suppression (OBS)` and `Overlapping Box Merging (OBM)`".

Overall, this work demonstrates that selective full-resolution inference on dynamically chosen image subregions, guided by learned segmentation and tracking-based ROI selection, offers a highly effective strategy for detecting small and tiny objects in high-resolution imagery. The SegTrackDetect system integrates three core components: a segmentation-based `ROI Estimator` that improves upon naive sliding-window methods, a tracking-based `ROI Predictor` that enhances temporal consistency and recall, and a custom `Global Filtering Module` that mitigates common issues with partial detections using `OBS` and `OBM`. Together, these components confirm all proposed theses: that learned `ROI Estimation` and tracking-based `Prediction` considerably improve both detection quality and inference speed compared to traditional methods, and that postprocessing specifically designed for overlapping-window artifacts further enhances detection quality. Evaluations across diverse datasets demonstrate that SegTrackDetect provides a scalable, real-time solution for small object detection in high-resolution images.

8.3 Future work

Most limitations of the initial system (TinyROI) were addressed during the development of the SegTrackDetect framework. However, as noted in Chapter 2, the IoU-based assignment commonly used for training and evaluating deep neural networks has several shortcomings when applied to tiny objects. The training of the `Estimator` could be improved by replacing IoU with a metric specifically designed for tiny objects, such as the Normalized Wasserstein Distance (NWD) [153] adapted for semantic segmentation. This limitation is particularly pronounced for the `Estimator`, which is trained on highly downsampled images, in contrast to the `Local Object Detector`, where objects are processed at their original resolution.

While the proposed `Local Object Detection Block` has proven highly effective for small and tiny objects under a relative-size definition, the framework could be further extended to handle small objects defined in absolute terms – that is, objects with very low pixel counts where preserving the original resolution alone may be insufficient. To achieve this, traditional methods for small object detection could be integrated within the `Local Detector`, such as Scale-Aware approaches based on Feature Pyramid Networks (FPN) [82]. Additionally, video object detection techniques could be leveraged to exploit temporal continuity during training, further improving detection robustness.

Another promising direction is to enhance the `ROI Prediction Module` with non-linear motion models. The current approach assumes linear motion between frames, which can introduce inaccuracies in scenes with abrupt or complex object trajectories. Incorporating more advanced predictive models could improve ROI stability and object recovery in such scenarios. Additionally, the `ROI Estimator` could be applied at higher resolution at the start of a sequence during inference to improve recall for smaller objects, which would then be accurately tracked by the `Predictor`. However, a more detailed analysis would be required to evaluate the speed-quality trade-offs in this setup.

The `Global Filtering Module`, comprising `OBS` and `OBM`, also presents opportunities for further development. Future research could explore class-aware merging strategies, particularly for cases involving visually similar object classes. Moreover, a learned suppression and merging mechanism based on the outputs of `OBS` and `OBM` could generalize these techniques to a broader range of scenarios, further enhancing the robustness and adaptability of window-based detection systems.

Although the modular design of SegTrackDetect ensures flexibility and ease of customization, future work could investigate tighter integration between the `ROI Estimation` and `Detection Modules`. Specifically, unifying their backbones via shared feature extractors or multitask architectures may significantly reduce computational overhead and memory usage, enabling more efficient deployment without compromising detection quality. Such integration was intentionally avoided in this thesis to maintain modularity, but it may offer clear advantages in resource-constrained environments.

Collectively, these extensions have the potential to further improve the quality, efficiency, and adaptability of ROI-based small object detection systems for high-resolution images, particularly in resource-constrained environments.

Bibliography

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[2] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection. In *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, October 2022. doi: 10.1109/icip46576.2022.9897990. URL `http://dx.doi.org/10.1109/ICIP46576.2022.9897990`.

[3] Jumabek Alikhanov and Hakil Kim. Online Action Detection in Surveillance Scenarios: A Comprehensive Review and Comparative Study of State-of-the-Art Multi-Object Tracking Methods. *IEEE Access*, 11:68079–68092, 2023. doi: 10.1109/ACCESS.2023.3292539.

[4] Samreen Anjum and Danna Gurari. CTMC: Cell Tracking With Mitosis Detection Dataset Challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 982–983, 2020.

[5] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Finding tiny faces in the wild with generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–30, 2018.

[6] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. SOD-MTGAN: Small Object Detection via Multi-Task Generative Adversarial Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 206–221, 2018.

[7] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016.

[8] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[9] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[10] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[11] Erik Bochinski, Tobias Senst, and Thomas Sikora. Extending IOU based multi-object tracking by visual information. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[13] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – improving object detection with one line of code. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5562–5570, 2017. doi: 10.1109/ICCV.2017.593.

[14] Elizabeth Bondi, Raghav Jain, Palash Aggrawal, Saket Anand, Robert Hannaford, Ashish Kapoor, Jim Piavis, Shital Shah, Lucas Joppa, Bistra Dilkina, et al. BIRDSAI: A dataset for detection and tracking in aerial thermal infrared videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1747–1756, 2020.

[15] Brais Bosquet, Manuel Mucientes, and Victor M Brea. STDnet: Exploiting high resolution feature maps for small object detection. *Engineering Applications of Artificial Intelligence*, 91:103615, 2020.

[16] Brais Bosquet, Daniel Cores, Lorenzo Seidenari, Víctor M Brea, Manuel Mucientes, and Alberto Del Bimbo. A full data augmentation pipeline for small object detection based on generative adversarial networks. *Pattern Recognition*, 133:108998, 2023.

[17] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-centric SORT: Rethinking SORT for robust multi-object tracking. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9686–9696, 2023. doi: 10.1109/CVPR52729.2023.00934.

[18] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[19] Changrui Chen, Yu Zhang, Qingxuan Lv, Shuo Wei, Xiaorui Wang, Xin Sun, and Junyu Dong. RRNet: A hybrid detector for object detection in drone-captured images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[20] Chenyi Chen, Ming-Yu Liu, Oncel Tuzel, and Jianxiong Xiao. R-CNN for small object detection. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part V 13*, pages 214–230. Springer, 2017.

[21] Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *2018 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2018.

[22] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10337–10346, 2020.

[23] Gong Cheng, Junwei Han, Peicheng Zhou, and Lei Guo. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 98:119–132, 2014.

[24] Gong Cheng, Xiang Yuan, Xiwen Yao, Kebing Yan, Qinghua Zeng, Xingxing Xie, and Junwei Han. Towards large-scale small object detection: Survey and benchmarks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[25] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. TransMOT: Spatial-Temporal Graph Transformer for Multiple Object Tracking. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. doi: 10.1109/WACV56688.2023.00485.

[26] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[27] Yiming Cui, Liqi Yan, Zhiwen Cao, and Dongfang Liu. TF-Blender: Temporal Feature Blender for Video Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8138–8147, October 2021.

[28] Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Tao: A large-scale benchmark for tracking any object. In *European conference on computer vision*, pages 436–454. Springer, 2020.

[29] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020.

[30] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7023–7032, 2019.

[31] Sutao Deng, Shuai Li, Ke Xie, Wenfeng Song, Xiao Liao, Aimin Hao, and Hong Qin. A global-local self-adaptive network for drone-view object detection. *IEEE Transactions on Image Processing*, 30:1556–1569, 2020.

[32] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/TPAMI.2021.3117983.

[33] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 370–386, 2018.

[34] Chengzhen Duan, Zhiwei Wei, Chi Zhang, Siying Qu, and Hongpeng Wang. Coarse-grained Density Map Guided Object Detection in Aerial Images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2789–2798, 2021.

[35] Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, and Yubin Kuang. The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale. In *European Conference on Computer Vision*, pages 68–84. Springer, 2020.

[36] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[37] Jiamei Fu, Xian Sun, Zhirui Wang, and Kun Fu. An anchor-free method based on feature balancing and refinement network for multiscale ship detection in SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 59(2):1331–1344, 2020.

[38] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6926–6935, 2018.

[39] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[40] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237, 2013.

[41] Johannes Gilg, Torben Teepe, Fabian Herzog, Philipp Wolters, and Gerhard Rigoll. Do We Still Need Non-Maximum Suppression? Accurate Confidence Estimates and Implicit Duplication Modeling with IoU-Aware Calibration. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4838–4847, 2024. doi: 10.1109/WACV57701.2024.00478.

[42] Andreu Girbau, Xavier Giró-i Nieto, Ignasi Rius, and Ferran Marqués. Multiple object tracking with mixture density networks for trajectory estimation. *arXiv preprint arXiv:2106.10950*, 2021.

[43] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[44] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[45] Munkhjargal Gochoo, Munkh-Erdene Otgonbold, Erkhembayar Ganbold, Jun-Wei Hsieh, Ming-Ching Chang, Ping-Yang Chen, Byambaa Dorj, Hamad Al Jassmi, Ganzorig Batnasan, Fady Alnajjar, et al. FishEye8K: A benchmark and dataset for fisheye camera object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5305–5313, 2023.

[46] Yuqi Gong, Xuehui Yu, Yao Ding, Xiaoke Peng, Jian Zhao, and Zhenjun Han. Effective fusion factor in FPN for tiny object detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1160–1168, 2021.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[48] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *arXiv preprint arXiv:1703.06870*, 2017.

[49] Mingbo Hong, Shuiwang Li, Yuchao Yang, Feiyu Zhu, Qijun Zhao, and Li Lu. SSPNet: Scale selection pyramid network for tiny person detection from UAV images. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021.

[50] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *International conference on machine learning*, pages 4364–4375. PMLR, 2020.

[51] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–8. Ieee, 2013.

[52] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[53] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017.

[54] Xiaowei Hu, Xuemiao Xu, Yongjie Xiao, Hao Chen, Shengfeng He, Jing Qin, and Pheng-Ann Heng. SINet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE transactions on intelligent transportation systems*, 20(3):1010–1019, 2018.

[55] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[56] Hong Ji, Zhi Gao, Tiancan Mei, and Bharath Ramesh. Vehicle detection in remote sensing images leveraging on simultaneous super-resolution. *IEEE Geoscience and Remote Sensing Letters*, 17(4):676–680, 2019.

[57] Kui Jiang, Zhongyuan Wang, Peng Yi, Guangcheng Wang, Tao Lu, and Junjun Jiang. Edge-enhanced GAN for remote sensing image superresolution. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5799–5812, 2019.

[58] Licheng Jiao, Ruohan Zhang, Fang Liu, Shuyuan Yang, Biao Hou, Lingling Li, and Xu Tang. New Generation Deep Learning for Video Object Detection: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 33(8):3195–3215, 2022. doi: 10.1109/TNNLS.2021.3053249.

[59] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. doi: 10.1109/TPAMI.2011.239.

[60] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[61] Jung Uk Kim, Sungjune Park, and Yong Man Ro. Robust small-scale pedestrian detection with cued recall via memory learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3050–3059, 2021.

[62] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment Anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi: 10.1109/ICCV51070.2023.00371.

[63] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019.

[64] Aleksandra Kos. Overlapping Box Suppression Algorithm for Window-Based Object Detection. In *Progress in Polish Artificial Intelligence Research 5 : Proceedings of the 5th Polish Conference on Artificial Intelligence (PP-RAI'2024), 18-20.04.2024, Warsaw, Poland*, pages 325–330. Politechnika Warszawska, 2024.

[65] Aleksandra Kos. Overlapping Box Suppression and Merging Algorithms for Window-Based Object Detection. *Foundations of Computing and Decision Sciences*, 50(3):403–423, 2025. doi: 10.2478/fcds-2025-0016. URL `https://doi.org/10.2478/fcds-2025-0016`.

[66] Aleksandra Kos and Karol Majek. BDOT10k-seg: A dataset for semantic segmentation. *Wojciechowski A.(Ed.), Lipiński P.(Ed.)., Progress in Polish Artificial Intelligence Research 4, Seria: Monografie Politechniki Łódzkiej Nr. 2437, Wydawnictwo Politechniki Łódzkiej, Łódź 2023, ISBN 978-83-66741-92-8, doi: 10.34658/9788366741928.*, 2023.

[67] Aleksandra Kos, Karol Majek, and Dominik Belter. Where to look for tiny objects? ROI prediction for tiny object detection in high resolution images. In *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 721–726. IEEE, 2022.

[68] Aleksandra Kos, Karol Majek, and Dominik Belter. Dataset Augmentation for Detecting Small Objects in Fisheye Road Images. In *Proceedings of the 24th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2025), Zakopane, 2025 (in press)*, 2025.

[69] Aleksandra Kos, Karol Majek, and Dominik Belter. Enhanced lightweight detection of small and tiny objects in high-resolution images using object tracking-based region of interest proposal. *Engineering Applications of Artificial Intelligence*, 153:110852, 2025. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2025.110852. URL `https://www.sciencedirect.com/science/article/pii/S0952197625008528`.

[70] Aleksandra Kos, Karol Majek, and Dominik Belter. SegTrackDetect: A window-based framework for tiny object detection via semantic segmentation and tracking. *SoftwareX*, 30:102110, 2025.

[71] Onur Can Koyun, Reyhan Kevser Keser, İbrahim Batuhan Akkaya, and Behçet Uğur Töreyin. Focus-and-Detect: A small object detection framework for aerial images. *Signal Processing: Image Communication*, 104:116675, 2022.

[72] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[73] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xView: Objects in Context in Overhead Imagery. *arXiv preprint arXiv:1802.07856*, 2018.

[74] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[75] Changlin Li, Taojiannan Yang, Sijie Zhu, Chen Chen, and Shanyue Guan. Density map guided object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 190–191, 2020.

[76] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. DSFD: dual shot face detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5060–5069, 2019.

[77] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017.

[78] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:296–307, 2020.

[79] Yangyang Li, Qin Huang, Xuan Pei, Yanqiao Chen, Licheng Jiao, and Ronghua Shang. Cross-layer attention network for small object detection in remote sensing imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:2148–2161, 2020.

[80] Xi Liang, Jing Zhang, Li Zhuo, Yuzhao Li, and Qi Tian. Small object detection in unmanned aerial vehicle images using feature fusion and scaling-based single shot detector with spatial context analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1758–1770, 2019.

[81] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[82] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[83] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[84] Songtao Liu, Di Huang, and Yunhong Wang. Learning spatial fusion for single-shot object detection. *arXiv preprint arXiv:1911.09516*, 2019.

[85] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[86] Yingjie Liu, Fengbao Yang, and Peng Hu. Small-object detection in UAV-captured images via multi-branch parallel feature pyramid networks. *IEEE access*, 8:145740–145750, 2020.

[87] Ziming Liu, Guangyu Gao, Lin Sun, and Li Fang. IPG-net: Image pyramid guidance network for small object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 1026–1027, 2020.

[88] Xiaocong Lu, Jian Ji, Zhiqi Xing, and Qiguang Miao. Attention and feature fusion SSD for remote sensing object detection. *IEEE Transactions on Instrumentation and Measurement*, 70:1–9, 2021.

[89] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking. *International Journal of Computer Vision*, 129(2):548–578, 2021.

[90] Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, and Kris Kitani. Deep OC-Sort: Multi-pedestrian tracking by adaptive re-identification. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 3025–3029, 2023. doi: 10.1109/ICIP49359.2023.10222576.

[91] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[92] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

[93] Mahyar Najibi, Pouya Samangouei, Rama Chellappa, and Larry S Davis. SSH: Single stage headless face detector. In *Proceedings of the IEEE international conference on computer vision*, pages 4875–4884, 2017.

[94] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017.

[95] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9725–9734, 2019.

[96] F Ozge Unel, Burak O Ozkalayci, and Cevahir Cigla. The power of tiling for small object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[97] Xingjia Pan, Fan Tang, Weiming Dong, Yang Gu, Zhichao Song, Yiping Meng, Pengfei Xu, Oliver Deussen, and Changsheng Xu. Self-supervised feature augmentation for large image object detection. *IEEE Transactions on Image Processing*, 29:6745–6758, 2020.

[98] Ye Pan and Feng Dong. Suppression and Enhancement of Overlapping Bounding Boxes Scores in Object Detection. In *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 1–4, 2019. doi: 10.1109/ISSPIT47144.2019.9001826.

[99] Jiangmiao Pang, Cong Li, Jianping Shi, Zhihai Xu, and Huajun Feng. $\mathcal{R}^2$-CNN: Fast tiny object detection in large-scale remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5512–5524, 2019. doi: 10.1109/TGRS.2019.2899955.

[100] Malte Pedersen, Joakim Bruslund Haurum, Stefan Hein Bengtson, and Thomas B Moeslund. 3D-ZeF: A 3d zebrafish tracking benchmark dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2436, 2020.

[101] Aleksis Pirinen and Cristian Sminchisescu. Deep Reinforcement Learning of Region Proposal Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6945–6954, 2018.

[102] George Plastiras, Christos Kyrkou, and Theocharis Theocharides. Efficient ConvNet-Based Object Detection for Unmanned Aerial Vehicles by Selective Tile Processing. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, ICDSC '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450365116. doi: 10.1145/3243394.3243692.

[103] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. DetectoRS: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10213–10224, 2021.

[104] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-Net: Going deeper with nested U-structure for salient object detection. *Pattern recognition*, 106:107404, 2020.

[105] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[106] Qiong Ran, Qing Wang, Boya Zhao, Yuanfeng Wu, Shengliang Pu, and Zijin Li. Lightweight oriented object detection using multiscale context and enhanced channel attention in remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:5786–5795, 2021.

[107] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017. doi: 10.1109/CVPR.2017.690.

[108] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[109] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[110] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[111] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection over Union: A Metric and a Loss for Bounding Box Regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[112] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016.

[113] Si-Dong Roh and Ki-Seok Chung. DiffusionVID: Denoising object boxes with spatio-temporal conditioning for video object detection. *IEEE Access*, 2023.

[114] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[115] Azriel Rosenfeld and Mark Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, 100(5):562–569, 1971.

[116] Vít Růžička and Franz Franchetti. Fast and accurate object detection in high resolution 4K and 8K video using GPUs. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2018.

[117] Niels Ole Salscheider. FeatureNMS: Non-Maximum Suppression by Learning Feature Embeddings. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7848–7854, 2021. doi: 10.1109/ICPR48806.2021.9412930.

[118] Vladislav Igorevich Shakhuro and AS Konouchine. Russian Traffic Sign Images Dataset. *Computer Optics*, 40(2):294–300, 2016.

[119] Avishag Shapira, Alon Zolfi, Luca Demetrio, Battista Biggio, and Asaf Shabtai. Phantom Sponges: Exploiting Non-Maximum Suppression to Attack Deep Object Detectors. In

2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pages 4560–4569, 2023. doi: 10.1109/WACV56688.2023.00455.

[120] Andrew J. Shepley, Greg Falzon, Paul Kwan, and Ljiljana Brankovic. Confluence: A Robust Non-IoU Alternative to Non-Maxima Suppression in Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):11561–11574, 2023. doi: 10.1109/TPAMI.2023.3273210.

[121] Yuheng Shi, Naiyan Wang, and Xiaojie Guo. YOLOV: Making still image object detectors great at video object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2254–2262, 2023.

[122] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted Boxes Fusion: Ensembling Boxes from Different Object Detection Models. *Image and Vision Computing*, pages 1–6, 2021.

[123] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[124] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple Object Tracking with Transformer. *arXiv preprint arXiv:2012.15460*, 2020.

[125] Ramana Sundararaman, Cedric De Almeida Braga, Eric Marchand, and Julien Pettre. Tracking Pedestrian Heads in Dense Crowd. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3865–3875, 2021.

[126] Xu Tang, Daniel K Du, Zeqiang He, and Jingtuo Liu. Pyramidbox: A context-assisted single shot face detector. In *Proceedings of the European conference on computer vision (ECCV)*, pages 797–813, 2018.

[127] Yunjie Tian, Qixiang Ye, and David Doermann. YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv preprint arXiv:2502.12524*, 2025.

[128] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[129] Radu Timofte, Karel Zimmermann, and Luc Van Gool. Multi-view traffic sign detection, recognition, and 3D localisation. *Machine vision and applications*, 25(3):633–647, 2014.

[130] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2): 154–171, 2013.

[131] Adam Van Etten, Dave Lindenbaum, and Todd M. Bacastow. SpaceNet: A Remote Sensing Dataset and Challenge Series. *arXiv preprint arXiv:1807.01232*, 2018.

[132] Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and 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*, pages 2260–2270, 2022.

[133] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and

segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7942–7951, 2019.

[134] Paul Voigtlaender, Lishu Luo, Chun Yuan, Yong Jiang, and Bastian Leibe. Reducing the annotation effort for video object segmentation datasets. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3060–3069, 2021.

[135] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et al. YOLOv10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 37: 107984–108011, 2025.

[136] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.

[137] Haoran Wang, Zexin Wang, Meixia Jia, Aijin Li, Tuo Feng, Wenhua Zhang, and Licheng Jiao. Spatial attention for multi-scale feature refinement for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[138] Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny object detection in aerial images. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3791–3798. IEEE, 2021.

[139] Robert J Wang, Xiang Li, and Charles X Ling. Pelee: A Real-Time Object Detection System on Mobile Devices. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1967–1976. Curran Associates, Inc., 2018.

[140] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. InternImage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14408–14419, 2023.

[141] Xiaobin Wang, Dekang Zhu, and Ye Yan. Towards Efficient Detection for Small Objects via Attention-Guided Detection Network and Data Augmentation. *Sensors*, 22(19):7663, 2022.

[142] Yi Wang, Youlong Yang, and Xi Zhao. Object detection using clustering algorithm adaptive searching regions in aerial images. In *European Conference on Computer Vision*, pages 651–664. Springer, 2020.

[143] Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint object detection and multi-object tracking with graph neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13708–13715. IEEE, 2021.

[144] Zhongdao Wang, Hengshuang Zhao, Ya-Li Li, Shengjin Wang, Philip Torr, and Luca Bertinetto. Do Different Tracking Tasks Require Different Appearance Models? *Advances in Neural Information Processing Systems*, 34:726–738, 2021.

[145] Syed Waqas Zamir, Aditya Arora, Akshita Gupta, Salman Khan, Guolei Sun, Fahad Shahbaz Khan, Fan Zhu, Ling Shao, Gui-Song Xia, and Xiang Bai. isaid: A large-scale dataset for instance segmentation in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 28–37, 2019.

[146] Mark Weber, Jun Xie, Maxwell Collins, Yukun Zhu, Paul Voigtlaender, Hartwig Adam, Bradley Green, Andreas Geiger, Bastian Leibe, Daniel Cremers, et al. STEP: Segmenting and Tracking Every Pixel. *arXiv preprint arXiv:2102.11859*, 2021.

[147] Longyin Wen, Dawei Du, Pengfei Zhu, Qinghua Hu, Qilong Wang, Liefeng Bo, and Siwei Lyu. Detection, tracking, and counting meets drones in crowds: A benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7812–7821, 2021.

[148] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 748–756. IEEE, 2018.

[149] Jialian Wu, Chunluan Zhou, Qian Zhang, Ming Yang, and Junsong Yuan. Self-mimic learning for small-scale pedestrian detection. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2012–2020, 2020.

[150] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12352–12361, 2021.

[151] Xingxing Xie, Gong Cheng, Qingyang Li, Shicheng Miao, Ke Li, and Junwei Han. Fewer is more: Efficient object detection in large aerial images. *Science China Information Sciences*, 67(1):1–19, 2024.

[152] Chang Xu, Jinwang Wang, Wen Yang, and Lei Yu. DOT Distance for Tiny Object Detection in Aerial Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1192–1201, 2021.

[153] Chang Xu, Jinwang Wang, Wen Yang, Huai Yu, Lei Yu, and Gui-Song Xia. Detecting tiny objects in aerial images: A normalized Wasserstein distance and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 190:79–93, 2022.

[154] Chang Xu, Jinwang Wang, Wen Yang, Huai Yu, Lei Yu, and Gui-Song Xia. RFLA: Gaussian receptive field based label assignment for tiny object detection. In *European conference on computer vision*, pages 526–543. Springer, 2022.

[155] Jingtao Xu, Ya-Li Li, and Shengjin Wang. AdaZoom: Towards scale-aware large scene object detection. *IEEE Transactions on Multimedia*, 25:4598–4609, 2022.

[156] Chenhongyi Yang, Zehao Huang, and Naiyan Wang. QueryDet: Cascaded sparse query for accelerating high-resolution small object detection. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 13668–13677, 2022.

[157] Fan Yang, Heng Fan, Peng Chu, Erik Blasch, and Haibin Ling. Clustered object detection in aerial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8311–8320, 2019.

[158] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[159] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Xian Sun, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8232–8241, 2019.

[160] Kai Yi, Zhiqiang Jian, Shitao Chen, and Nanning Zheng. Feature selective small object detection via knowledge-based recurrent attentive neural network. *arXiv preprint arXiv:1803.05263*, 2018.

[161] Senthil Yogamani, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9308–9318, 2019.

[162] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2636–2645, 2020.

[163] Xuehui Yu, Yuqi Gong, Nan Jiang, Qixiang Ye, and Zhenjun Han. Scale match for tiny person detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1257–1265, 2020.

[164] Xuehui Yu, Pengfei Chen, Di Wu, Najmul Hassan, Guorong Li, Junchi Yan, Humphrey Shi, Qixiang Ye, and Zhenjun Han. Object Localization under Single Coarse Point Supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4868–4877, 2022.

[165] Du Yunhao, Zhao Zhicheng, Song Yang, Zhao Yanyun, Su Fei, Gong Tao, and Meng Hongying. StrongSORT: Make DeepSORT great again. *IEEE Transactions on Multimedia*, 25:8725–8737, 2023. doi: 10.1109/TMM.2023.3240881.

[166] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. MOTR: End-to-End Multiple-Object Tracking with Transformer. In *Computer Vision – ECCV 2022*, pages 659–675. Springer Nature Switzerland, 2022.

[167] Gongjie Zhang, Shijian Lu, and Wei Zhang. CAD-Net: A context-aware detection network for objects in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 57(12):10015–10024, 2019.

[168] Hui Zhang, Kunfeng Wang, Yonglin Tian, Chao Gou, and Fei-Yue Wang. MFR-CNN: Incorporating multi-scale features and global information for traffic object detection. *IEEE Transactions on Vehicular Technology*, 67(9):8019–8030, 2018.

[169] Junyi Zhang, Junying Huang, Xuankun Chen, and Dongyu Zhang. How to fully exploit the abilities of aerial image detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[170] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z. Li. S3FD: Single Shot Scale-Invariant Face Detector. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 192–201, 2017.

[171] Tianyi Zhang, Chunyun Chen, Yun Liu, Xue Geng, Mohamed M. Sabry Aly, and Jie Lin. PSRR-MaxpoolNMS++: Fast Non-Maximum Suppression with Discretization and Pooling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–15, 2024. doi: 10.1109/TPAMI.2024.3485898.

[172] Xindi Zhang, Ebroul Izquierdo, and Krishna Chandramouli. Dense and small object detection in uav vision based on cascade network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[173] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021.

[174] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-object tracking by associating every detection box. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 1–21, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20047-2.

[175] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 589–597, 2016. doi: 10.1109/CVPR.2016.70.

[176] Zhewen Zhang, Fuliang Wu, Yuming Qiu, Jingdong Liang, and Shuiwang Li. Tracking small and fast moving objects: A benchmark. In Lei Wang, Juergen Gall, Tat-Jun Chin, Imari Sato, and Rama Chellappa, editors, *Computer Vision – ACCV 2022*, pages 552–569, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-26293-7.

[177] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12993–13000, 2020.

[178] Jingkai Zhou, Chi-Man Vong, Qiong Liu, and Zhenyu Wang. Scale adaptive image cropping for UAV object detection. *Neurocomputing*, 366:305–313, 2019.

[179] Qianyu Zhou, Xiangtai Li, Lu He, Yibo Yang, Guangliang Cheng, Yunhai Tong, Lizhuang Ma, and Dacheng Tao. TransVOD: End-to-End Video Object Detection With Spatial-Temporal Transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7853–7869, 2023. doi: 10.1109/TPAMI.2022.3223955.

[180] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[181] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[182] Chenchen Zhu, Ran Tao, Khoa Luu, and Marios Savvides. Seeing small faces from robust anchor's perspective. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5127–5136, 2018.

[183] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 366–382, 2018.

[184] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7380–7399, 2021.

[185] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 408–417, 2017.

[186] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2349–2358, 2017.

[187] Yabin Zhu, Chenglong Li, Yao Liu, Xiao Wang, Jin Tang, Bin Luo, and Zhixiang Huang. Tiny Object Tracking: A Large-scale Dataset and A Baseline, 2022.

[188] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2110–2118, 2016.

[189] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.

[190] Zhuofan Zong, Guanglu Song, and Yu Liu. DETRs with collaborative hybrid assignments training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6748–6758, 2023.