UNIVERSIDADE
BEIRA INTERIOR

Waste Container Detection System using Computer Vision

João Miguel Baltazar Martins

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor João Carlos Raposo Neves
Co-orientador: Prof. Doutor Hugo Pedro Martins Carriço Proença

junho 2025

Declaração de Integridade

Eu, João Miguel Baltazar Martins, que abaixo assino, estudante com o número de inscrição M13939 do Mestrado em Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o Código de Integridades da Universidade da Beira Interior.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 11/06/2025

Assinado por: **João Miguel Baltazar Martins**
Num. de Identificação: 15177664

Agradecimentos

Esta dissertação é o resultado de muitos contributos, e é com profunda gratidão que a dedico a todos os que, de alguma forma, me acompanharam nesta jornada.

Em primeiro lugar, agradeço à Universidade da Beira Interior por me ter proporcionado uma formação académica e científica de elevada qualidade, num ambiente que estimula o pensamento crítico, a curiosidade intelectual e o rigor na investigação, fatores determinantes para o crescimento pessoal e profissional que experienciei ao longo deste ciclo de estudos.

Aos meus orientadores, o Professor Doutor João Carlos Raposo Neves e o Professor Doutor Hugo Pedro Proença, expresso um agradecimento profundo pela orientação exigente, pela clareza nos conselhos, pela visão crítica e pela constante disponibilidade para acompanhar e melhorar cada etapa deste projeto.

Reconheço também a importância da empresa EVOX pelos dados que me foram disponibilizados, essenciais para a realização prática deste trabalho, sendo de destacar ainda a boa comunicação e colaboração com os responsáveis pelo projeto.

Aos meus colegas do SociaLab, Bernardo, Diogo e Sara, agradeço profundamente o companheirismo, o apoio técnico e emocional nos momentos mais desafiantes, e a constante partilha de conhecimento e motivação, que tornaram este percurso mais leve, colaborativo e enriquecedor.

Agradeço ainda à minha família e, em especial, à minha mãe, pela presença incansável, pelo apoio nos momentos de maior vulnerabilidade e por me ter transmitido, com o seu exemplo, a força e a capacidade de resiliência que me acompanharam nos dias mais difíceis.

Resumo

Esta dissertação propõe um sistema baseado em visão computacional para a deteção e contagem automática de contentores de resíduos urbanos, utilizando vídeos captados por veículos de recolha de lixo. A solução visa apoiar cidades inteligentes, permitindo o mapeamento geolocalizado de contentores e a otimização de rotas.
O sistema é suportado por um dataset dividido em duas fases: uma fase base e uma fase aumentada, na qual foram adicionados mais vídeos à fase inicial. A versão final (fase aumentada) contém 144 vídeos (cerca de 49 minutos), com aproximadamente 35,000 instâncias anotadas de 379 contentores únicos. Foram comparadas as performances de dois métodos de deteção: o YOLOv11 (baseado em imagem) e o DiffusionVID (baseado em vídeo), tanto na fase base como na fase aumentada do dataset. Em ambas as fases, o YOLOv11 apresentou melhor desempenho global, com destaque para a fase aumentada, na qual obteve um *mAP@0.5* de 0,938. Apesar da capacidade do DiffusionVID em detetar contentores de pequenas dimensões, o YOLOv11 demonstrou superioridade consistente.
Para a contagem de contentores, o *YOLOv11* foi integrado com o *ByteTrack* e melhorado com três heurísticas: (H1) filtragem de trajetórias curtas, (H2) fusão de identidades e (H3) consistência espacial. A combinação H1 + H2 + H3 resultou em melhorias significativas no erro médio absoluto (*MAE*, até -77%) e na soma das diferenças absolutas (*SAD*, até -77%), no conjunto expandido.
A robustez do sistema foi validada em vídeos reais (cerca de 2 horas cada), revelando que a eficácia das heurísticas varia de acordo com o vídeo. Ainda assim, a configuração H1 + H2 + H3 mostrou o melhor equilíbrio e generalização, sendo a recomendada para implementação prática.
Este trabalho contribui com um *dataset*, uma *pipeline* de deteção e seguimento de contentores, e heurísticas específicas para melhorar a fiabilidade do sistema. Como trabalho futuro, será avaliado o impacto do desequilíbrio entre classes no desempenho do sistema e investigada a escalabilidade das heurísticas em vídeos de maior duração.

Palavras-chave

Deteção de Objetos em Vídeo; Contagem de Contentores de Resíduos; Análise Temporo-Espacial; Visão Computacional em Ambientes Urbanos; Heurísticas de Pós-Processamento; Otimização da Recolha de Resíduos; Cidades Inteligentes.

Resumo alargado

Esta dissertação apresenta um sistema baseado em visão computacional para a deteção e contagem automática de contentores de resíduos urbanos, utilizando vídeos capturados por veículos operacionais de recolha de lixo. Projetado para apoiar infraestruturas urbanas sustentáveis e inteligentes, o sistema fornece uma base sólida para o mapeamento geolocalizado de contentores, a otimização de rotas e a redução de ineficiências operacionais.

A base da abordagem assenta num *dataset* construído em duas fases. A primeira fase consiste em 84 vídeos com um total de 33 minutos e 2 segundos, com 11 minutos e 26 segundos contendo contentores visíveis e 21 minutos e 36 segundos capturando cenas de fundo (sem contentores). A segunda fase expande o conjunto para 144 vídeos, com uma duração total de 48 minutos e 44 segundos, incluindo 16 minutos e 42 segundos de imagens de contentores e 32 minutos e 2 segundos de fundo. Esta evolução resultou em aproximadamente 35 000 instâncias anotadas de contentores e 379 contentores únicos, um aumento face aos 218 da fase inicial.

Para a deteção, foram avaliados dois modelos de referência no estado da arte: o *YOLOv11*, baseado em imagens, e o *DiffusionVID*, baseado em vídeo. Embora o *DiffusionVID* tenha demonstrado eficácia na deteção de contentores pequenos (como os contentores de pilhas), o *YOLOv11* alcançou um desempenho global superior, atingindo um *mAP@0.5* geral de 0,9384 após a ampliação do *dataset*. Esta ampliação foi realizada com o objetivo de avaliar se o método baseado em vídeo conseguiria superar o modelo baseado em imagens com um volume maior de dados. No entanto, os resultados sugerem que ou o aumento de dados ainda não foi suficiente para beneficiar plenamente o *DiffusionVID*, ou que o modelo baseado em imagens apresenta, de forma intrínseca, uma capacidade superior de generalização para este domínio específico.

Para suportar o seguimento multi-objeto, o *YOLOv11* foi integrado com o *ByteTrack* e posteriormente refinado através de um conjunto de heurísticas de pós-processamento. As heurísticas desenvolvidas — filtragem temporal de trajetórias de curta duração (H1), fusão de identidades (H2) e aplicação de consistência espacial (H3) — foram concebidas com o objetivo de aumentar a fiabilidade e precisão do seguimento.

Estas combinações foram inicialmente avaliadas no *dataset* anotado (base e aumentado), onde a configuração composta por H1 + H2 + H3 apresentou os melhores resultados globais. No *dataset* base, esta configuração reduziu o erro médio absoluto na contagem de contentores (*MAE*) de 2,00 para 0,50 e a soma das diferenças absolutas na contagem (*SAD*) de 16 para 4, o que representa uma melhoria de 75% em ambos os indicadores. Já no conjunto aumentado, o *MAE* caiu de 3,41 para 0,77 e o *SAD* de 75 para 17, com melhorias de 77,4% e 77,3%, respetivamente.

A robustez do sistema foi ainda avaliada em vídeos contínuos de longa duração com cerca de duas horas cada. Nestes videos, verificou-se que a combinação ótima de heurísticas variou: no primeiro, a combinação H1 + H2 foi a mais eficaz, reduzindo o número de contentores falsamente previstos de 469 para 24 (uma redução de 94,9%); no segundo, a heurística H2 isoladamente foi a mais eficiente, com uma redução de 361 para 48 (menos 86,7%). Estes

resultados mostram que não existe uma única configuração superior em todos os contextos. Ainda assim, a combinação completa H1 + H2 + H3 demonstrou o desempenho mais equilibrado e generalizável no *dataset* aumentado e manteve bons resultados nos vídeos que simulam as condições a que o sistema vai ser exposto. Por esse motivo, esta configuração é recomendada como a abordagem padrão para implementação prática do sistema.

Em conclusão, este trabalho contribui com um *dataset* anotado a nível urbano, uma *pipeline* de deteção e seguimento de contentores de resíduos, e heurísticas de pós-processamento específicas para melhorar a contagem dos contentores. A solução proposta é prática e apresenta potencial de integração em sistemas de cidades inteligentes.

Como trabalho futuro, será avaliado o impacto da quantidade de dados nas classes subrepresentadas através de subamostragem progressiva do conjunto de treino. No seguimento, serão analisados qualitativamente os casos de falha (*failure cases*) para refinar as heurísticas existentes. Além disso, pretende-se expandir o conjunto de vídeos contínuos, com rotas completas de recolha de resíduos, para avaliar a escalabilidade das heurísticas e obter uma visão mais realista do desempenho do *tracker*.

Abstract

We propose a computer vision system for the automatic detection and counting of urban waste containers in video streams captured by garbage collection vehicles. Designed to support smart city infrastructure, the system enables geolocated container mapping and route optimization.

Our approach is validated on a two-phase dataset comprising 144 videos (49 minutes) with over 35,000 annotated instances spanning 379 unique containers. We benchmark two detection models—YOLOv11 (image-based) and DiffusionVID (video-based)—across both dataset phases. YOLOv11 consistently outperforms DiffusionVID, particularly on the augmented dataset, achieving a mAP@0.5 of 0.938, despite the latter's strengths in detecting small-scale objects.

For counting, YOLOv11 is integrated with ByteTrack and enhanced using three domain-specific heuristics: (H1) short track filtering, (H2) identity merging, and (H3) spatial consistency. This configuration yields substantial improvements in accuracy, reducing the Mean Absolute Error (MAE) and Sum of Absolute Differences (SAD) by up to 77% on the augmented dataset.

System robustness is further validated on real-world deployment videos (2 hours each), demonstrating that the effectiveness of heuristics varies from video to video. Nonetheless, the H1+H2+H3 combination demonstrates the best generalization and is recommended for practical deployment.

Our contributions include: (i) a novel annotated dataset for urban waste container detection, (ii) a detection–tracking pipeline, and (iii) tailored heuristics for improving counting accuracy. Future work will address class imbalance, conduct failure case analysis, and evaluate scalability on continuous, long-duration video streams representing full waste collection routes.

Keywords

Video Object Detection; Waste Container Tracking; Spatio-Temporal Analysis; Computer Vision in Urban Environments; Post-Processing Heuristics; Waste Collection Optimization; Smart Cities.

Contents

1 Introduction **1**
1.1 Motivation and Objectives . . . 1
1.2 Expected Contributions . . . 2
1.3 Tasks and Timeline . . . 2
1.3.1 Task 1 - Literature Review (3 months) . . . 2
1.3.2 Task 2 - Dataset Construction (2 months) . . . 3
1.3.3 Task 3 - Detector Baseline Evaluation (1 month) . . . 3
1.3.4 Task 4 - Tracker Baseline Evaluation (1 month) . . . 3
1.3.5 Task 5 - Post-processing Heuristics for Object Tracking (2 months) . . . 3
1.3.6 Task 6 - Dataset Expansion and Re-evaluation (1 month) . . . 3
1.3.7 Task 7 - Heuristic Evaluation on Raw 2-Hour Vehicle Videos (1 month) 4
1.3.8 Task 8 - Dissertation Writing and Preparation of a Journal or Conference Paper (10 months) . . . 4
1.4 Document Organization . . . 4

2 State-of-the-Art **5**
2.1 Introduction . . . 5
2.2 Overview of State-of-the-Art One-Phase Image Object Detectors . . . 5
2.2.1 You Only Look Once (YOLO): You Only Look Once: Unified, Real-Time Object Detection [1] . . . 6
2.2.2 Detection Transformer (DETR): End-to-End Object Detection with Transformers [2] . . . 9
2.2.3 Single-Shot Detector (SSD): Single Shot MultiBox Detector [3] . . . 11
2.3 Overview of State-of-the-Art Two-Phase Image Object Detectors . . . 11
2.3.1 Fast Region-based Convolutional Network (Fast R-CNN) [4] . . . 12
2.3.2 Mask R-CNN [5] . . . 15
2.4 Overview of State-of-the-Art Video Object Detectors . . . 18
2.4.1 PTSEFormer: Progressive Temporal-Spatial Enhanced TransFormer Towards Video Object Detection [6] . . . 18
2.4.2 DiffusionVID: Denoising Object Boxes With Spatio–Temporal Conditioning for Video Object Detection [7] . . . 21
2.4.3 YOLOV: Making Still Image Object Detectors Great at Video Object Detection [8] . . . 24
2.4.4 YOLOV++: Practical Video Object Detection via Feature Selection and Aggregation [9] . . . 25
2.5 Overview of Multi-Object Tracking (MOT) . . . 26
2.5.1 Motion-Based Trackers . . . 27
2.5.2 Appearance-Based Trackers . . . 28
2.5.3 Tracker Selection Justification . . . 28

2.6 Conclusions 29

3 Proposed Dataset and Heuristic-Based Track Refinement for Waste Container Counting 31
3.1 Introduction 31
3.2 Dataset 31
3.3 Post-Processing Tracking Heuristics 33
3.3.1 Metrics 34
3.3.2 Motivation: Understanding Tracker Errors Through Object Scale Dynamics 35
3.3.3 Heuristic's 35
3.4 Conclusion 37

4 Experiments 39
4.1 Introduction 39
4.2 Base Dataset 39
4.2.1 Experiments with an Image-Based Detector [10] 41
4.2.2 Experiments with a Video-Based Detector [7] 44
4.2.3 Analysis of Image- vs. Video-Based Detection on the Base Dataset . . . 48
4.3 Dataset Augmentation 48
4.3.1 Dataset Growth Statistics 48
4.3.2 Impact on Detection Performance 51
4.3.3 Decision Based on Results 52
4.4 Tracker Heuristic's Evaluation 52
4.4.1 First Stage: Base Dataset 53
4.4.2 Second Stage: Dataset Augmented with Additional Videos 53
4.4.3 Heuristic Evaluation Using Raw 2-Hour Vehicle Camera Footage . . . 54
4.5 Summary and Conclusions 56

5 Conclusion and Future Work 59
5.1 Main Contributions 59
5.2 Future Work 60

Bibliography 61

List of Figures

1.1 Gantt diagram regarding the development of the tasks. 2

2.1 Output Bounding Box, Confidence, and Class Probability Map for YOLO [1]. . 7
2.2 Key Architectural Modules in YOLOv11 [10]. 8
2.3 DETR Architecture [2]. 9
2.4 Fast R-CNN vs Region-based Convolutional Neural Network (R-CNN) [11]. . . 13
2.5 Faster Region-based Convolutional Network (Faster R-CNN) [11]. 14
2.6 RoIAlign [5]. 16
2.7 Head Architecture [5]. 17
2.8 PTSEFormer Architecture [6]. 19
2.9 Overview of DiffusionVID [7]. 22
2.10 YOLOV [8] Architecture [8]. 24
2.11 YOLOV++ [9] Architecture. 25
2.12 ByteTrack [12] pipeline adapted from [13]. 27

3.1 Overview of the proposed waste container detection and tracking pipeline. The image-based detector [10] performs per-frame object detection, while the tracker algorithm [12] manages multi-object tracking across video frames. The resulting tracks are post-processed using three heuristics: (H1) minimum duration filtering, (H2) temporal merging based on detection gaps, and (H3) spatial proximity constraints. Final evaluation metrics (Mean Absolute Error (MAE), Sum of Absolute Differences (SAD)) are used to guide both tracker configuration and heuristic tuning. 34
3.2 Early frames show incorrect track ID switches, highlighted in yellow (each track ID is uniquely color-coded to improve distinction). 35
3.3 Heuristic H1 eliminate the initial track. 36
3.4 Area trajectory showing a track fragment caused by occlusion. The baseline tracker segments this into separate tracks (each track ID is uniquely color-coded to improve distinction). 36
3.5 Heuristic H1 remove the incorrect idswitch (12) and H2 merged the tracks (6, 14). 36

4.1 Precision-Recall Curve for YOLOv11 [10]. 41
4.2 YOLOV11 [10] Scenario 1. 42
4.3 YOLOV11 [10] Scenario 2. 42
4.4 YOLOV11 [10] Scenario 3. 42
4.5 YOLOV11 [10] Scenario 4. 43
4.6 YOLOV11 [10] Scenario 5. 43
4.7 YOLOV11 [10] Scenario 6. 43
4.8 YOLOV11 [10] Scenario 7. 44

4.9 YOLOV11 [10] Scenario 8. 44
4.10 Precision-Recall Curve for DiffusionVID [7]. 45
4.11 DiffusionVID [7] Scenario 1. 45
4.12 DiffusionVID [7] Scenario 2. 46
4.13 DiffusionVID [7] Scenario 3. 46
4.14 DiffusionVID [7] Scenario 4. 46
4.15 DiffusionVID [7] Scenario 5. 46
4.16 DiffusionVID [7] Scenario 6. 47
4.17 DiffusionVID [7] Scenario 1. 47
4.18 DiffusionVID [7] Scenario 2. 47
4.19 Dataset scaling: Comparison of total images, background/non-background images, and container instance counts. The augmentation significantly increased data volume for both overall training (left) and test evaluation (right). 49
4.20 Video content expansion: Increase in the number of unique videos containing containers and background-only scenes, reinforcing scenario diversity for both overall data (left) and the test split (right). 49
4.21 Temporal data augmentation: Growth in total video minutes for container-present and background-only footage (10 Frames Per Second (FPS)), contributing to valuable temporal context overall (left) and in the test split (right). . . . 50
4.22 Class distribution enhancement: Instance count growth per container class, illustrating efforts to mitigate imbalance for both the overall dataset (left) and the test split (right). This improves learning for rare classes. 50
4.23 Unique object track augmentation: Growth in the number of distinct tracked container instances. Per-class track increases (left) and overall unique object growth (right) support improved Re-ID learning and tracking evaluation. . . . 51
4.24 Precision-Recall curve for DiffusionVID [7] after training on the augmented dataset. 51
4.25 Precision-Recall curve for YOLOv11 [10] after training on the augmented dataset. 52
4.26 Impact of post-processing heuristics on SAD for Video 1 (Right). The combination H1 + H2 achieves the lowest SAD. 55
4.27 Impact of post-processing heuristics on SAD for Video 2 (Left). Here, H2 (identity merging) alone yields the lowest SAD, with H1 (temporal filtering) close behind. 56

List of Tables

3.1 Summary of Dataset Statistics by Video Split. 32
3.2 Summary of Dataset Statistics by Image Split. 32
3.3 Number of Videos per Split in which Each Class Appears. 33
3.4 Class Instances per Split and Total Across All Splits. 33

4.1 Summary of Video Statistics for Training, Validation, and Test Splits (Base Dataset). 40
4.2 Summary of Image Statistics for Training, Validation, and Test Splits (Base Dataset). 40
4.3 Class Instances per Split and Total Across All Splits (Base Dataset). 40
4.4 Number of Videos per Split in which Each Class Appears (Base Dataset). . . . 40
4.5 Overall evaluation across heuristic configurations on the first-stage dataset (base). Note: H2 and H3 are only applied in conjunction with H1. SAD is the total absolute error, while $Pred - GT$ is the signed difference, showing bias (positive = overestimation). 53
4.6 Overall evaluation across heuristic configurations on the second-stage dataset (augmented). Note: H2 and H3 are only applied in conjunction with H1. SAD is the total absolute error, while $Pred - GT$ is the signed difference, showing bias (positive = overestimation). 53
4.7 Overprediction analysis of the baseline tracker on Video 1. A significant surplus of container tracks is observed. 54
4.8 Overprediction analysis of the baseline tracker on Video 2. While inflated predictions remain, the overcount is slightly lower than in Video 1. 55

List of Acronyms

AP Average-Precision

C2PSA Convolutional block with Parallel Spatial Attention

CBS Convolution-BatchNorm-SiLU

CNN Convolutional Neural Network

CPU Central Processing Unit

CV Computer Vision

DCC Dynamic Coreset Conditioning

DETR Detection Transformer

FAM Feature Aggregation Module

Fast R-CNN Fast Region-based Convolutional Network

Faster R-CNN Faster Region-based Convolutional Network

FCOS Fully Convolutional One-Stage Detector

FFN Feed-Forward Network

FPN Feature Pyramid Network

FPS Frames Per Second

FSM Feature Selection Module

GIoU Generalized Intersection over Union

GPU Graphic Processing Unit

HOTA Higher Order Tracking Accuracy

IoU Intersection over Union

JPG Joint Photographic Expert Group

LBR Local Batch Refinement

MAE Mean Absolute Error

MADRL Multi-Agent Deep Reinforcement Learning

mAP Mean Average Precision

MOT Multi-Object Tracking

MOTA Multiple Object Tracking Accuracy

MOTP Multiple Object Tracking Precision

NMS Non-Maximum Suppression

OBB Oriented Object Detection

PAN Path Aggregation Network

PR Precision-Recall

QAM Query Assembling Module

R-CNN Region-based Convolutional Neural Network

RFID Radio-Frequency Identification

ROI Regions of Interest

RPN Region Proposal Network

SAD Sum of Absolute Differences

SORT Simple Online and Realtime Tracking

SPPF Spatial Pyramid Pooling - Fast

SSD Single-Shot Detector

STAM Spatial Transition Awareness Module

SVM Support Vector Machine

TBD Tracking by Detection

TFAM Temporal Feature Aggregation Module

UBI *Universidade da Beira Interior*

VOD Video Object Detection

YOLO You Only Look Once

Chapter 1

Introduction

The problem of waste container detection holds significant importance within the context of smart cities and sustainable urban environments. Effective waste management is a key element of urban sustainability, and optimizing waste collection processes can drastically reduce operational inefficiencies. Achieving these goals relies heavily on the availability of precise information regarding the placement and categories of waste containers. This information enables the design of optimized waste collection routes, leading to reduced fuel consumption, shorter collection times, and ultimately lower carbon emissions and operational costs. Moreover, it supports smart city initiatives by modernizing waste collection processes.

1.1 Motivation and Objectives

Urban waste collection is a critical component of sustainable city management. Optimizing these operations requires not only efficient route planning but also accurate, up-to-date information regarding the spatial distribution and status of waste containers. Traditional methods usually depend on static records or manual checks.

To address these challenges, this work aims to develop a computer vision system for the automatic detection and counting of urban waste containers in video streams captured by garbage collection vehicles. The proposed system uses video data from side-mounted cameras on collection vehicles to detect and track containers encountered during routine routes. Post-processing heuristics are applied to improve tracking accuracy and consistency, enabling the automatic generation of georeferenced maps of container locations.

The main challenge lies in achieving robust detection and tracking under diverse real-world conditions such as occlusions, varying lighting, and complex urban backgrounds. To explore solutions, we investigate both image-based and video-based detection methods. A detector is integrated with a MOT module based on ByteTrack [12] to enable persistent container identification over time. Additionally, heuristics are designed to refine temporal consistency and reduce identity switches.

To support this development, we created a dataset of 29,242 frames (approximately 48.7 minutes), including 10,022 annotated frames (around 16.4 minutes) with container instances. The dataset encompasses a variety of urban scenarios, including rare container types and challenging occlusions. We plan evaluations to assess the detection, tracking, and counting performance of urban waste containers across different models, dataset augmentations, and video lengths.

1.2 Expected Contributions

The outcomes of this dissertation are as follows:

- A dissertation titled *"Waste Container Detection System Using Computer Vision"*, presenting contributions to the field of Computer Vision (CV) with a focus on urban environments;
- The integration of post-processing heuristics into a state-of-the-art tracking pipeline to improve MOT and container counting performance in real-world scenarios;
- A journal or conference paper publishing the constructed dataset for waste container detection in urban environments, encompassing multiple container types and diverse urban scenes.

1.3 Tasks and Timeline

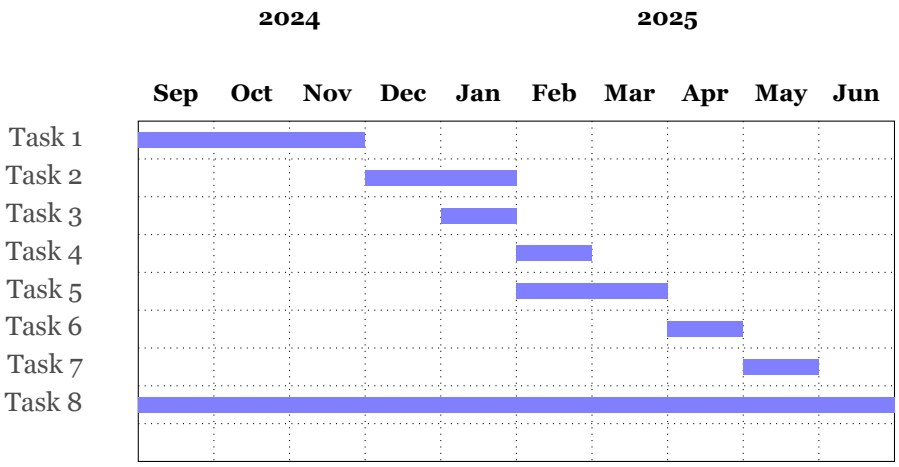

Figure 1.1: Gantt diagram regarding the development of the tasks.

To accomplish the objectives outlined in this dissertation, a series of planned tasks have been systematically organized and are illustrated in the Gantt chart presented in Figure 1.1.

1.3.1 Task 1 - Literature Review (3 months)

This task involves searching for existing waste container detection systems and datasets of waste containers. It includes gathering various types of image object detectors, including both one-stage and two-stage models, and analyzing their differences, strengths, and weaknesses to determine which approaches are most suitable for the problem presented in this report. In addition, it requires reviewing and studying recent methodologies in the field of Video Object Detection (VOD) and their application to the problem. The task concludes with an overview of state-of-the-art MOT algorithms, including motion-based approaches such as

SORT [14] and ByteTrack [12], as well as appearance-based approaches such as DeepSORT [15], and an analysis of their suitability for waste container tracking in urban scenarios.

1.3.2 Task 2 - Dataset Construction (2 months)

This task involves dividing the video data into segments, separating those with the presence of containers from those without (background), annotating the videos, and organizing the dataset into a structure suitable for image object detection and for VOD.

1.3.3 Task 3 - Detector Baseline Evaluation (1 month)

This task consists of applying an image-based detection method to the dataset to evaluate its performance. This method processes each image individually, without leveraging temporal information. In parallel, a video-based detection method is applied, which incorporates temporal context to improve detection accuracy. A comparative analysis of the results from both methods is performed to identify strengths, limitations, and baseline performance metrics.

1.3.4 Task 4 - Tracker Baseline Evaluation (1 month)

This task includes applying the ByteTrack [12] algorithm to track waste containers, as justified in Section 2.5.2.2. Tracking performance is evaluated using the MAE per video metric, where N is the total number of videos, GT_i is the ground-truth count, and Pred_i is the predicted count for video i. Based on the MAE and SAD results, hyperparameters are fine-tuned to establish the baseline.

1.3.5 Task 5 - Post-processing Heuristics for Object Tracking (2 months)

This task involves designing heuristics informed by patterns observed in the video data to address issues such as identity switches and fragmented tracks. These heuristics are implemented on the output files generated by the tracker. Each heuristic or combination of heuristics is tested to assess its effect on object count accuracy, and the results are evaluated by comparing the post-processed object counts for each class and each video with the ground truth annotations.

1.3.6 Task 6 - Dataset Expansion and Re-evaluation (1 month)

In this task, additional video data is annotated. Both YOLOv11 [10] and the video object detection method DiffusionVID [7] are evaluated on the expanded dataset, including retraining using the best hyperparameter settings for each. The model that achieves the best overall performance across all classes is selected. Post-processing heuristics are then applied to the tracking outputs of the selected method to minimize tracking errors and improve temporal consistency. The system's performance is evaluated using the previously defined metrics.

1.3.7 Task 7 - Heuristic Evaluation on Raw 2-Hour Vehicle Videos (1 month)

This task evaluates the selected heuristics by applying them to two raw, continuous videos of approximately 2 hours each, captured directly from garbage collection vehicle cameras. Tracking performance is assessed by comparing object counts before and after heuristic application to identify the most effective approach for improving tracking consistency under real-world conditions.

1.3.8 Task 8 - Dissertation Writing and Preparation of a Journal or Conference Paper (10 months)

This task involves the gradual development of the dissertation alongside prior tasks. It also includes preparing a journal or conference paper presenting the constructed dataset for urban waste container detection.

1.4 Document Organization

1. The first chapter – Introduction – introduces the problem of waste container detection, its importance for smart cities, and the objectives of this project dissertation report. It also highlights the main contributions and the relevance of the solution presented;

2. The second chapter – State of the Art – surveys the existing literature on object detection, VOD, and MOT. It reviews both one-stage and two-stage image detectors (e.g., YOLO [10], SSD [16], Fast R-CNN [4]), as well as VOD methods (e.g., PTSEFormer [6], DiffusionVID [7]). Regarding MOT, the chapter analyzes both motion-based and appearance-based approaches, including algorithms such as SORT [14], DeepSORT [15], ByteTrack [12], and BoT-SORT [17]. Based on this analysis, the algorithm that best aligns with the objectives and constraints of the proposed system is identified.

3. The third chapter – Proposed Dataset and Heuristic-Based Track Refinement for Waste Container Counting – presents the main contributions of this work: a dataset for waste container detection in urban scenes, and a set of heuristic-based post-processing techniques to improve multi-object tracking outputs. The chapter details the dataset structure and augmentation process, describes the detection and tracking pipeline, and introduces three heuristics designed to improve track consistency. These components work together to provide more accurate container counting over video sequences.

4. Chapter Four – Experiments – evaluates image [10] and video [7] object detection models on both base and augmented versions of the dataset. It analyzes the effects of augmentation and heuristic tracking refinements using quantitative metrics (e.g., Mean Average Precision (mAP), Precision-Recall (PR) curves) and qualitative visualizations. The chapter concludes with an assessment on long, real-world deployment videos.

5. The fifth chapter – Conclusion and Future Work – provides a summary of the research and its outcomes, as well as insights into future work.

Chapter 2

State-of-the-Art

2.1 Introduction

This chapter discusses image-based and video-based object detectors, focusing on one-phase and two-phase image detectors. It then reviews VOD methods, highlighting their use of temporal information and assessing their suitability for waste container detection. An overview of Multiple Object Tracking (MOT) is also provided, dividing trackers into motion-based and appearance-based categories with examples of each. Based on this analysis, the most suitable tracker for the task addressed in this work was selected.

2.2 Overview of State-of-the-Art One-Phase Image Object Detectors

Single-stage object detectors, also known as one-phase detectors, have gained significant attention due to their ability to directly predict object locations and classifications from input images without relying on a separate region proposal stage.
A characteristic of single-stage detectors is their unified architecture. These models integrate object localization and classification into a single network, enabling simultaneous optimization of both tasks.
Historically, many single-stage detectors have relied on anchor-based methodologies. Models such as YOLO (You Only Look Once) [1] and SSD (Single Shot MultiBox Detector) [3] utilize predefined anchor boxes, which vary in scale and aspect ratio, to detect objects of different sizes and shapes. During training, these anchors are matched to ground truth objects, serving as references for the model to learn object characteristics. However, this approach introduces challenges related to hyperparameter tuning and computational overhead, especially in scenarios with a large number of anchors.
To address these limitations, recent advancements have focused on anchor-free methods. Approaches like CornerNet [18] and Fully Convolutional One-Stage Detector (FCOS) (Fully Convolutional One-Stage Object Detection) [19] represent a shift in paradigm by eliminating the reliance on predefined anchors. Instead, these models predict keypoints (e.g., object corners) or centroids directly.
Another pivotal innovation in single-stage detectors is the use of Feature Pyramid Network (FPN). By merging low-level features, which capture fine-grained spatial details, with high-level semantic information, FPNs enhance the detector's ability to identify objects of varying scales.
The development of advanced loss functions has also played a crucial role in the success of single-stage detectors. For instance, the introduction of focal loss in RetinaNet [20] ad-

dresses the class imbalance problem by emphasizing hard-to-classify examples during training.
More recently, the integration of transformer architectures has brought a new perspective to single-stage detection. DETR (DEtection TRansformer) [2] exemplifies this shift by leveraging attention mechanisms to model global relationships within the image, thereby enhancing object localization accuracy. Additionally, DETR eliminates the need for traditional post-processing steps such as Non-Maximum Suppression (NMS).
In terms of applications, single-stage detectors excel in scenarios requiring rapid inference, such as autonomous vehicles, robotics, and surveillance systems. Although earlier models struggled with detecting small objects or performing in highly cluttered scenes, ongoing innovations in anchor-free designs and transformer-based approaches have significantly narrowed the performance gap between single-stage and two-stage detectors. These advancements demonstrate the potential of single-stage detectors to meet the demands of both efficiency and accuracy in increasingly complex visual tasks.

2.2.1 YOLO: You Only Look Once: Unified, Real-Time Object Detection [1]

YOLO [1], developed by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, transforms object detection into a unified regression task, predicting bounding boxes and their corresponding class probabilities for spatially distinct regions. By analyzing the entire image in a single forward pass, YOLO [1] effectively incorporates global contextual information to enhance its predictions.

2.2.1.1 YOLO [1] Algorithm

The YOLO [1] algorithm divides an input image into an $N \times N$ grid, where each grid cell is tasked with detecting an object if the object's center lies within its boundaries. Each grid cell concurrently handles both object localization and classification. For each cell, the model outputs B bounding boxes along with their corresponding confidence scores. The confidence score indicates the probability of an object being present in the cell. If no object is detected, the confidence score is set to zero; otherwise, it represents the Intersection over Union (IoU) between the predicted bounding box and the ground truth, providing a measure of the prediction's spatial accuracy. [21]
The following formula in equation 2.1 gives the value of confidence score.

$$\text{Confidence Score} = \Pr(\text{Object}) \times \text{IOU}(\text{truth pred}), \quad (2.1)$$

where:

- Confidence Score is the combined measure of how confident the model is that an object is present and how well the predicted bounding box overlaps with the actual object;
- Pr(Object) is the likelihood (a value between 0 and 1) that the object is present;

- IoU is the ratio between the area of overlap and the area of union of the predicted and actual bounding boxes, also a value between 0 and 1.

2.2.1.2 YOLO [1] Bouding Boxes

Bounding boxes in YOLO [1] are assigned confidence scores to indicate the likelihood of containing objects, and NMS is applied to filter out redundant boxes with low probabilities. While YOLO [1] excels in speed and efficiency, its localization accuracy, especially for smaller objects, tends to lag behind two-stage detectors. To address these challenges, later versions of YOLO [1] introduced significant improvements. These include advanced backbone architectures such as DarkNet19 [22], DarkNet53 [23], and CSPDarkNet53 [24], along with innovative modules like Spatial Pyramid Pooling - Fast (SPPF) and Path Aggregation Network (PAN). Additionally, enhancements in image preprocessing and loss calculation have further refined the model's accuracy and robustness.

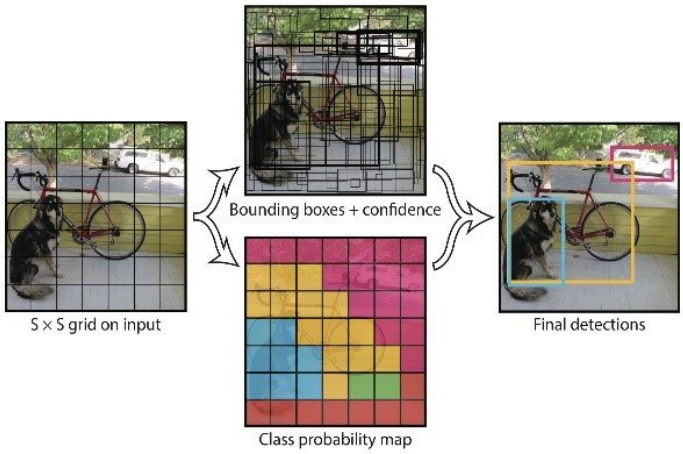

Figure 2.1: Output Bounding Box, Confidence, and Class Probability Map for YOLO [1].

As depicted in Figure 2.1, the YOLO [1] system formulates object detection as a regression task. The image is partitioned into an $S \times S$ grid, where each grid cell predicts B bounding boxes, their associated confidence scores, and C class probabilities. These outputs are represented as a tensor of dimensions $S \times S \times (B \times 5 + C)$.

2.2.1.3 YOLOv11: An Overview of the Key Architectural Enhancements [10]

YOLOv11 [10], the latest advancement in the YOLO [1] series of object detection algorithms, introduces transformative enhancements in architecture and functionality. This model incorporates innovative modules, such as the C3k2 block, SPPF, and Convolutional block with Parallel Spatial Attention (C2PSA), enabling superior feature extraction and real-time performance.

The Figure 2.2 presents an overview of the YOLOv11 [10] architecture, highlighting its components aimed at enhancing object detection capabilities. The architecture is organized into three main parts:

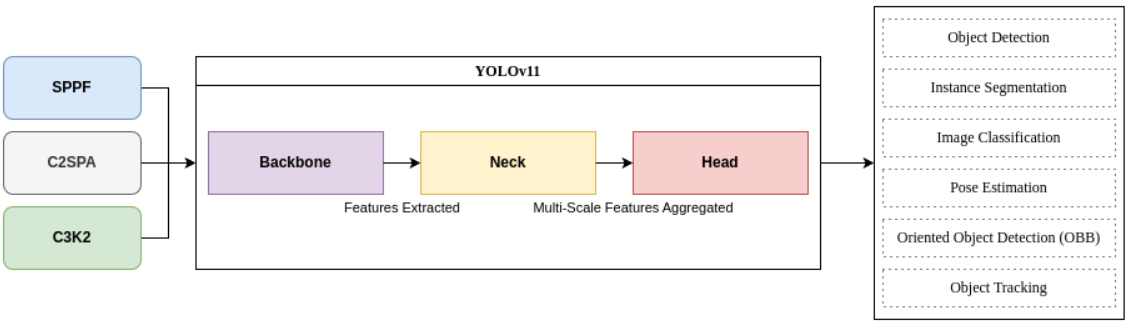

Figure 2.2: Key Architectural Modules in YOLOv11 [10].

- **Backbone:** The backbone serves as the model's feature extractor. YOLOv11 [10] introduces:
  - **C3k2 Block:** A lightweight, efficient variant of the CSP bottleneck, utilizing smaller kernels for faster computation;
  - **C2PSA Module:** A spatial attention mechanism that focuses on critical image regions, improving detection of occluded or small objects.
- **Neck:** The neck aggregates multi-scale features for prediction:
  - Enhanced with the C3k2 block, ensuring faster and more efficient feature fusion;
  - Integration of spatial attention mechanisms enhances focus on objects of interest.
- **Head:** The head generates final predictions:
  - Uses multiple C3k2 blocks for refined multi-scale feature processing;
  - Includes Convolution-BatchNorm-SiLU (CBS) blocks for stable feature refinement;
  - Outputs bounding boxes, objectness scores, and class probabilities through optimized detection layers.

In conclusion, YOLOv11 [10] introduces several improvements in object detection. The model includes new components like the C3k2 block, SPPF, and C2PSA, to enhance feature extraction and processing, improving detection accuracy in complex scenarios. YOLOv11 [10] can also perform multiple tasks beyond object detection, including instance segmentation, image classification, pose estimation, and oriented object detection.
The C2PSA component improves the model's attention to important areas in an image, helping it detect occluded or complex objects. Overall, YOLO11 balances speed, scalability, and versatility, making it suitable for a wide range of computer vision tasks.

2.2.1.4 Suitability of YOLO [1] for Waste Container Detection Tasks

YOLO [1] is a highly efficient and unified framework for object detection that processes an image in a single forward pass. Its exceptional inference speed makes it an excellent candidate for real-time waste container detection from collection vehicles. By analyzing the entire

image context simultaneously, YOLO effectively identifies objects in relatively simple scenes. Furthermore, its iterative improvements, such as those seen in YOLOv11 [10], introduce advanced features like C2PSA and SPPF, which enhance its accuracy and robustness in handling moderately challenging scenarios.
However, YOLOv11 [10] has notable limitations that restrict its application in complex environments. It struggles with detecting small objects and maintaining precision in scenarios involving occlusion, which are common in urban waste management contexts. Its reliance on anchor-based mechanisms often results in underperformance when handling densely cluttered environments or objects with varying scales. Additionally, YOLO's localization capabilities are less effective in scenarios of overlapping or partially visible containers.

2.2.2 DETR: End-to-End Object Detection with Transformers [2]

DETR [2] uses a special loss function with matching to make sure each prediction matches one ground truth object. With a transformer architecture, it uses a fixed number of object queries to understand relationships between objects and the whole image. This lets it make all predictions at once.

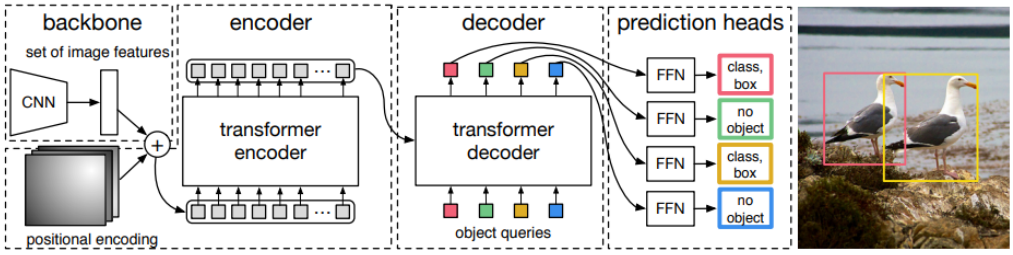

Figure 2.3: DETR Architecture [2].

At a high level, as illustrated in Figure 2.3, this approach uses Convolutional Neural Network (CNN) followed by a transformer to detect objects, employing a bipartite matching strategy during training. This is the primary reason why it is so simple. Breaking it down into two steps:

- An image is processed through an CNN encoder because these types of networks are particularly effective at extracting features from images. After passing through the CNN, the essential image features are retained, resulting in a higher-order representation with a rich set of feature channels;

- The feature map is fed into a transformer encoder-decoder to predict bounding boxes, each with a class label. A challenge arises from the absence of a "nothing" class in annotations and from handling similar objects close to each other. DETR [2] addresses this using a bipartite matching loss, which matches predicted boxes to ground truth annotations, including a "none" class. The total number of predicted boxes is padded with "none" boxes to match the total number of annotations. The Hungarian algorithm

is used to find the optimal one-to-one matching by minimizing a cost matrix, ensuring that both object presence and absence are properly handled.

The main components of DETR [2] are:

- The backbone: Features extracted from a CNN, along with positional encodings, are passed to the next stage;
- The transformer encoder: The transformer, being a sequence processing unit, requires the input tensors to be flattened. It processes the sequence and outputs an equally long sequence of features;
- The transformer decoder: Object queries are provided as input to the decoder, allowing it to condition its output based on these queries;
- Prediction Feed-Forward Network (FFN): The output from the transformer is passed through a feed-forward network, which consists of a classifier that predicts the class labels and bounding boxes, as discussed earlier.

2.2.2.1 Evaluation of DETR [2]

From the results exposed in [2], DETR's architecture is highly flexible, making it easily adaptable to panoptic segmentation tasks, which combine both object detection and semantic segmentation, all while maintaining competitive results. A key strength of DETR is its superior performance on large object detection, owing to its ability to leverage global context via the self-attention mechanism. However, the architecture also presents several challenges, particularly in terms of training efficiency, optimization, and performance with small objects. While similar issues have taken years to resolve in other detection models, there is optimism that future research will address these limitations effectively in the context of DETR [2].

2.2.2.2 Suitability of DETR [2] for Waste Container Detection Tasks

The DETR [2] redefines object detection by treating it as a set prediction problem. Its transformer-based architecture models global relationships within a scene, allowing it to detect objects accurately even in densely packed or occluded environments. Unlike traditional methods, DETR [2] eliminates the need for heuristic post-processing steps, such as NMS, simplifying the detection pipeline. Additionally, its attention mechanisms enhance robustness across varying object scales, making it suitable for detecting containers of different sizes and perspectives.
However, DETR [2] strengths come with significant limitations. Its high computational overhead and slow convergence during training pose challenges for deployment in real-time applications, particularly on embedded systems used in waste collection vehicles. Moreover, DETR [2] reliance on global attention mechanisms leads to underperformance in detecting small objects, making it less reliable in scenarios where containers are partially visible or located at a distance.

2.2.3 SSD: Single Shot MultiBox Detector [3]

SSD [3] operates on a feed-forward convolutional network that generates a fixed-size set of bounding boxes, along with confidence scores indicating the likelihood of object class instances within those boxes. This is followed by a NMS step to refine the final detections.

2.2.3.1 Comparison of SSD [3] with Region-Based Methods

Although Fast R-CNN [4] and Faster R-CNN [4] are highly accurate, these methods are computationally demanding and unsuitable for embedded systems. Even when deployed on high-end hardware, they remain too slow for real-time applications. As highlighted in SSD [3] paper, with an input size of 300×300, significantly surpasses the 448×448 YOLO model in both accuracy and speed.

The SSD [3] framework simplifies object detection compared to methods that rely on object proposals, such as Faster R-CNN [25]. It does this by eliminating the need for separate proposal generation stages and the complex pixel or feature resampling that typically follows. Instead, SSD [3] integrates all these tasks into a single neural network, making it easier to train and faster to use.

2.2.3.2 Suitability of SSD [3] for Waste Container Detection Tasks

The SSD [3] achieves a balance between speed and accuracy by employing multi-scale feature pyramids, enabling the detection of objects across diverse sizes. This characteristic is particularly valuable in waste container detection, where containers can vary significantly in size. SSD [3] simplified training process and end-to-end architecture make it accessible for rapid deployment and easier fine-tuning to specific tasks.

Nonetheless, SSD [3] exhibits several weaknesses that limit its utility in complex waste collection scenarios. The use of predefined anchor boxes constrains its ability to handle densely cluttered scenes or objects that are partially obscured by nearby infrastructure or vehicles. While SSD [3] offers higher speeds compared to two-stage detectors, it sacrifices precision, particularly when detecting small or distant objects. These trade-offs can impair its effectiveness in detecting waste containers in challenging, real-world environments.

2.3 Overview of State-of-the-Art Two-Phase Image Object Detectors

Two-phase image object detectors employ a two-step approach, consisting of region proposal generation followed by refined classification and localization. The introduction of Region Proposal Network (RPN) in Faster R-CNN [25] revolutionized this pipeline by enabling the generation of high-quality region proposals. These proposals are created by predicting objectness scores and bounding box adjustments, which significantly enhance detection efficiency. Feature extraction plays a crucial role in these detectors, with deep feature extractors, used to capture rich semantic information. These backbone networks are often augmented with FPN to improve multi-scale detection.

Once region proposals are obtained, techniques like Regions of Interest (ROI) Pooling or ROI Align are used to extract and resize regions to a fixed size. Notably, ROI Align, introduced in Mask R-CNN [5], addresses quantization errors, leading to more precise localization and segmentation. Cascade R-CNN extends this two-phase paradigm by employing a multi-stage refinement process for both classification and localization. This iterative approach mitigates the risk of overfitting to suboptimal proposals, thereby boosting overall performance. In addition, attention mechanisms, such as those found in Dynamic R-CNN and transformer-based models, enhance the model's ability to focus on relevant features, especially in complex or cluttered scenes. Modern detectors also integrate contextual reasoning modules to leverage surrounding information, which further improves accuracy in dense or occluded scenarios.

These two-phase detectors consistently achieve state-of-the-art performance on benchmarks like COCO [26] and PASCAL VOC [27], making them highly suitable for applications that demand high reliability, such as medical imaging, satellite image analysis, and high-resolution video analytics. However, despite their superior accuracy, their higher computational cost and inference latency often limit their applicability in resource-constrained or real-time settings.

2.3.1 Fast R-CNN [4]

Fast R-CNN [4] is among the most widely used object detection architectures, exploiting a CNN, similar to models like YOLO [1]. Introduced in 2015 by Ross Girshick, Kaiming He, Jian Sun, and Shaoqing Ren, Fast R-CNN achieves a balance between the accuracy of deeper models and significantly improved speed. It builds upon the principles of R-CNN, an earlier model that utilized high-capacity CNNs to process region proposals for object localization and segmentation.

In Fast R-CNN [4], the input image is processed by a CNN to produce a convolutional feature map. Instead of directly feeding region proposals into the network, they are first identified and transformed into fixed-size squares using a ROI pooling layer. These proposals are then reshaped into a consistent size and passed through fully connected layers. A Softmax layer is subsequently applied to the ROI feature vectors to classify the proposed regions and compute bounding box offsets for more precise localization. [28]

2.3.1.1 Fast R-CNN [4] vs R-CNN

Fast R-CNN [4] has certain limitations, particularly in its reliance on region proposals generated by the selective search algorithm. Since selective search operates exclusively on the Central Processing Unit (CPU), it introduces a considerable computational bottleneck, leading to significant delays during this stage.

As illustrated in Figure 2.4, Fast R-CNN [4] is faster than R-CNN because it eliminates the need to pass a large number of region proposals through a CNN each time. Instead, the convolution operation is performed only once per image, after which a feature map is generated. This improvement is achieved by introducing a ROI pooling layer, which maps the feature

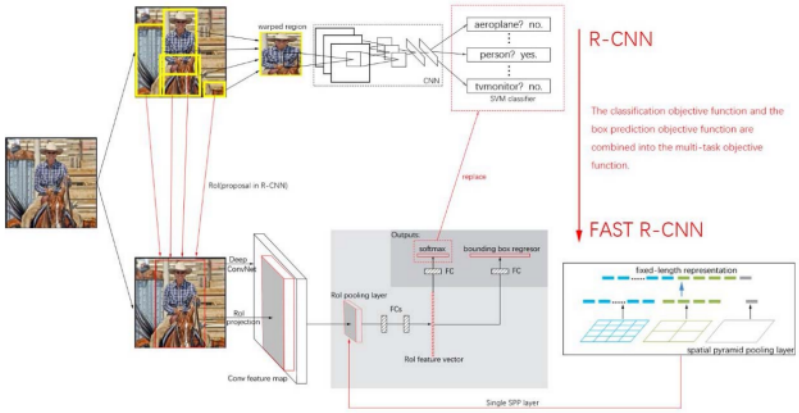

Figure 2.4: Fast R-CNN vs R-CNN [11].

map regions corresponding to each ROI into a fixed-size feature vector. This eliminates the redundancy of running the convolutional network multiple times for overlapping regions, significantly reducing computational overhead.
Concluding, instead of processing each ROI separately, Fast R-CNN processes the entire image in a single pass through the network. It shares the computation and memory for all ROIs from the same image during both the forward and backward passes.

2.3.1.2 Suitability of Fast R-CNN [4] for Waste Container Detection Tasks

Fast R-CNN [4] improves upon its predecessor, R-CNN, by introducing a shared convolutional feature map for all ROI, significantly enhancing computational efficiency. It performs classification and bounding box regression in a single stage after ROI pooling, achieving strong localization and classification accuracy. Its ability to process the entire image once and share computations across ROI makes it relatively efficient compared to earlier two-phase methods.
However, Fast R-CNN [4] relies on external region proposal methods like Selective Search, which operate on the CPU, creating a bottleneck that hampers real-time performance. This reliance also limits its adaptability to scenarios with rapid environmental changes, such as moving vehicles or dynamic lighting, which are common in waste collection tasks. The method may struggle in highly cluttered urban scenes where region proposals fail to capture smaller or partially occluded containers.

2.3.1.3 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [25]

To address the issue of Fast R-CNN [4] time-consuming candidate frame generation, Ross B. Girshick introduced Faster R-CNN [25] in 2016. The key contribution is its integration of candidate region generation and feature extraction into a unified deep network framework, which eliminates redundant operations. This entire process is handled on the Graphic Processing Unit (GPU), significantly improving processing speed.
The Figure 2.5 illustrates the four most important parts of this innovation:

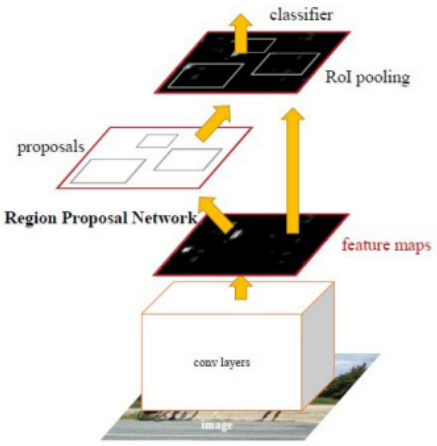

Figure 2.5: Faster R-CNN [11].

1. Convolutional Layers: Extracts image feature maps using a combination of convolution, ReLU, and pooling layers. The feature map is shared by subsequent layers;

2. RPN: Generates region proposals using a fully convolutional network, replacing Selective Search. Softmax and bounding box regression are used for more accurate anchor points;

3. ROI Pooling: Collects feature maps and proposals, then extracts proposal feature maps, sending them to the fully connected layer for classification;

4. Classification: Determines the object category and refines the bounding box for precise detection.

The most important structure of Faster R-CNN [25] is RPN. This layer uses Softmax to determine whether the anchor value is positive or negative, and then uses Bounding Box Regression [29] to get accurate recommendations [11].
In summary, Faster R-CNN [25] addresses the limitations of Fast R-CNN [4] by optimizing the detection pipeline through the integration of region proposal generation and feature extraction within a single deep network. By leveraging the RPN for efficient and accurate region proposals, it eliminates redundant computations and improves both speed and precision. This approach has established Faster R-CNN [25] as a highly influential model in the field of object detection.

2.3.1.4 Suitability of Faster R-CNN [25] for Waste Container Detection Tasks

Faster R-CNN [25] addresses the inefficiency of external region proposal generation by integrating a RPN into its pipeline. The RPN shares features with the detection network, enabling end-to-end training and significantly improving both speed and accuracy. This innovation allows Faster R-CNN [25] to handle densely cluttered scenes and varying scales, making it highly effective for urban waste container detection. Its robust backbone architecture, such as ResNet, further enhance its ability to localize objects with high precision.

Despite its strengths, Faster R-CNN [25] is still computationally heavy because it uses a two-stage process. This makes it less suitable for real-time applications, especially on resource-constrained devices like the Jetson Nano used in waste collection vehicles. Additionally, like Fast R-CNN [4], its accuracy drops when detecting small or heavily hidden objects.

2.3.2 Mask R-CNN [5]

Mask R-CNN [5] is an enhancement of Faster R-CNN [25] that extends its functionality beyond object detection to also include instance segmentation.
The key differences that Mask R-CNN [5] brings compared to Faster R-CNN [25] are:

1. ROIAlign: Mask R-CNN [5] replaces ROIPool with ROIAlign to address the issue of misalignments between feature maps and the ROI grids. Unlike ROIPool, which rounds coordinates and introduces approximation errors, ROIAlign uses bilinear interpolation to preserve spatial alignment, leading to more accurate feature extraction;

2. FPN: The model incorporates an FPN to enhance its ability to detect objects of varying scales. This creates a multi-scale feature representation by combining high-resolution and low-resolution features, enabling better detection of both small and large objects while reusing features efficiently;

3. Mask Head: Mask R-CNN [5] adds a new branch, called the mask head, to the Faster R-CNN [25] framework. This branch is a small CNN that predicts a binary mask for each detected object, enabling pixel-level segmentation.

While Faster R-CNN [25] focuses solely on object detection and bounding box localization, Mask R-CNN [5] extends this functionality by providing high-quality instance-level segmentation. This capability is critical for applications such as medical imaging, autonomous driving, and video object segmentation, where precise delineation of object boundaries is essential.

2.3.2.1 Mask R-CNN [5] Architecture

Mask R-CNN [5] extends the Faster R-CNN [25] framework to include instance segmentation, enabling precise object boundaries at the pixel level.

- Backbone Network with FPN
  - The backbone uses pre-trained networks like ResNet or ResNeXt to extract features;
  - To handle the challenge of objects appearing at different scales, Mask R-CNN [5] includes a FPN.
    * Feature Fusion: The FPN integrates high-level features, which provide abstract, semantic information such as object categories and global context, with low-level features, which capture fine-grained spatial details like edges and

textures. This fusion creates a multi-resolution feature pyramid, enhancing the model's ability to detect objects at various scales;

  - Multi-Scale Representation: The pyramid covers different resolutions, making it easier to detect objects of various sizes. This approach helps both object detection and accurate mask generation for segmentation.

- The use of FPN ensures that both local and global context are captured effectively.

- RPN

  - The RPN, working with the FPN-generated feature maps, proposes potential regions of interest (ROIs). It outputs bounding boxes with high object confidence, which are refined in the next steps.

- ROIAlign

  - One of Mask R-CNN's key innovations is replacing ROI pooling with ROIAlign, shown in Figure 2.6. This solves the problem of spatial misalignment caused by quantization in ROI pooling:

    - Grid Alignment: ROIs are divided into a fixed grid, and features are extracted using bilinear interpolation, rather than using coarse quantization;
    - Precision: This ensures that features align properly with the ROIs, maintaining fine spatial details essential for accurate segmentation.

  - ROIAlign is especially important for accurate mask predictions, particularly for small objects or those with complex shapes.

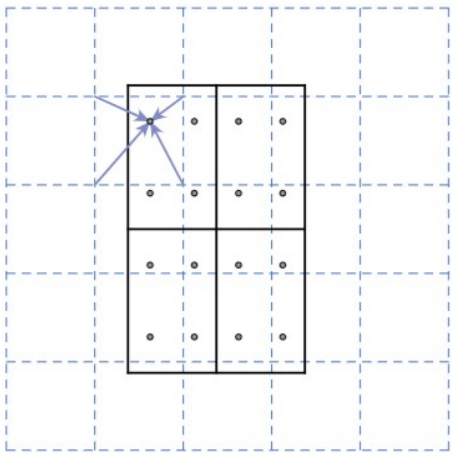

Figure 2.6: RoIAlign [5].

- Mask Head

  - The mask head, shown in Figure 2.7, is a fully convolutional branch that predicts a binary segmentation mask for each ROI. It works alongside the classification and bounding box regression branches, maintaining the model's modular structure. The mask head predicts one mask per object class, independent of class-specific probabilities.

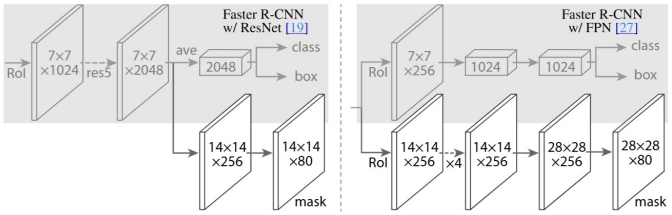

Figure 2.7: Head Architecture [5].

In summary, Mask R-CNN [5] integrates FPN for multi-scale feature extraction, ROIAlign for accurate spatial alignment, and the mask head for instance segmentation.

2.3.2.2 Mask R-CNN [5] Limitations

Mask R-CNN [5] includes several limitations:

- Computational Complexity
  - Mask R-CNN [5] can be computationally heavy, requiring significant resources, especially for high-resolution images or large datasets. Training and inference can be slow.
- Small-Object Segmentation
  - The model has difficulty segmenting very small objects. Due to the limited pixel information and the resolution of feature maps, Mask R-CNN [5] struggles with fine-grained details, leading to poor performance on small objects or low-resolution images.
- Data Requirements
  - Training Mask R-CNN [5] effectively requires a large amount of annotated data.
- Limited Generalization to Unseen Categories
  - Mask R-CNN [5] has trouble generalizing to new object categories that weren't part of the training data. The model needs additional labeled data for each new class.

2.3.2.3 Suitability of Mask R-CNN [5] for Waste Container Detection Tasks

Mask R-CNN [5] extends Faster R-CNN [25] by adding a branch for pixel-level instance segmentation, enabling it to detect waste containers and delineate their boundaries precisely. This capability is particularly valuable in scenarios where containers are partially visible or closely packed. The introduction of ROIAlign instead of ROI Pooling addresses spatial misalignments, ensuring higher accuracy in both detection and segmentation tasks. Mask R-CNN's incorporation of FPN further enhances its ability to detect containers of varying sizes and perspectives.

However, Mask R-CNN [5] added complexity increases its computational demands significantly, making it the slowest among the two-phase detectors discussed. While its segmentation capabilities are a notable advantage, they may not be essential for all waste container detection applications, where bounding box predictions often suffice. Additionally, its performance on small objects or in low-resolution images still face challenges due to feature resolution limitations.

2.4 Overview of State-of-the-Art Video Object Detectors

Video Object Detection (VOD) is a critical area in computer vision that focuses on detecting objects across video sequences while addressing challenges such as motion blur, occlusion, and camera defocus, which are particularly common in urban environments due to factors like occlusions from vehicles or pedestrians and varying lighting conditions. By leveraging temporal information across frames, VOD methods enhance detection accuracy and robustness, effectively mitigating ambiguities found in individual frames. Recent advancements have enabled VOD to narrow the performance gap with image-based detectors, demonstrating comparable accuracy while offering superior handling of dynamic and complex scenarios, making it increasingly viable for real-world applications.

Post-processing techniques, such as Seq-NMS [30], [31], and BLR [32], refine detections by combining bounding box predictions across frames to improve recall and tracking consistency. Feature enhancement methods, including optical flow-based (e.g., Deep Feature Flow, FGFA), attention-based (e.g., SESLA [33], RDN [34]), and tracking-based approaches, align and aggregate features to strengthen keyframe representations. While effective, these methods often incur high computational costs.

Emerging approaches like TransVOD [35] utilize spatial-temporal Transformers to better capture dependencies across frames, offering improved robustness. However, trade-offs between accuracy and efficiency persist, as seen in methods like EOVOD [36], which prioritize speed at the expense of performance. Balancing these aspects remains a critical focus in advancing VOD.

While these advancements have significantly improved VOD performance, challenges remain in balancing detection accuracy, computational efficiency, and real-time applicability. The following subsections will provide an in-depth review of VOD methods.

2.4.1 PTSEFormer: Progressive Temporal-Spatial Enhanced TransFormer Towards Video Object Detection [6]

PTSEFormer [6] is the latest deformable DETR-based [2] method, which is based on deformable attention modules that simultaneously learn the pixel location for reference and the attention weight.

While incorporating standard components such as the Temporal Feature Aggregation Module (TFAM) for leveraging temporal information and the Spatial Transition Awareness Mod-

ule (STAM) for capturing spatial transitions between frames, PTSEFormer [6] distinguishes itself with two novel contributions:

- Gated Correlation Model – Introduced to resolve imbalances in the Transformer decoder caused by residual connections through a gating mechanism;
- Query Assembling Module (QAM) – Dynamically generates object queries from context frames, enabling more accurate position inference without relying on fixed parameters from training data.

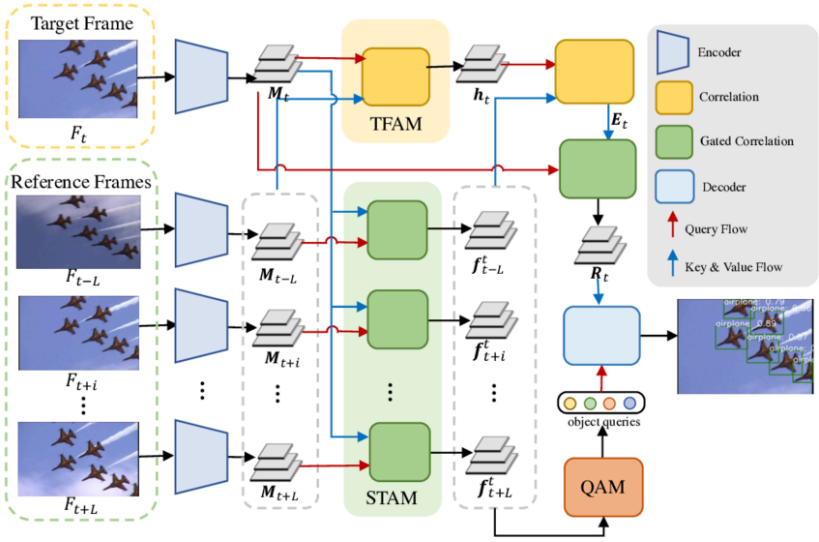

Figure 2.8: PTSEFormer Architecture [6].

A high-level overview of the proposed PTSEFormer [6] architecture is presented in Figure 2.8. Given a target frame F_t and its surrounding context frames $F_t^c = \{F_{t+i}\}_{i=-L:L}$, PTSEFormer [6] aims to detect the class and bounding boxes of objects within the target frame F_t. Initially, image features are extracted, followed by the application of two modules: the TFAM, which captures the temporal dynamics of objects, and the STAM, which models the spatial transitions of objects across frames. These temporal and spatial features are progressively aggregated to build a comprehensive representation. The resulting aggregated feature, along with object queries generated by the QAM, is then processed by a decoder to infer the final detection results. Notably, the object queries are conditioned on the context frames, allowing for more precise object localization and improved overall detection accuracy.

2.4.1.1 Temporal and Spatial Encoding

This section introduces an innovative approach to extract temporal and spatial memories from the target frame F_t and its context frames $F_t^c = \{F_{t+i}\}_{i=-L:L}$. Using a transformer-based encoder, latent feature maps are generated for both the target and context frames, denoted as M_t and M_t^c, respectively. Temporal and spatial memories are then extracted using two key modules:

- TFAM extracts temporal memory h_t by capturing motion information between the target and context frames, leveraging a correlation operator;

- STAM designed to capture the positional transitions of objects between frames. It employs a novel Gated Correlation operation to address the imbalance often introduced by the standard correlation operator.

The integration of both temporal and spatial context significantly enhances the model's ability to detect objects across frames, especially in complex video sequences where motion and positional shifts occur. A key innovation in PTSEFormer [6] is the Gated Correlation operation, which improves the attention mechanism by mitigating the imbalance often seen in traditional attention models. This enhancement proves critical for handling dynamic object movements and occlusions, leading to more accurate object detection.

2.4.1.2 Enhanced Memory Decoding

The original DETR [2] uses fixed object queries, limiting the integration of context frames. PTSEFormer [6] addresses this with a QAM which uses a transformer decoder to propagate position distribution through object queries over time. The final object queries are formed by concatenating the primal queries with those derived from context frames via a shallow decoder.
The QAM enhances PTSEFormer [6] by dynamically adapting object queries across frames, improving detection accuracy over methods with fixed queries. The shallow decoder balances efficiency with capturing dynamic object motion.

2.4.1.3 Learning PTSEFormer [6]

PTSEFormer [6] employs the Hungarian algorithm to match ground truth with predictions. The model optimizes a combined loss function that includes:

- Focal loss for classification;
- L1 loss and Generalized Intersection over Union (Generalized Intersection over Union (GIoU)) loss for bounding box regression.

The hyperparameters λ_{cls}, λ_{box}, λ_{L1}, and λ_{giou} adjust the contribution of each loss type to the overall loss. This approach is consistent with standard DETR-based models and is tailored for classification and bounding box regression tasks.

2.4.1.4 Network Details

PTSEFormer [6], built upon the DETR [2] framework, introduces key modifications to enhance efficiency and performance. By reducing the encoder and decoder to two layers, the model achieves a balance between computational speed and precision, suitable for real-time applications, though its impact on complex scenarios requires evaluation. Multi-scale features improve small object detection, while a ResNet-101 backbone and two-layer structures in TFAM, STAM, and Gated Correlation optimize the architecture. With six attention heads and 100 object queries, PTSEFormer [6] effectively balances computational efficiency and detection performance, making it well-suited for diverse video object detection tasks.

2.4.1.5 Suitability of PTSEFormer [6] for Waste Container Detection Tasks

PTSEFormer [6] is a deformable DETR-based framework that excels in modeling spatial and temporal relationships. The integration of the TFAM and STAM enhances its ability to capture object dynamics and transitions across frames. Additionally, the QAM dynamically generates object queries, improving localization and robustness in detecting waste containers despite occlusions and motion blur.
While PTSEFormer [6] demonstrates remarkable accuracy in complex and cluttered video sequences make it unsuitable for real-time deployment, such as on waste collection vehicles. Moreover, the reliance on computationally intensive attention mechanisms limits its scalability for larger datasets or scenarios requiring rapid processing.

2.4.2 DiffusionVID: Denoising Object Boxes With Spatio–Temporal Conditioning for Video Object Detection [7]

DiffusionVID [7] is a method that utilizes a diffusion model to leverage spatio-temporal information. It refines randomly generated noise boxes to accurately recover the original object boxes in a video sequence.
Comparing this method with PTSEFormer [6] mentioned in 2.4, PTSEFormer [6] achieves higher accuracy but at the cost of slower inference (314.0 ms per frame). In contrast, DiffusionVID [7] offers a better speed-accuracy trade-off, running 14.6 times faster (21.5 ms per frame) while maintaining competitive performance and providing flexibility for further refinement.
To effectively refine the object boxes in degraded images from the videos, the method employed three novel approaches: cascade refinement, dynamic coreset conditioning, and local batch refinement.
The process involves using a video sequence as input, represented as $V \in \mathbb{R}^{N \times H \times W \times 3}$, where N is the number of frames, H is the height, and W is the width of the frames. Along with the video, a set of N_q queries is used.
The output consists of class predictions $c_i \in \mathbb{R}^{N_q \times N_c}$, where N_c is the number of classes, and bounding box regressions $b_i \in \mathbb{R}^{N_q \times 4}$, which describe the center position (c_x, c_y), width w, and height h for each detected object.
Initially, noisy bounding boxes $x_T = b_i^0$ are generated for each frame, and a neural network f_θ refines these predictions to match the ground truth bounding boxes $x_0 = b_i$. This process aims to enhance object detection and localization in the video.
A depiction of the proposed method is presented in Figure 2.9, and the following sections will explain each component in detail.

2.4.2.1 Query Initialization

In DiffusionVID [7], object-level queries are initialized from the interior regions of randomly generated noisy bounding boxes b_i^0 to cover most of the image frame. These queries are refined through cascade stages to improve object detection.
The initial object queries z_i^0 in an image I_i are computed as:

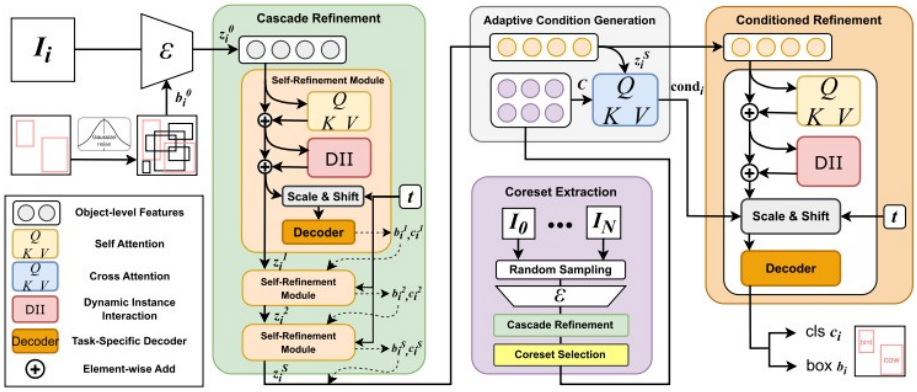

Figure 2.9: Overview of DiffusionVID [7].

$$z_i^0 = E(b_i^0, f_i), \quad \textbf{(2.2)}$$

where $f_i = F(I_i)$ is the feature map extracted from the image using a feature extractor F, such as a ResNet or Swin Transformer. RoI-Align is applied using b_i^0 and f_i to extract regional information, and the pooled features are averaged spatially to generate z_i^0.
These initialized queries are then refined in subsequent stages to obtain the final box coordinates and class predictions.

2.4.2.2 Cascade Refinement

Cascade refinement is an iterative process aimed at improving object detection by refining initial predictions over multiple stages. It starts with rough bounding box predictions and gradually refines them by adding more context and information.
In the first stage, bounding boxes are generated to identify potential object locations. These predictions guide the system's focus on relevant regions but are initially imprecise.
The core of the refinement process is self-refinement using self-attention, which adjusts queries by leveraging both spatial and visual context. Mathematically, this is expressed as:

$$z_s^i = S_s(z_{s-1}^i, b_{s-1}^i, f_i, t)$$

Where:

- z_s^i is the refined query at stage s;
- b_{s-1}^i are the previous bounding boxes;
- f_i is the feature map;
- t is the diffusion time step.

Over several iterations, self-refinement gradually improves query accuracy. By the final stage, the system can accurately localize objects and classify them with high precision.

However, the process depends on image quality (it uses only the current image information). Noisy, blurry, or low-resolution images can hinder the refinement, making it difficult to improve the queries or detect small details.
In summary, cascade refinement enhances object detection by iteratively improving initial predictions using self-attention and context. While it boosts localization and classification, its effectiveness is limited by the quality of the input image. In Section 2.4.2.3, a method will be introduced to mitigate the limitations of the cascade approach.

2.4.2.3 Dynamic Coreset Conditioning

The Dynamic Coreset Conditioning (DCC) method enhances VOD by addressing the limitations of cascade refinement mentioned in Section 2.4.2.2. DCC overcomes this by leveraging data from multiple frames.
The method is divided into three main components:

1. Coreset Construction:
   DCC constructs a coreset, a compact set of representative object queries selected from multiple frames. Instead of processing each frame individually, the coreset captures information from the entire video;

2. Adaptive Condition Generation:
   An attention mechanism creates condition vectors for each query in the coreset. These query-specific condition vectors pool information from different frames. Through this adaptive conditioning, the refinement step accommodates changes in the video data;

3. Conditioned Refinement:
   The condition vectors are integrated into the query refinement process using two methods:

   - Add: The condition vector is directly added to the queries;
   - Adaptive Norm: The normalization process is modified using the condition vector.

   This integration improves the refinement of object queries based on information from multiple frames, rather than a single frame.

In summation, DCC enhances detection performance while reducing computational cost by aggregating information from multiple frames and dynamically adjusting the refinement stage.

2.4.2.4 Local Batch Refinement

Local Batch Refinement (LBR) is a method designed to make VOD faster and more efficient. Traditional techniques process each video frame individually and repeatedly calculate the same frame features, which wastes time and slows down the system.
LBR solves this problem by grouping multiple frames together and processing them as a batch. This way, the system reduces redundant computation and increases per-GPU utilization.

2.4.2.5 Suitability of DiffusionVID [7] for Waste Container Detection Tasks

DiffusionVID [7] introduces a diffusion-based approach that iteratively refines noisy initial predictions to recover accurate bounding boxes. Its cascade refinement strategy and dynamic coreset conditioning leverage spatio-temporal features effectively, ensuring robustness against occlusions and environmental noise. Importantly, DiffusionVID offers a superior speed-accuracy trade-off, achieving competitive performance while operating 14.6 times faster than PTSEFormer (21.5 ms per frame).
Despite these advantages, DiffusionVID's [7] reliance on iterative refinement can lead to diminished precision in highly dynamic environments where frame-to-frame changes are rapid. While its design prioritizes efficiency, the model's performance may plateau in scenarios requiring exceptional accuracy, such as detecting small or partially obscured containers in crowded urban areas.

2.4.3 YOLOV: Making Still Image Object Detectors Great at Video Object Detection [8]

YOLOV [8], introduced by Yuheng Shi, Tong Zhang, and Xiaojie Guo, is designed to address these challenges, incurring minimal computational overhead while achieving significant improvements in accuracy. Unlike traditional two-stage pipelines, this method emphasizes the selection of key regions following the initial one-stage detection, thereby reducing the processing of numerous low-quality candidates. Additionally, the authors evaluate the relationships between the target frame and reference frames to optimize the aggregation of features effectively.

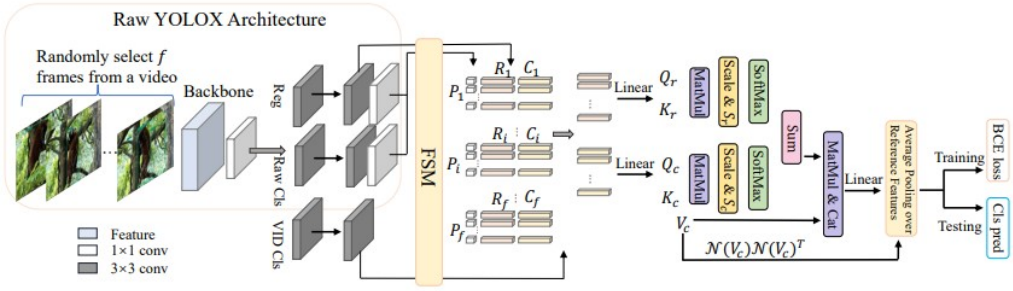

Figure 2.10: YOLOV [8] Architecture [8].

Figure 2.10 provides an overview of the framework of the proposed YOLOV [8] design, using YOLOX [37] as the base detector. The process begins with randomly sampling multiple frames from a video, which are then input into the YOLOX [37] detector to extract features. Based on the predictions from YOLOX [37], the Feature Selection Module (FSM) selects the top k confident proposals and applies NMS for further refinement. The features identified by the FSM are then fed into the Feature Aggregation Module (FAM) for final classification. This approach combines the efficiency of one-stage detectors with the accuracy benefits of temporal aggregation.

2.4.4 YOLOV++: Practical Video Object Detection via Feature Selection and Aggregation [9]

YOLOV++ [9] is an advanced version of the previous YOLOV [8] model, developed to enhance VOD performance.

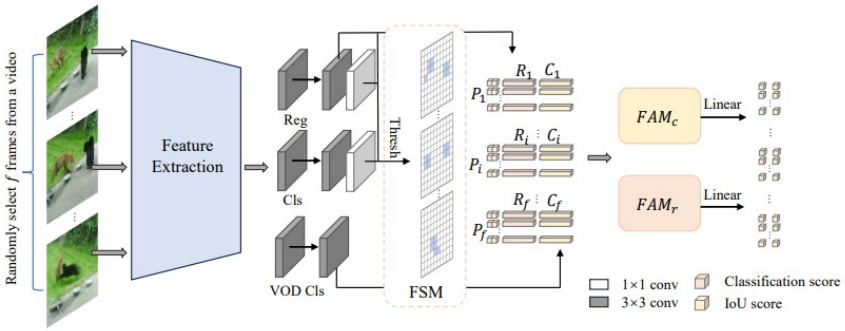

Figure 2.11: YOLOV++ [9] Architecture.

What sets YOLOV++ [9] apart from YOLOV [8] is:

- Revised NMS: While YOLOV's FSM uses NMS to keep only the most confident bounding boxes by discarding overlapping ones, YOLOV++ removes NMS entirely. Instead, it retains all candidates above a confidence threshold, reducing the chance of overlooking valid detections and enabling full use of the baseline detector's label assignment strategy;
- Enhanced Score Optimization in the FAM: In YOLOV [8], the FAM only adjusts classification scores. In contrast, YOLOV++ [9] refines both classification and IoU scores, leading to more precise alignment between predictions and actual object locations.

The Figure 2.11 illustrates the overall framework of the proposed method. It starts by randomly sampling several frames from the same video, which are then fed into the YOLOX [37] base detector to extract features. The FSM selects dense foreground proposals based on YOLOX [37] predictions, and these selected features are passed to the FAM for further refinement, generating classification and IoU scores. This two-step process—initial prediction followed by feature refinement using temporal aggregation—combines the efficiency of one-stage detectors with the improved accuracy provided by aggregating information across frames.

Key features and improvements of the method include the effective utilization of temporal information across frames through a newly introduced feature aggregation module, which enhances detection accuracy. By performing region selection after rough predictions, the method reduces computational load, significantly cutting down inference time while maintaining high accuracy. It also outperforms the base detector, YOLOV [8], in scenarios with heavy occlusion, showing improved robustness. Additionally, the method eliminates the need for NMS to reduce redundancy, retaining all candidates above a predefined confidence threshold, which improves the accuracy of selecting the correct regions. Finally, the model processes frames approximately three times faster than its predecessor, making it highly suitable for real-time applications.

Despite its robustness, YOLOV++ [9] still relies heavily on adequate training data diversity to generalize effectively to less common container types or lighting conditions. While its computational efficiency is impressive, achieving consistent performance in scenarios involving significant environmental variability, such as nighttime detection or severe occlusions, remains a challenge.

2.4.4.1 Suitability of YOLOV++ [9] for Waste Container Detection Tasks

YOLOV++ builds upon YOLOV by eliminating the need for NMS and introducing improved score optimization for classification and IoU. These advancements allow the model to retain more candidates during detection, reducing false negatives and enhancing its ability to handle occlusions and dense clutter. YOLOV++ [9] achieves approximately three times faster inference compared to YOLOV [8], making it highly practical for real-time waste collection scenarios.

Despite its robustness, YOLOV++ [9] still relies heavily on adequate training data diversity to generalize effectively to less common container types or lighting conditions. While its computational efficiency is impressive, achieving consistent performance in scenarios involving significant environmental variability, such as nighttime detection or severe occlusions, remains a challenge.

2.5 Overview of MOT

MOT enables the consistent identification and tracking of multiple objects across video frames. In the context of this dissertation, MOT is vital for identifying and counting waste containers in urban environments, a task with practical implications for waste management systems as it enables automated estimation of container type, quantity, and location within a given area. Most modern MOT approaches follow the Tracking by Detection (TBD) paradigm, where objects are first detected in each video frame and then linked across time to form continuous tracks. Within this paradigm, data association strategies can be broadly divided into two main families: motion-based and appearance-based trackers. The first category, motion-based trackers, relies exclusively on spatial and temporal information such as bounding box coordinates and motion patterns to associate object detections across consecutive frames. These methods are typically lightweight and efficient but may struggle in complex scenarios like occlusions or when multiple objects move similarly. The second category, appearance-based trackers, extends the capabilities of motion-based approaches by incorporating deep appearance features, often in the form of re-identification embeddings. By using visual features, these trackers improve consistent identities over time and are more robust in visually ambiguous scenes. This contrast illustrates a trade-off in MOT between computational efficiency and robustness to visual ambiguity.

The following sections will explore representatives of each category and conclude with a discussion justifying the selection of the most appropriate tracker for this thesis based on application requirements: robustness to occlusion and accuracy in container count estimation.

2.5.1 Motion-Based Trackers

Among motion-based trackers, prominent examples include Simple Online and Realtime Tracking (SORT) [14] and ByteTrack [12]. By relying solely on object position, velocity, and trajectory, they offer high efficiency, though often at the expense of robustness in visually complex scenes.

2.5.1.1 SORT [14]

SORT [14] uses a Kalman Filter to predict the future position of an object based on its motion and employs the Hungarian algorithm to associate current detections with predicted object states. However, it has limitations in challenging scenarios; for example, it often assigns new identities to the same object when temporary occlusions or detection failures occur, leading to frequent ID switches.

2.5.1.2 ByteTrack: Multi-Object Tracking by Associating Every Detection Box [12]

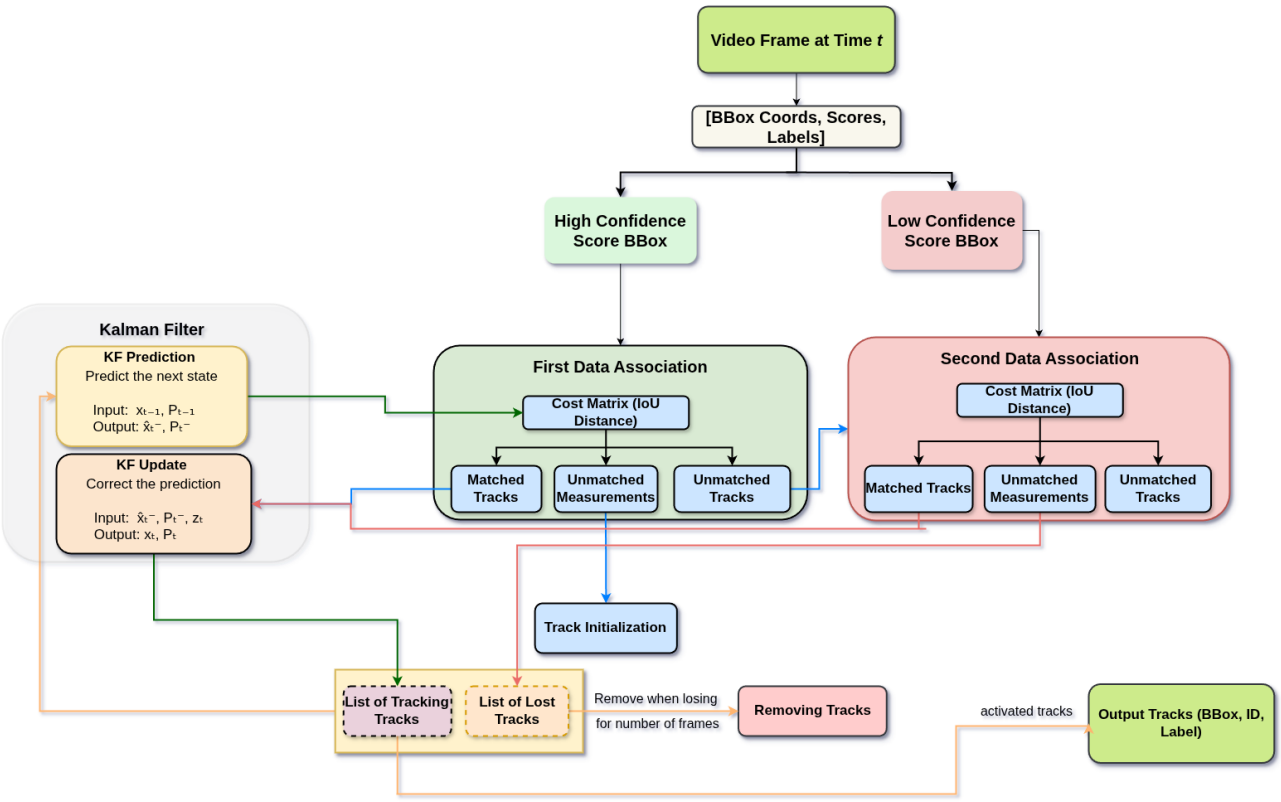

Figure 2.12: ByteTrack [12] pipeline adapted from [13].

ByteTrack [12] utilizes bounding box detection results, including both high and low confidence scores. As shown in Figure 2.12, for each video frame at time t, it first obtains object detections and separates them into high-confidence and low-confidence groups based on a set threshold. A Kalman Filter [38] predicts the next position and uncertainty of each tracked object. In the first stage, it matches these predicted tracks with high-confidence detections using IoU similarity. Matched tracks are updated with the new detections, while unmatched

high-confidence detections initialize new tracks. In the second stage, tracks that were not matched in the first stage attempt to match with low-confidence detections to recover any missed associations. These matches are also updated using the Kalman Filter [38]. Tracks that remain unmatched after this process are marked as lost and removed if they stay unmatched for n consecutive frames. This two-stage matching process helps reduce identity switches caused by occlusions or changes in object scale as objects move closer or farther from the camera. The final output at each frame is a list of active tracks, each containing the object's bounding box, ID, and class label.

2.5.2 Appearance-Based Trackers

Appearance-based trackers incorporate deep learning models that generate feature embeddings for each detected object. These embeddings, often derived from CNN, provide information about the visual identity of objects.

2.5.2.1 DeepSORT: Simple Online and Realtime Tracking with a Deep Association Metric [15]

DeepSORT [15] extends the original SORT [14] by introducing a deep association metric that compares appearance features between detections and existing tracks. This helps to reduce ID switches and re-identify objects after they re-enter the frame. While it performs better in crowded scenes, it requires more computation due to its use of a CNN for feature extraction.

2.5.2.2 BoT-SORT: Robust Associations Multi-Pedestrian Tracking [17]

BoT-SORT [17] combines ByteTrack's [12] matching logic with ReID-based appearance features and motion consistency filtering, but introduces increased latency.

2.5.3 Tracker Selection Justification

In our work, we address the task of tracking and counting waste containers in videos captured by a moving vehicle. These videos presents several challenges: containers often appear only briefly, may be partially occluded by obstacles, and vary in scale due to the vehicle's motion. Although the system runs on a Jetson Nano, real time inference is not required, as detection and tracking are performed post hoc, allowing for additional processing using heuristic rules. Given this setup, appearance-based trackers such as DeepSORT [15] and BoT-SORT [17] offer strong identity consistency but are computationally demanding, making them unsuitable for deployment on resource-constrained edge devices. In contrast, SORT [14] is fast and lightweight but suffers from frequent identity switches in scenarios with occlusion or missed detections.

ByteTrack [12] presents a favorable trade-off between robustness and efficiency. By associating both high- and low-confidence detections, it mitigates ID switches and track fragmentation while avoiding the computational overhead of deep appearance models. This makes

ByteTrack [12] the most appropriate choice for our pipeline, meeting both the challenges posed by the dataset and the constraints of the target hardware and post-processing strategy.

2.6 Conclusions

This chapter discussed the strengths and limitations of current state-of-the-art object detectors when applied to Waste Container Detection tasks.

One-stage image detectors, such as YOLO [1] and SSD [3], excel in real-time applications due to their speed and efficiency. However, they struggle with detecting small objects and handling cluttered environments, which are common challenges in waste collection scenarios. This issue is particularly relevant for certain container types, such as battery disposal bins, which are significantly smaller than general waste, recycling, or organic waste containers. Conversely, two-stage detectors like Faster R-CNN [25] and Mask R-CNN [5] achieve higher accuracy, especially in complex scenes. Yet, their computational demands make them impractical for real-time deployment in waste collection systems.

Video object detectors are well-suited for waste collection scenarios, where varying lighting conditions and container occlusions pose significant challenges. These methods leverage temporal information to improve detection robustness, making them particularly valuable for real-world waste detection tasks. PTSEFormer [6] enhances feature aggregation but suffers from significant computational overhead, limiting its real-time applicability. DiffusionVID [7] offers a better balance between speed and accuracy, making it a strong candidate for practical deployment. YOLOV++ [9] excels in real-time performance, aligning well with the operational constraints of waste collection vehicles.

Considering the studied methods, the most prominent image-based detector for waste container detection is YOLO, particularly its latest iterations, due to their efficiency and adaptability to real-world conditions. For video-based detection, DiffusionVID stands out as the best choice, as it offers an optimal balance between detection accuracy and computational efficiency.

Regarding multiple object tracking, appearance-based trackers offer improved identity consistency but come with increased computational costs, which can be prohibitive on edge devices like the Jetson Nano. Motion-based trackers such as SORT [14] are lightweight but suffer from ID switches during occlusions. ByteTrack [12] provides an optimal trade-off by using both high- and low-confidence detections to reduce ID switches without incurring heavy computational demands. Therefore, ByteTrack was selected for this work due to its robustness and efficiency, making it well-suited to the specific challenges of waste container tracking and the constraints of the deployed hardware.

Chapter 3

Proposed Dataset and Heuristic-Based Track Refinement for Waste Container Counting

3.1 Introduction

This chapter introduces the main contributions of our work: a novel dataset for waste container detection in urban environments, and a set of post-processing heuristics designed to improve tracking consistency and enable accurate container counting across video sequences.

To extract actionable insights from our data, we design a detection and tracking pipeline that integrates a frame-based object detector [10] (see justification in Section 4.3) with a multi-object tracker [12] (see Section 2.5.2.2). To address common challenges such as fragmented tracks, short-lived detections, and inconsistencies caused by occlusion or motion blur, we propose a set of post-processing heuristics. Specifically, we introduce three heuristics: (H1) filtering tracks below a minimum duration threshold, (H2) temporally merging tracks separated by short detection gaps, and (H3) spatially merging tracks based on proximity. These heuristics are applied to the raw tracker output to mitigate issues such as ID switches and fragmented tracks resulting in more coherent and reliable trajectories.

The dataset used in this work is introduced in detail in the following subsection. While our experiments, presented in Chapter 4, were conducted on both the base dataset and an augmented version that includes additional videos, results are reported separately. This section presents the statistics of the final, augmented dataset. The remainder of this section also explains the detection and tracking components and details the post-processing heuristics that enable our system to approximate real-world container counts more accurately.

3.2 Dataset

To support the development and evaluation of waste container detection and tracking systems in urban environments, we introduce a new dataset comprising 144 videos, totaling 48 minutes and 44 seconds of footage. Of this, 16 minutes and 42 seconds depict visible waste containers, while 32 minutes and 2 seconds contain only background scenes. The dataset consists of 29,248 frames, preprocessed from video data provided by EVOX [39].

The dataset is divided into three non-overlapping splits: training (101 videos / 18,518 images), validation (21 videos / 5,176 images), and test (22 videos / 5,554 images). Care was taken to ensure that frames from a given video are present in only one split, preserving the integrity of the evaluation process and preventing data leakage.

Each visible waste container instance is annotated with a bounding box, resulting in a total of 34,407 labeled instances. In addition to annotated containers, the dataset includes frames

that contain only background, providing valuable negative samples to train models to distinguish between container and non-container scenes. Background frames account for approximately twice the number of container frames, creating a realistic class imbalance scenario encountered in real-world applications. A summary of the distribution of videos across the dataset splits is provided in Table 3.1, distinguishing between those containing waste containers and those with background-only footage. Since the images were extracted from these videos, we also report the distribution of images across the dataset splits in Table 3.2, including counts of bounding boxes and background images. These statistics provide context for the subsequent experiments, which include both an image-based detection model [10] and a video-based detection model [7].

The videos were recorded under natural daylight, with all scenes captured in sunny weather conditions. While the dataset encompasses a variety of lighting conditions and occlusions, it does not include nighttime or adverse weather scenarios. This limits the applicability of models trained on this dataset to such conditions. To improve robustness and generalization, future extensions of the dataset should include footage recorded under low-light and severe weather conditions.

Dataset Metric	**Split**			**Total**
	Training	**Validation**	**Test**	
Videos with Containers	66	13	14	93
Background Videos (No Containers)	35	8	8	51
Total Videos	101	21	22	144

Table 3.1: Summary of Dataset Statistics by Video Split.

Dataset Metric	**Split**			**Total**
	Training	**Validation**	**Test**	
Images with Containers	6,520	1,828	1,674	10,022
Images without Containers (Background)	11,998	3,348	3,880	19,226
Total Images	18,518	5,176	5,554	29,248

Table 3.2: Summary of Dataset Statistics by Image Split.

The dataset features a generally balanced distribution of videos across most classes in the training, validation, and test splits. The `container_default` class is the most prevalent, appearing in the highest number of videos overall. However, some classes such as `container_ash` and `container_biodegradable` are notably underrepresented, which may present challenges for model generalization on these rarer categories.

In terms of annotated instances, `container_default` also dominates, comprising nearly half of all instances. Meanwhile, classes like `container_yellow`, `container_green`, and `container_blue` each contribute a significant portion, around 14–16% of the data. The remaining smaller classes collectively account for less than 8%, indicating a pronounced class imbalance.

This imbalance may lead to a model biased towards the more frequent classes, potentially limiting its performance on less common types.

Class	Train Split	Val Split	Test Split	Total Videos
container_yellow	33	5	9	47
container_green	36	5	9	50
container_default	61	12	13	86
container_blue	35	5	9	49
container_oil	4	1	3	8
container_battery	10	1	6	17
container_biodegradable	3	1	1	5
container_ash	1	1	1	3

Table 3.3: Number of Videos per Split in which Each Class Appears.

Class	Train Split	Val Split	Test Split	Total Instances
container_yellow	3,175 (15.17%)	668 (11.01%)	1,174 (15.85%)	5,017 (14.58%)
container_green	3,200 (15.28%)	720 (11.87%)	1,089 (14.71%)	5,009 (14.56%)
container_default	9,565 (45.69%)	3,519 (58.01%)	2,913 (39.34%)	15,997 (46.49%)
container_blue	3,527 (16.85%)	650 (10.72%)	1,305 (17.62%)	5,482 (15.93%)
container_oil	483 (2.31%)	155 (2.56%)	252 (3.40%)	890 (2.59%)
container_battery	693 (3.31%)	135 (2.23%)	466 (6.29%)	1,294 (3.76%)
container_biodegradable	208 (0.99%)	135 (2.23%)	179 (2.42%)	522 (1.52%)
container_ash	85 (0.41%)	84 (1.38%)	27 (0.36%)	196 (0.57%)
Total	**20,936 (100.00%)**	**6,066 (100.00%)**	**7,405 (100.00%)**	**34,407 (100.00%)**

Table 3.4: Class Instances per Split and Total Across All Splits.

3.3 Post-Processing Tracking Heuristics

The objective of our tracking pipeline is to generate consistent object tracks for garbage containers across video frames. The tracker operates on detections from the image-based detector [10] (justified in Section 4.3) and is implemented using a tracking algorithm [12] (justified in Section 2.5.2.2). An overview of the complete tracking pipeline, including the post-processing heuristics and evaluation flow, is illustrated in Figure 3.1. Our main contribution lies in the post-processing block of this pipeline, where we introduce a set of heuristics designed to improve track consistency and reliability.

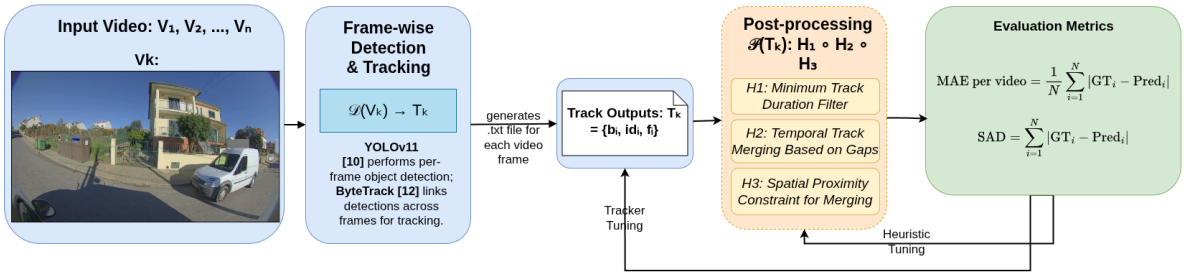

Figure 3.1: Overview of the proposed waste container detection and tracking pipeline. The image-based detector [10] performs per-frame object detection, while the tracker algorithm [12] manages multi-object tracking across video frames. The resulting tracks are post-processed using three heuristics: (H1) minimum duration filtering, (H2) temporal merging based on detection gaps, and (H3) spatial proximity constraints. Final evaluation metrics (MAE, SAD) are used to guide both tracker configuration and heuristic tuning.

3.3.1 Metrics

To evaluate tracking quality, we use two complementary metrics: the MAE per video and the SAD. These metrics are selected over standard MOT benchmarks such as Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), or IDF1, as our primary objective is not frame-level tracking accuracy, but rather consistent object counting across complete video sequences.

MAE per video quantifies the average discrepancy between the predicted and ground-truth container counts for each video. It is defined as:

$$\text{MAE per video} = \frac{1}{N}\sum_{i=1}^{N}|\text{GT}_i - \text{Pred}_i| \quad (3.1)$$

where N is the total number of videos, GT_i is the ground-truth number of unique containers in video i, and Pred_i is the predicted count. This metric equally penalizes overcounting and undercounting and reflects the system's ability to maintain consistent identities across frames.

SAD captures the cumulative absolute error across the dataset and is defined as:

$$\text{SAD} = \sum_{i=1}^{N}|\text{GT}_i - \text{Pred}_i| \quad (3.2)$$

While MAE provides a normalized view of error on a per-video basis, SAD reflects the total deviation in container counts over the entire evaluation set. Both metrics are particularly suited for evaluating urban waste container tracking systems, where the goal is accurate object quantification to support real-world logistics and resource planning.

3.3.2 Motivation: Understanding Tracker Errors Through Object Scale Dynamics

The data was collected using a moving vehicle, which naturally introduces perspective distortion: as the vehicle approaches a container, its bounding box area increases; as it drives past, the area decreases. This typically results in a bell-shaped pattern when plotting the bounding box area over time for a given object.
We leverage this phenomenon to assess identity consistency. However, frequent identity switches were observed, particularly when containers are farther from the camera, which leads to broken or short-lived tracks.
By analyzing these patterns across multiple videos and container classes, we developed post-processing heuristics specifically designed to mitigate ID fragmentation issues.
Figures 3.2, 3.3, 3.4, and 3.5 illustrate representative examples from specific videos and container classes. In most cases, the bell-shaped area curve is clearly visible, along with identity switches and fragmented tracks. These tracking inconsistencies result in significant errors in container counts.

3.3.3 Heuristic's

- H1: Minimum Track Duration Filter
  Tracks shorter than a threshold (N frames) are removed. These short-lived tracks often result from unreliable detections when the container is distant from the camera, as indicated by small bounding box areas and unstable ID assignments. Removing them reduces false positives and temporal noise. The effect of this heuristic is illustrated in Figure 3.2 (prior to filtering) and Figure 3.3 (after filtering).

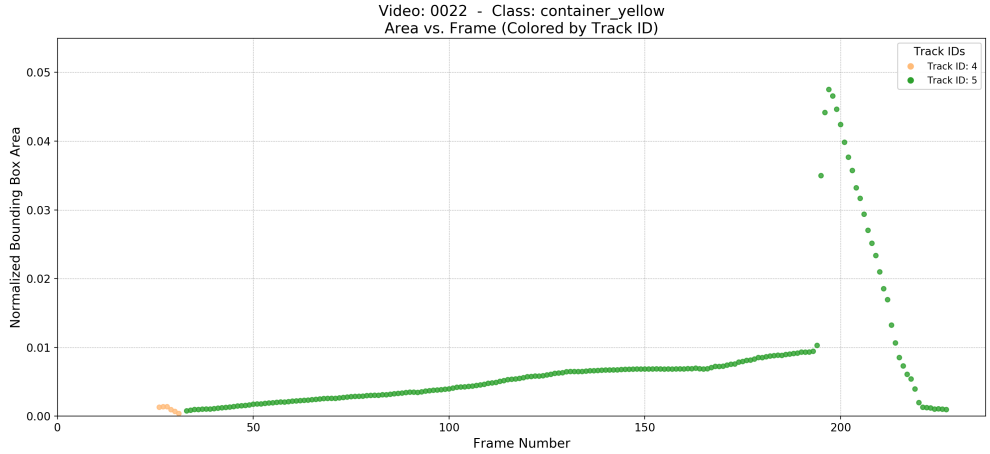

Figure 3.2: Early frames show incorrect track ID switches, highlighted in yellow (each track ID is uniquely color-coded to improve distinction).

- H2: Temporal Track Merging Based on Gaps
  Two tracks of the same class are merged if the temporal gap between them is less than or equal to N frames. This heuristic addresses track fragmentation by assuming such gaps may still belong to the same object. Figure 3.4 shows fragmented IDs before merging,

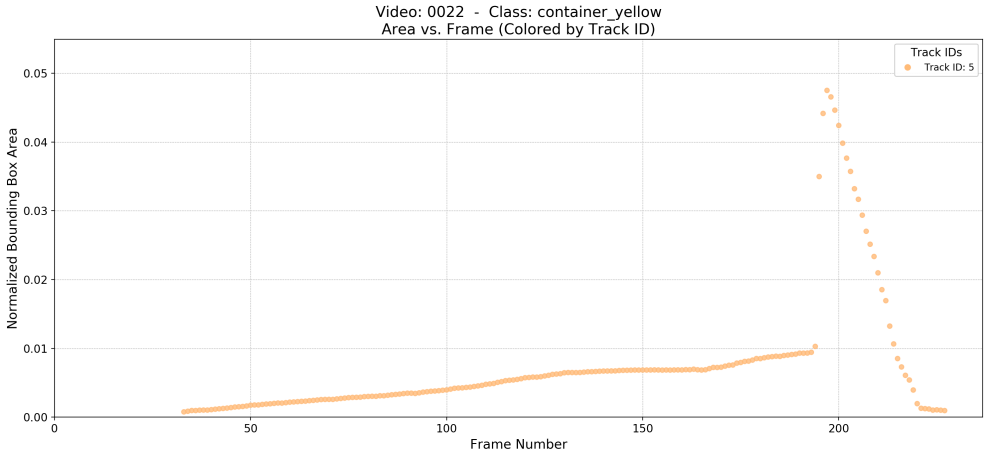

Figure 3.3: Heuristic H1 eliminate the initial track.

while Figure 3.5 demonstrates the result after applying this heuristic.

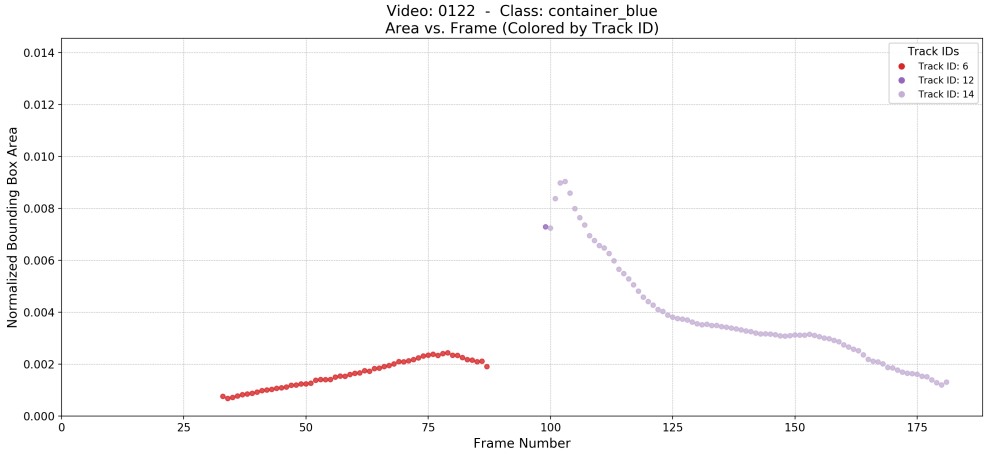

Figure 3.4: Area trajectory showing a track fragment caused by occlusion. The baseline tracker segments this into separate tracks (each track ID is uniquely color-coded to improve distinction).

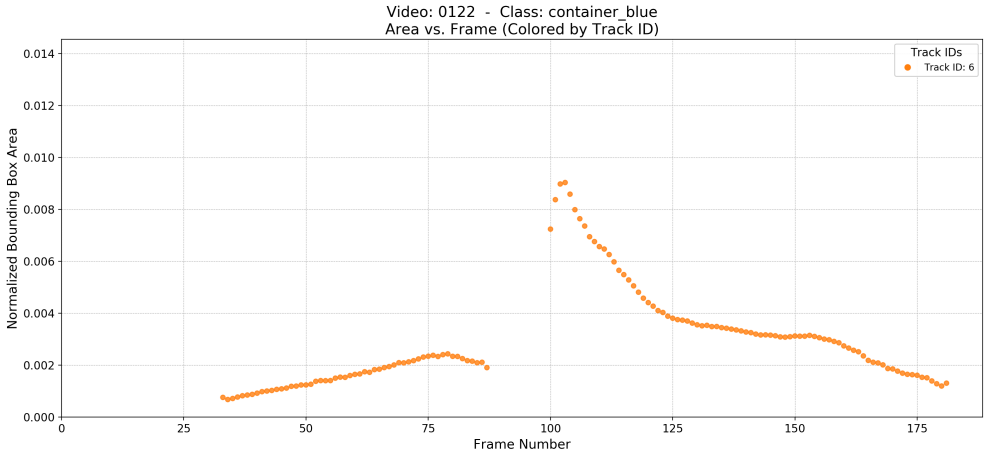

Figure 3.5: Heuristic H1 remove the incorrect idswitch (12) and H2 merged the tracks (6, 14).

- H3: Spatial Proximity Constraint for Merging

Extends H2 by enforcing a spatial proximity condition: the distance between the last detection of the first track and the first detection of the second track must be within a maximum center distance threshold (N normalized image units);

3.4 Conclusion

In this chapter, we introduced the contributions of this work: a dataset designed for waste container detection and tracking in urban environments, and a set of heuristic-based post-processing techniques aimed at improving tracking consistency and counting accuracy. The dataset comprises over 29,000 frames (more than 10,000 annotated), with a realistic distribution of container and background scenes (background frames are approximately twice as numerous as container frames), and includes eight waste container classes with varying levels of representation. This setup reflects real-world challenges such as class imbalance and visual occlusion.

We developed a detection and tracking pipeline that combines a frame-based object detector [10] (see Section 4.3) with a multi-object tracker [12] (see Section 2.5.2.2). We observed that the raw tracker output often suffered from fragmented and short-lived tracks, primarily due to object scale variations and occlusions. To address these issues, we proposed three post-processing heuristics: minimum duration filtering (H1), temporal gap-based merging (H2), and spatial proximity-based merging (H3). These heuristics aim to improve the coherence of track identities and reduce both overcounting and undercounting errors. Their effectiveness is evaluated using MAE and SAD, which are metrics focused on video-level object quantification, as opposed to frame-level tracking fidelity metrics such as MOTA and Higher Order Tracking Accuracy (HOTA), commonly used in tracking benchmarks.

Chapter 4

Experiments

4.1 Introduction

This chapter evaluates deep learning-based object detection models for urban waste container recognition, comparing the image-level YOLOv11 detector [10] with the video-based DiffusionVID [7], which leverages temporal context. Experiments are conducted on a two-stage dataset provided by EVOX [39]. The first stage serves as the base dataset, while the second stage, detailed in Section 3.2, introduces additional video recordings to augment the training material.

Our evaluation begins with YOLOv11 [10] as a baseline single-frame detector, followed by a comparison with DiffusionVID [7] to assess the benefits of temporal modeling. We also investigate whether the expanded dataset improves the performance of DiffusionVID [7] to a level that surpasses YOLOv11 [10].

Performance is analyzed both quantitatively using PR curves and mAP, and qualitatively through scenario-driven visualizations. Additionally, we address temporal consistency by integrating a ByteTrack-based tracker [12], enhanced with heuristics for temporal smoothing and spatial coherence. These refinements are evaluated on both dataset stages and on continuous, long-duration videos that simulate real-world deployment conditions.

Overall, this chapter presents a comprehensive analysis of frame-based and video-based detection, the impact of adding more data through augmentation, and the effectiveness of heuristic refinements in tracking for robust waste container detection in smart city applications.

4.2 Base Dataset

The base dataset comprises 84 videos with a total duration of 33 minutes and 2 seconds, corresponding to 19,826 frames. Of this, 11 minutes and 26 seconds contain waste containers, while the remaining 21 minutes and 36 seconds capture background scenes. The dataset is split into training (12,058 images), validation (3,689 images), and test (4,079 images) sets. To ensure temporal independence across splits and avoid data leakage, frames from each video are assigned exclusively to a single subset. Detailed statistics on image- and video-level distributions are provided in Table 4.2 and Table 4.1.

The statistics in Table 4.3 and Table 4.4 highlight the critical limitations that necessitated our data augmentation strategy. The base dataset exhibits a severe class imbalance, with rare classes such as `container_biodegradable`, `container_battery`, and `container_oil` collectively comprising less than 10% of all instances. More severely, our strict video-based partitioning led to the complete absence of these rare classes from the validation set. To

Video Metric	Split			Total
	Training	Validation	Test	
Videos with Containers	35	10	8	53
Background Videos (No Containers)	21	5	5	31
Total Videos	56	15	13	84

Table 4.1: Summary of Video Statistics for Training, Validation, and Test Splits (Base Dataset).

Image Metric	Split			Total
	Training	Validation	Test	
Images with Containers	4,351	1,371	1,139	6,861
Images without Containers (Background)	7,707	2,318	2,940	12,965
Total Images	12,058	3,689	4,079	19,826

Table 4.2: Summary of Image Statistics for Training, Validation, and Test Splits (Base Dataset).

overcome this, our augmentation (described in detail in Section 4.3) was designed to populate these classes.

Class	Train Split	Val Split	Test Split	Total Instances
`container_default`	7,078 (44.24%)	2,148 (84.90%)	1,941 (45.02%)	11,167 (48.89%)
`container_blue`	2,689 (16.81%)	164 (6.48%)	650 (15.08%)	3,503 (15.34%)
`container_yellow`	2,405 (15.03%)	172 (6.80%)	659 (15.29%)	3,236 (14.17%)
`container_green`	2,402 (15.01%)	46 (1.82%)	657 (15.24%)	3,105 (13.59%)
`container_oil`	594 (3.71%)	0 (0.00%)	137 (3.18%)	731 (3.20%)
`container_battery`	488 (3.05%)	0 (0.00%)	88 (2.04%)	576 (2.52%)
`container_biodegradable`	343 (2.14%)	0 (0.00%)	179 (4.15%)	522 (2.29%)
`container_ash`	0 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Total	**15,999 (100.00%)**	**2,530 (100.00%)**	**4,311 (100.00%)**	**22,840 (100.00%)**

Table 4.3: Class Instances per Split and Total Across All Splits (Base Dataset).

Class	Train Split	Val Split	Test Split	Total Videos
`container_yellow`	20	2	4	26
`container_green`	22	1	4	27
`container_default`	33	8	8	49
`container_blue`	21	2	4	27
`container_oil`	4	0	1	5
`container_battery`	6	0	1	7
`container_biodegradable`	4	0	1	5
`container_ash`	0	0	0	0

Table 4.4: Number of Videos per Split in which Each Class Appears (Base Dataset).

4.2.1 Experiments with an Image-Based Detector [10]

YOLOv11 [10] (`yolov11-l`) was trained on the waste container dataset (base dataset) using a set of specific hyperparameters. The model was trained with a learning rate of 1×10^{-4} and a batch size of 4. The training process was set to run for 300 epochs, with a patience of 20 to prevent overfitting. Images were resized to 640×640 pixels during training. Convergence occurred after 94 epochs, at which point the validation loss plateaued, indicating that the model had effectively learned from the data.

4.2.1.1 Image-Based Detector [10] Results

From the PR curve, illustrated in Figure 4.1, obtained on the test split, it is clear that the model performs well overall, achieving an mAP of 0.912. The default class stands out with an almost perfect PR curve and an mAP of 0.973, due to its abundant representation in the dataset (11,167 instances). Similarly, the biodegradable class (mAP 0.959) also performs well, despite having fewer instances. On the other hand, the battery class struggles, with an mAP of 0.772.

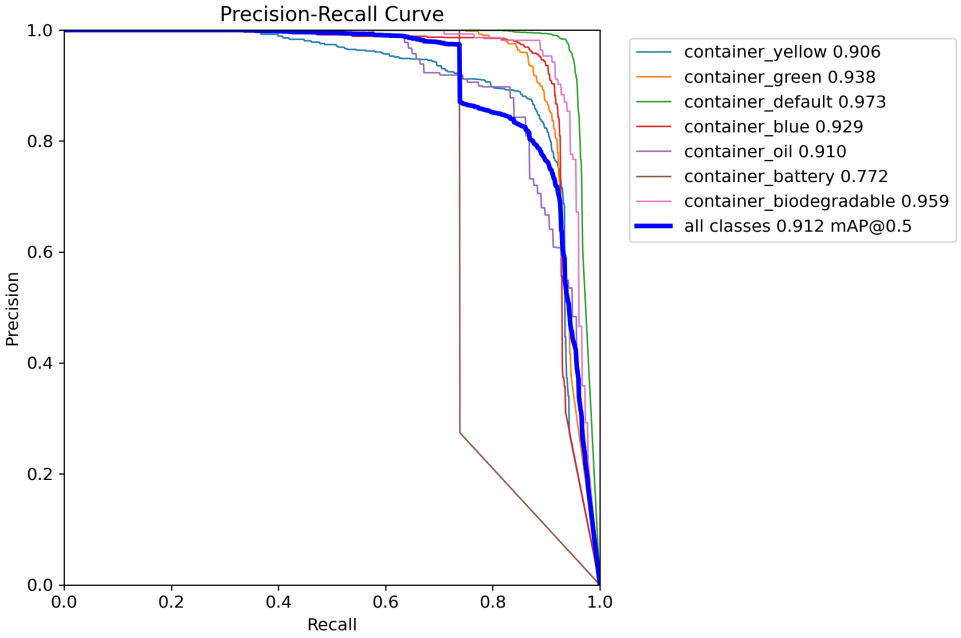

Figure 4.1: Precision-Recall Curve for YOLOv11 [10].

4.2.1.2 Strengths Analysis

In the first scenario, depicted in Figure 4.2, the model demonstrates its ability to detect all containers, even when they are at a greater distance from the camera. In the second scenario, shown in Figure 4.3, the model performs well in a closer setup, handling illumination variations effectively.

In the third scenario, presented in Figure 4.4, the model presents a complex scene where containers are surrounded by vehicles and other elements. It successfully focuses on the target

objects while ignoring irrelevant distractions. Additionally, the prediction image includes a bounding box around a portion of a container that was not labeled as such, because it did not meet the threshold for being considered a container during the labeling process. However, this part is indeed part of a container, highlighting the model's attention to detail.

The fourth scenario, described in Figure 4.5, presents a more challenging setup where containers are positioned close together, each shown from a different perspective. Despite these complexities, YOLOv11 [10] maintains accurate detection.

Finally, the fifth scenario, illustrated in Figure 4.6, highlights the model's ability to handle densely packed setups, accurately identifying and separating closely positioned containers.

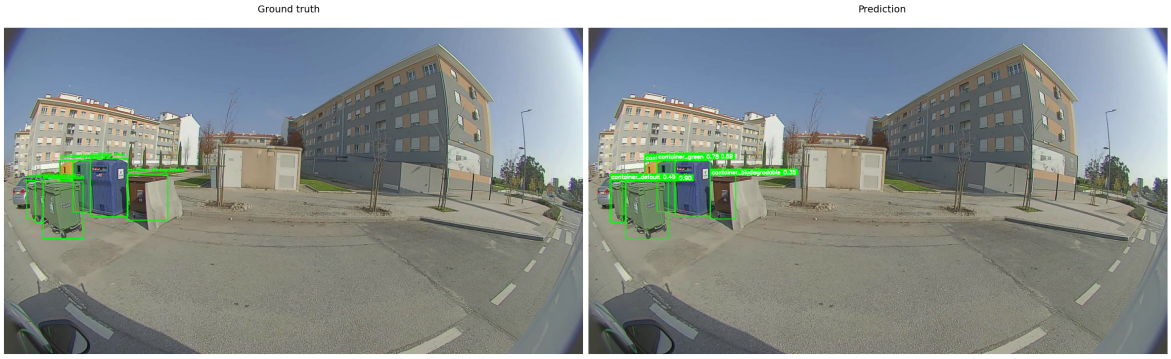

Figure 4.2: YOLOV11 [10] Scenario 1.

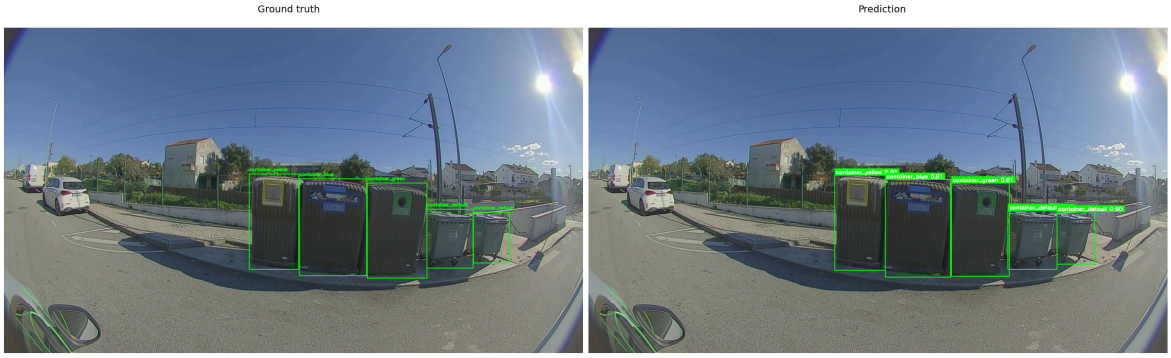

Figure 4.3: YOLOV11 [10] Scenario 2.

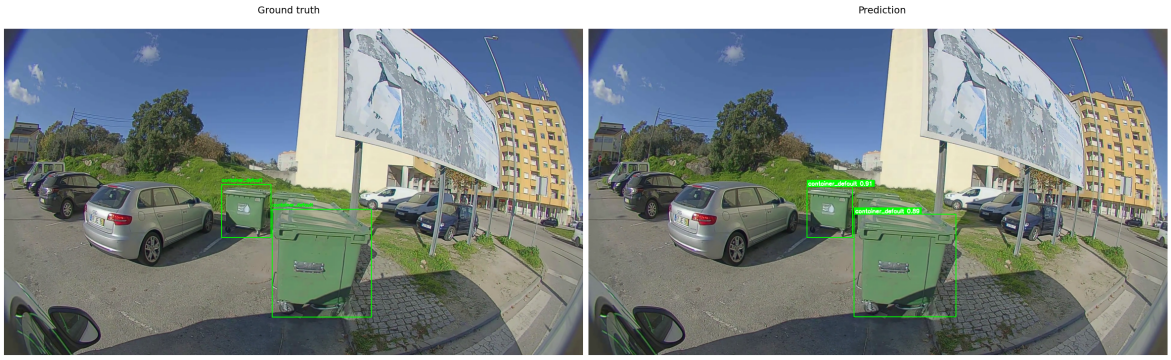

Figure 4.4: YOLOV11 [10] Scenario 3.

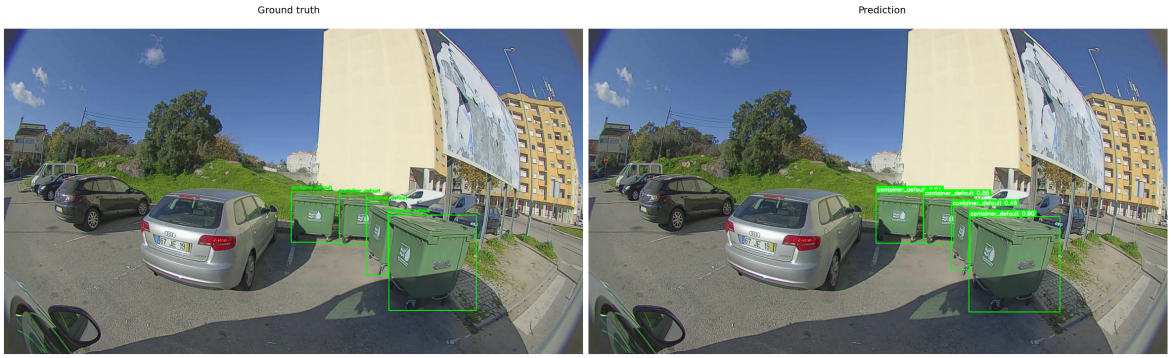

Figure 4.5: YOLOV11 [10] Scenario 4.

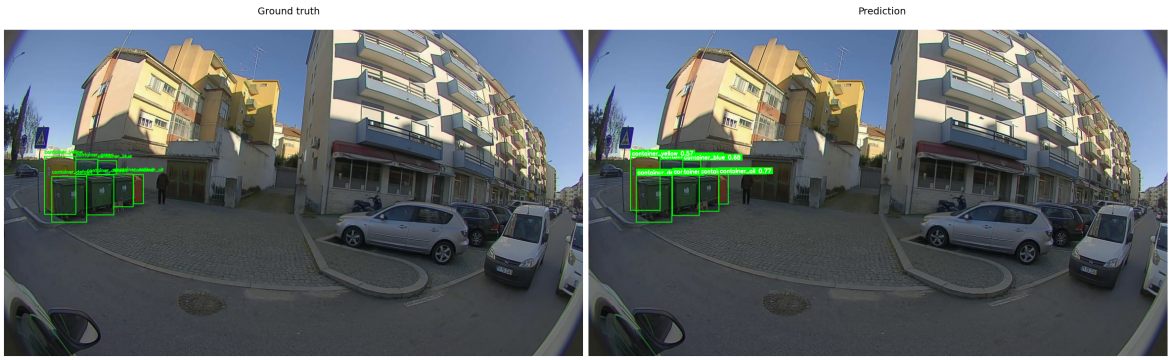

Figure 4.6: YOLOV11 [10] Scenario 5.

4.2.1.3 Limitations Analysis

This section highlights some challenges faced by the model in specific scenarios. In the sixth scenario, the model struggles with a scene where the containers are at a significant distance. While some containers are correctly detected, the model falsely identifies more containers than actually exist, indicating over-detection in this scenario.
In the seventh scenario, as depicted in Figure 4.8, the model fails to detect a battery container because it is too far from the camera. This limitation likely arises due to the lack of similar representations of distant battery containers in the training and validation data splits.
In the eighth scenario, as shown in Figure 4.9, the model predicts an incorrect bounding box, which corresponds to a part of the image's background rather than an actual container.

Figure 4.7: YOLOV11 [10] Scenario 6.

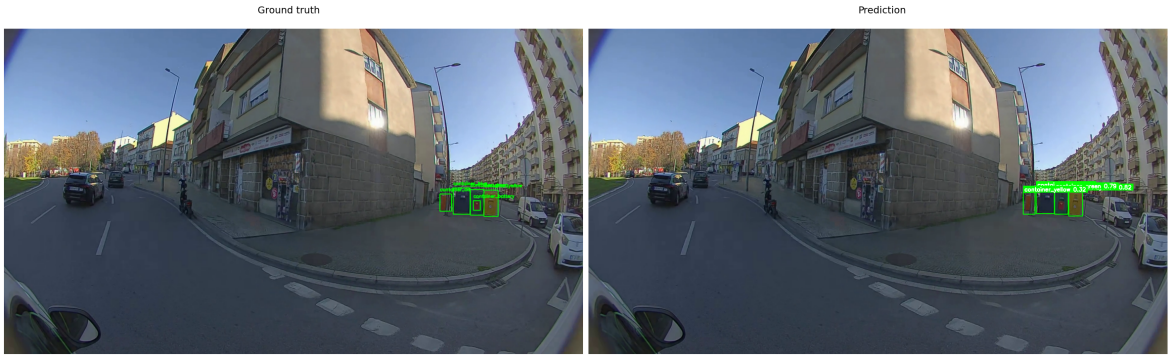

Figure 4.8: YOLOV11 [10] Scenario 7.

Figure 4.9: YOLOV11 [10] Scenario 8.

4.2.2 Experiments with a Video-Based Detector [7]

The DiffusionVID [7] model (`DiffusionVID_R101`) was trained on the waste container dataset (base dataset) using the following hyperparameters. The base learning rate was set to 1×10^{-4}. During inference, the effective batch size was 8, with an additional factor of 2 for accumulation steps. The training ran for a total of 80,000 epochs, which corresponds to the maximum number of iterations, and included a warmup period of 7,000 iterations.

4.2.2.1 Video-Based Detector [7] Results

Observing Figure 4.10, the highest Average-Precision (AP) is achieved for the default class (0.9623), followed closely by green (0.925) and blue (0.921). However, lower AP scores are observed for the underrepresented classes, such as battery (0.850) and biodegradable (0.900). Additionally, the yellow class shows a noticeable decline in performance.

4.2.2.2 Strengths Analysis

Analyzing the scenarios where DiffusionVID [7] performed in alignment with the ground truth, it's possible to note that in the first scenario, depicted in Figure 4.11, all containers were correctly identified despite challenging perspectives and considerable distance from

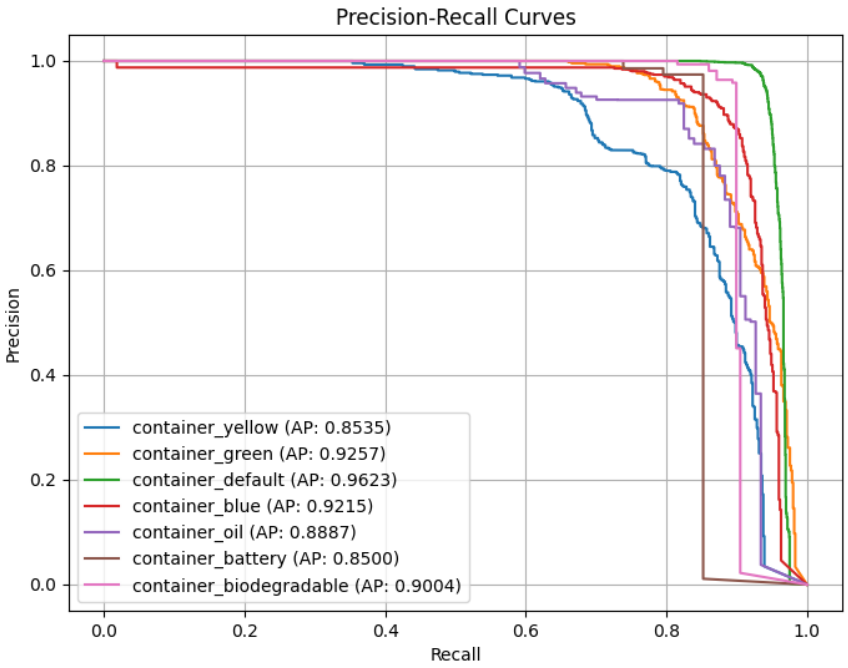

Figure 4.10: Precision-Recall Curve for DiffusionVID [7].

the objects. In the second scenario, presented in Figure 4.12, which represents the ideal case where the garbage truck is positioned very close to the containers.
In the third scenario, illustrated in Figure 4.13, the model successfully identified all types of containers, even under conditions of overloading. The fourth scenario demonstrates that despite the increased distance, the model was able to correctly identify all container types, including the battery container, which had previously been misclassified by YOLOv11 [10], as shown in Figure 4.9.
In the fifth scenario, depicted in Figure 4.15, all containers were correctly identified, with no false positives, in contrast to the eighth scenario of YOLOv11 [10], illustrated in Figure 4.9, where a container was falsely identified in the background. Finally, in the sixth scenario shown in Figure 4.16, the model exclusively identified the containers present in the scene with precision, unlike YOLOv11 [10] in the corresponding sixth scenario showed in Figure 4.7, where it erroneously detected more containers than were actually present in the image.

Figure 4.11: DiffusionVID [7] Scenario 1.

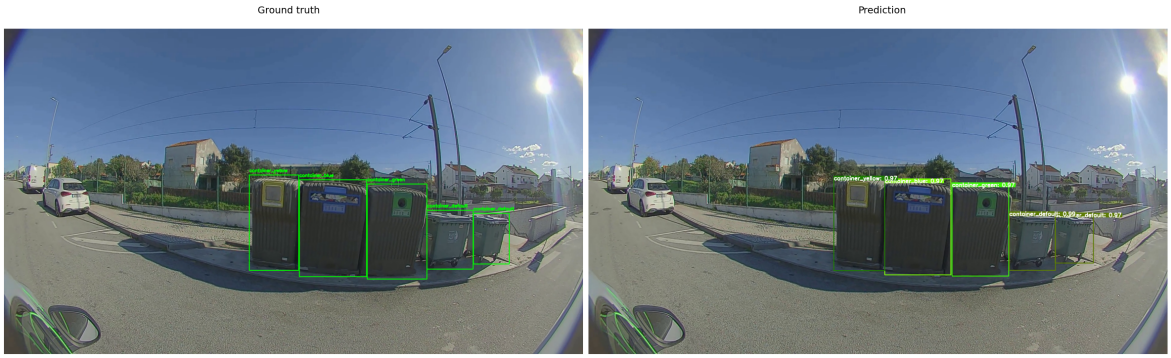

Figure 4.12: DiffusionVID [7] Scenario 2.

Figure 4.13: DiffusionVID [7] Scenario 3.

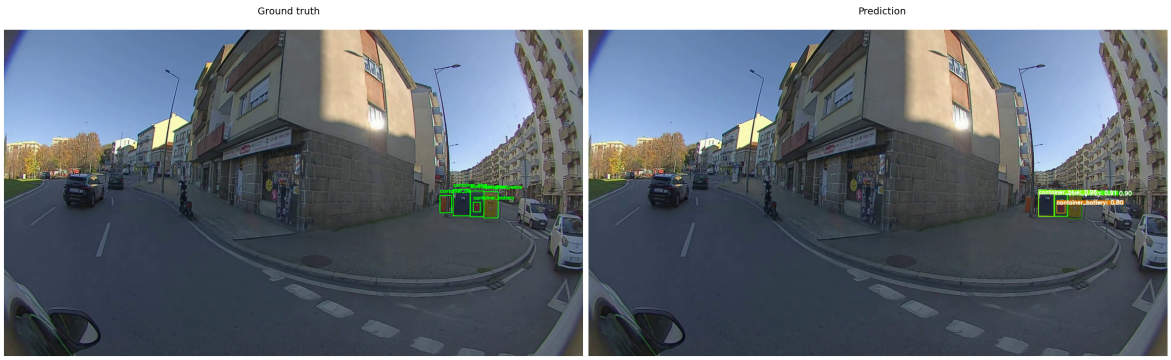

Figure 4.14: DiffusionVID [7] Scenario 4.

Figure 4.15: DiffusionVID [7] Scenario 5.

Figure 4.16: DiffusionVID [7] Scenario 6.

4.2.2.3 Limitations Analysis

DiffusionVID [7] exhibits limitations in some scenarios involving overlapping objects, where the model tends to merge closely placed objects, and in cluttered scenes, where containers are not detected.
In contrast, the figures illustrated in Section 4.2.1.2 highlight the model ability to effectively separate overlapping containers and detect the missing container in the cluttered scene. This performance is achieved through its refined NMS and the multi-scale feature representation approach discussed in Section 2.2.1.3.

Figure 4.17: DiffusionVID [7] Scenario 1.

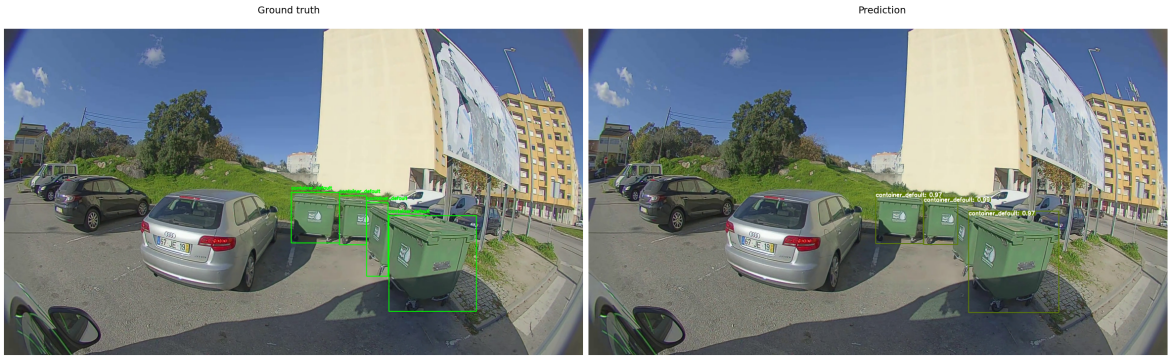

Figure 4.18: DiffusionVID [7] Scenario 2.

4.2.3 Analysis of Image- vs. Video-Based Detection on the Base Dataset

The precision-recall analysis reveals performance differences between the video-based detector [7] and the image-based detector [10]. For the green, blue, and default containers, both detectors perform similarly. However, the video-based detector [7] exhibits a significant drop in performance for the yellow container. The yellow container often appears white due to lighting conditions, particularly depending on the perspective, which complicates its detection. While the video-based detector [7] is theoretically better suited for handling such lighting variations and occlusions, its performance on this container is hindered, likely due to the dataset's limited representation of these challenging conditions. A more diverse dataset, with broader lighting scenarios and perspectives, could potentially enhance the video-based detector's [7] ability to detect the yellow container effectively.
For the underrepresented oil container, both detectors demonstrate similar performance, with the image-based detector [10] slightly outperforming the video-based detector [7] in terms of AP. This aligns with the container's smaller size and its underrepresentation in the dataset, which limits the detectors' ability to generalize. In contrast, the video-based detector [7] performs better on the battery container, the smallest of the objects, suggesting that it is more adept at detecting small objects compared to the image-based detector [10].
Overall, the image-based detector [10] outperforms the video-based detector [7] in terms of overall AP. However, the video-based detector [7] shows promise, particularly in handling smaller objects. Thus, while the image-based detector [10] provides a stronger baseline, the advantages of the video-based detector [7] in handling complex video object detection scenarios warrant further exploration, especially with an expanded dataset that incorporates a wider range of conditions.

4.3 Dataset Augmentation

As discussed in Section 4.2.2.3, model performance is sensitive to object scale, class representation, and scene complexity. These observations highlight the limitations of the base dataset, particularly the imbalance across container classes.
To address this, the dataset was significantly expanded with new video sequences, segmented into clips featuring a broad range of container types and background scenes. While the initial goal was to mitigate class imbalance especially for oil, battery, and biodegradable containers, the added data also increased overall dataset size and variability.
This augmentation introduces more representative object appearances, including different poses, occlusions, and environmental conditions.

4.3.1 Dataset Growth Statistics

This section quantifies the progression from the first stage (base) to the current stage (augmented with additional videos) of the dataset. Figure 4.19 illustrates the increase in both image and instance counts. The total number of images grew from 19,826 to 29,248 (+47.5%), while the number of container instances rose more significantly, from 22,840 to 34,407

(+50.6%). Notably, the test split experienced a 71.8% increase in container instances, rising from 4,311 to 7,405.

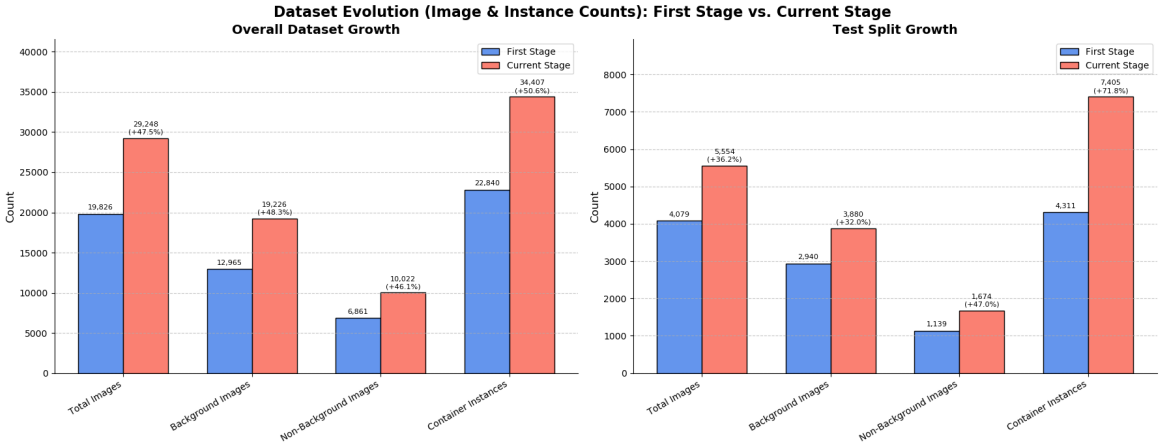

Figure 4.19: Dataset scaling: Comparison of total images, background/non-background images, and container instance counts. The augmentation significantly increased data volume for both overall training (left) and test evaluation (right).

The diversity of visual scenarios was expanded through the inclusion of new video sequences, as shown in Figure 4.20. The number of videos featuring containers increased from 53 to 93 (+75.5%) overall, diversifying environmental conditions, lighting, and container configurations.

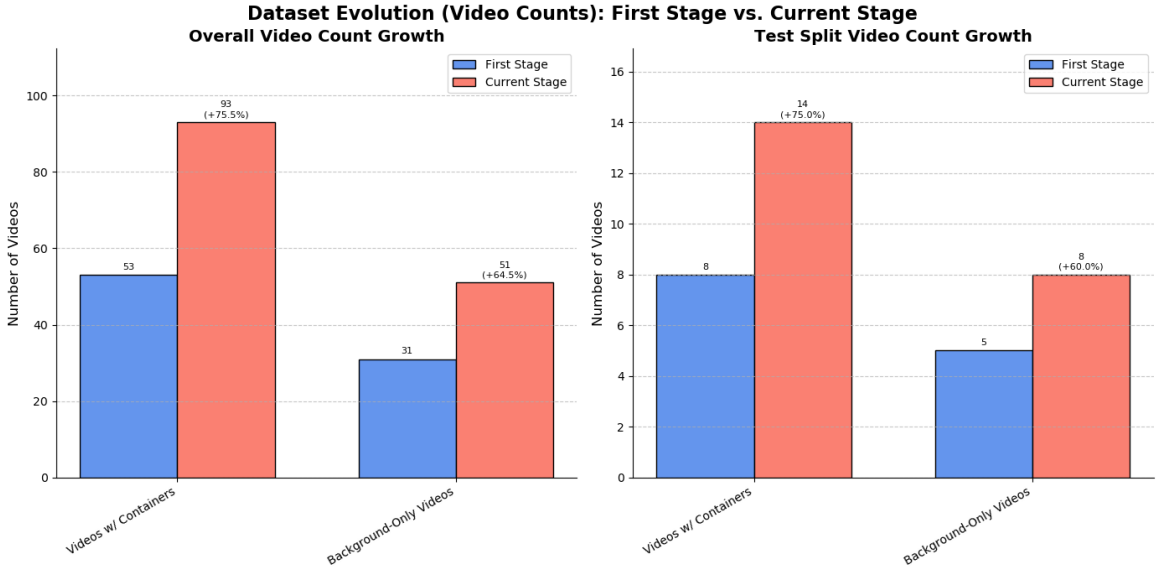

Figure 4.20: Video content expansion: Increase in the number of unique videos containing containers and background-only scenes, reinforcing scenario diversity for both overall data (left) and the test split (right).

Temporal context was also enriched, as quantified in Figure 4.21. Container presence time (at 10 FPS) increased by 46.1% overall (11.26 to 16.42 mins) and 47.0% in the test split (1.54 to 2.47 mins). This increase in temporal data provides more sequences for video-based analysis method [7], despite the frame-based detection approach [10].

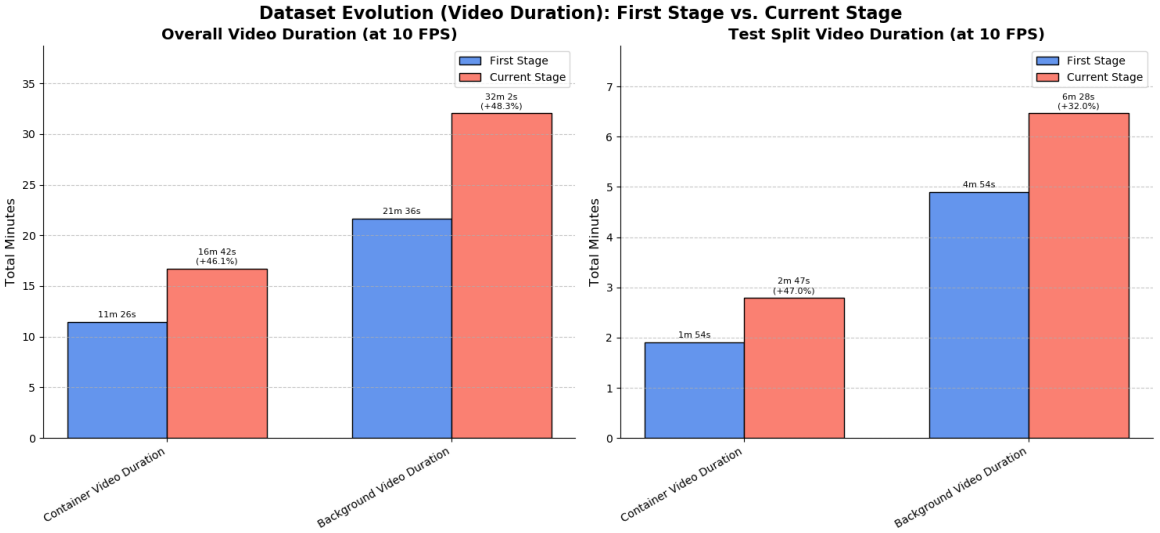

Figure 4.21: Temporal data augmentation: Growth in total video minutes for container-present and background-only footage (10 FPS), contributing to valuable temporal context overall (left) and in the test split (right).

A primary goal of the augmentation was to mitigate class imbalance. Figure 4.22 demonstrates considerable progress. For instance, the ‘container_ash‘ class grew from 0 to 196 instances, ‘container_battery‘ from 523 to 1,294 (+147.4%), and ‘container_oil‘ from 731 to 890 (+21.8%). Even well-represented classes like ‘container_default‘ saw a 43.2% increase.

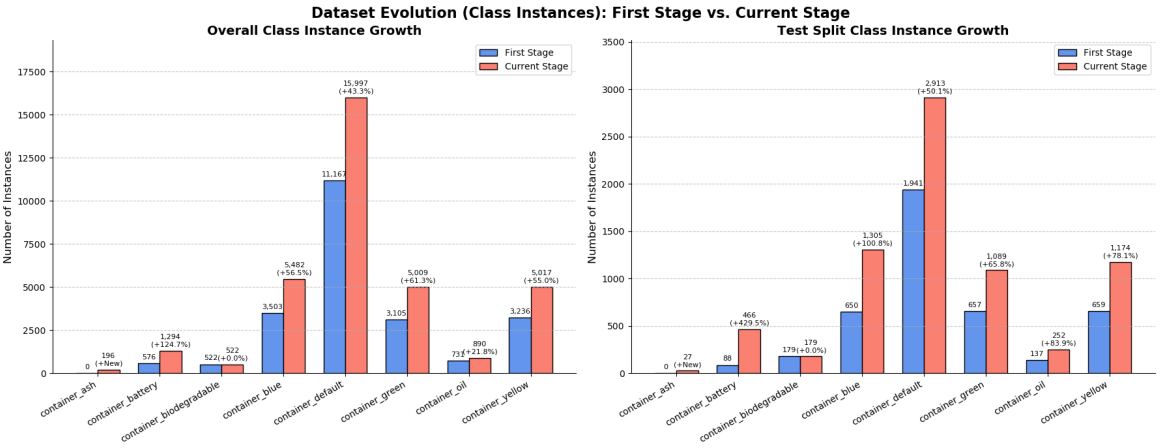

Figure 4.22: Class distribution enhancement: Instance count growth per container class, illustrating efforts to mitigate imbalance for both the overall dataset (left) and the test split (right). This improves learning for rare classes.

Finally, to support object tracking, we increased the density of unique object tracks. Figure 4.23 shows a 73.9% rise in overall unique container tracks (218 to 379). Per-class track counts (left panel) also show substantial gains, e.g., ‘Blue‘ (+75.0%), ‘Default‘ (+69.4%), and a relative increase for ‘Ash‘.

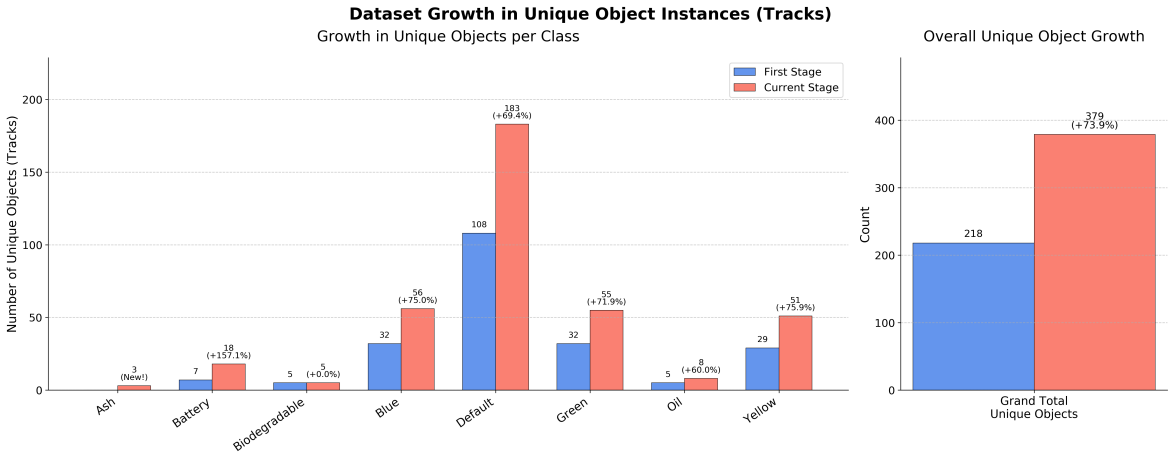

Figure 4.23: Unique object track augmentation: Growth in the number of distinct tracked container instances. Per-class track increases (left) and overall unique object growth (right) support improved Re-ID learning and tracking evaluation.

In summary, the augmentation provides a dataset substantially larger in volume, richer in visual and temporal diversity, more balanced in class distribution, and denser in unique object tracks.

4.3.2 Impact on Detection Performance

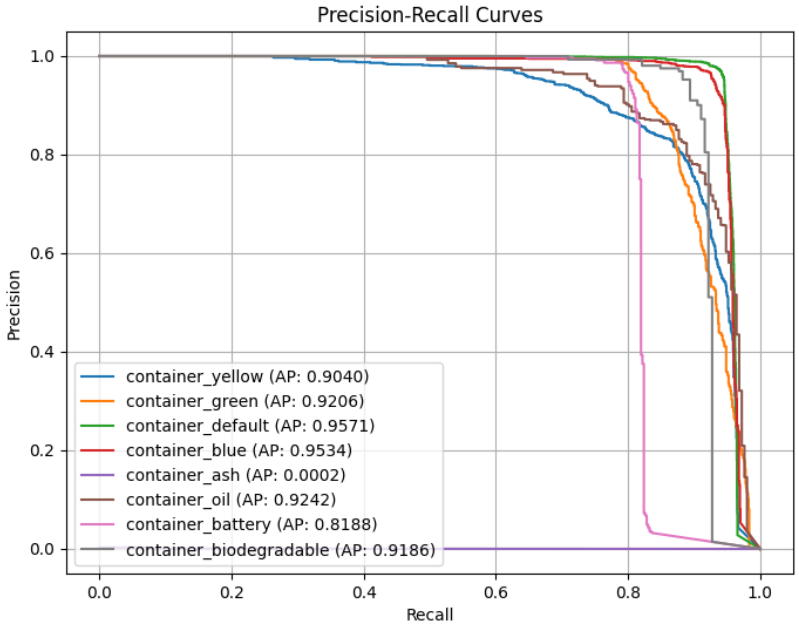

Figure 4.24: Precision-Recall curve for DiffusionVID [7] after training on the augmented dataset.

In order to assess the effect of data augmentation through additional videos (presented in Section 3.2), we compare the PR curves of YOLOv11 [10] (depicted in Figure 4.25) and DiffusionVID [7] (illustrated in Figure 4.24) models trained on the augmented dataset. Baseline results are reported in Figure 4.10 and Figure 4.1.

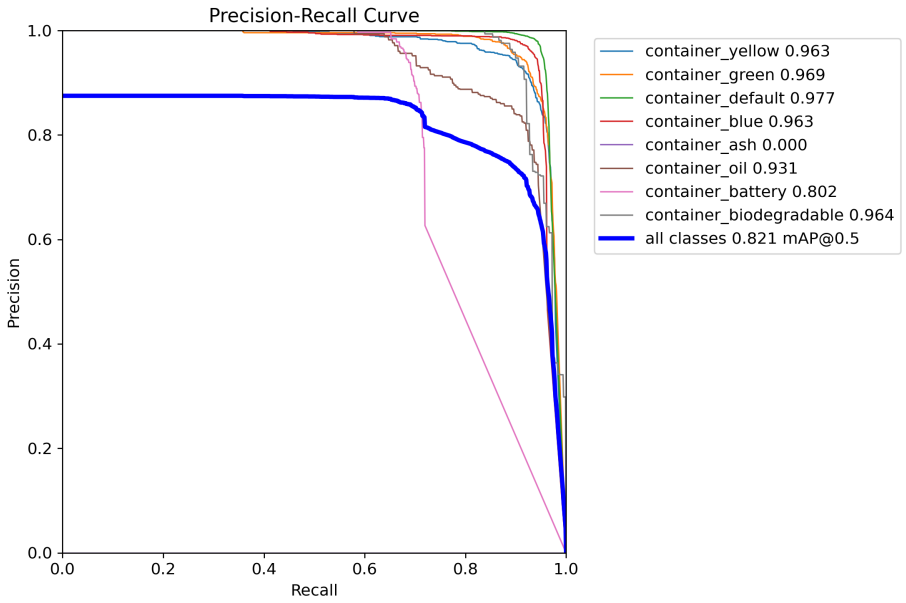

Figure 4.25: Precision-Recall curve for YOLOv11 [10] after training on the augmented dataset.

After augmentation, YOLOv11 [10] achieved a mAP@0.5 of 0.938, while DiffusionVID [7] reached 0.913, excluding the underrepresented class ash. YOLOv11 [10] outperformed DiffusionVID [7] across nearly all classes, with the notable exception of battery. This class contains the smallest objects in the dataset, reinforcing observations from Section 4.2.2.3 that DiffusionVID [7] may have greater sensitivity to small-scale targets.
The augmentation did not result in the expected performance improvement of DiffusionVID [7] over YOLOv11 [10], despite its theoretical advantage in handling occlusions.

4.3.3 Decision Based on Results

In addition to detection performance, computational overhead was an important consideration, as the models are intended to run on a Jetson Nano. YOLOv11 [10] requires fewer resources than DiffusionVID [7], which makes it a better fit for edge deployment where hardware is limited.
Taking both accuracy and practical constraints into account, YOLOv11 [10] was chosen as the primary detection model for the rest of the system.

4.4 Tracker Heuristic's Evaluation

We evaluate the proposed post-processing heuristics (described in Section 3.3) using the metrics defined in Section 3.3, specifically MAE and SAD. The objective is to identify a heuristic or combination of heuristics that consistently minimizes both metrics under varying operational conditions.
The experiments are conducted in three stages: (1) a controlled setting using the initial base

dataset, (2) a more challenging scenario with an augmented version of the dataset, and (3) real-world conditions using two raw approximately 2-hour videos recorded from a garbage collection vehicle.
Ultimately, we aim to find a configuration that not only reduces MAE and SAD in each dataset stage but also maintains strong performance across both real-world video sequences. Achieving minimal error in both cases would indicate the configuration's suitability for practical deployment.

4.4.1 First Stage: Base Dataset

The test set comprises 1 minute and 54 seconds of visible containers and 2 minutes and 47 seconds of background-only scenes (depicted in 4.21), totaling over 5 minutes of video data exposure. This distribution provides the baseline evaluation scenario for our method.
Table 4.5 summarizes the results on the base dataset. H1 provides a strong initial reduction in error. Adding H2 further improves MAE. H3 slightly increases MAE.

Table 4.5: Overall evaluation across heuristic configurations on the first-stage dataset (base). Note: H2 and H3 are only applied in conjunction with H1. SAD is the total absolute error, while $Pred - GT$ is the signed difference, showing bias (positive = overestimation).

Method	**MAE** ↓	**SAD** ↓	$Pred - GT$
Baseline (ByteTrack)	2.00	16	16
+ H1	0.75	6	6
+ H2	0.38	3	1
+ H3	0.50	4	4

4.4.2 Second Stage: Dataset Augmented with Additional Videos

The test set comprises 2 minutes and 47 seconds of visible containers and 6 minutes and 28 seconds of background-only scenes (depicted in Figure 4.21), totaling over 9 minutes of video data exposure.
In Table 4.6, we evaluate the heuristics on a more challenging augmented dataset. Although the baseline MAE increases substantially, the full set of heuristics (H1 + H2 + H3) reduces the MAE from 3.41 to 0.77 and significantly narrows the Pred - GT gap. This demonstrates the robustness and generalization of our approach under more varied conditions.

Table 4.6: Overall evaluation across heuristic configurations on the second-stage dataset (augmented). Note: H2 and H3 are only applied in conjunction with H1. SAD is the total absolute error, while $Pred - GT$ is the signed difference, showing bias (positive = overestimation).

Method	**MAE** ↓	**SAD** ↓	$Pred - GT$
Baseline (ByteTrack)	3.41	75	63
+ H1	1.09	24	0
+ H2	0.86	19	-17
+ H3	0.77	17	-11

4.4.2.1 Impact of Heuristics on Tracking Quality

Experimental results demonstrate that the proposed heuristics substantially improve tracking accuracy. On the base dataset, combining H1 and H2 yields the lowest error, reducing MAE by 81% and SAD by 81.25% relative to the baseline. While H3 does not contribute positively in this setting, it becomes essential in the augmented dataset, where its inclusion leads to the best overall performance. Specifically, H1+H2+H3 achieves a 77.4% reduction in MAE and a 77.3% drop in SAD compared to the baseline.

4.4.3 Heuristic Evaluation Using Raw 2-Hour Vehicle Camera Footage

In the final stage of the experimental pipeline, the effectiveness of the proposed post-processing heuristics is evaluated using two 2-hour videos recorded from the garbage collection vehicle.

4.4.3.1 Video 1 (Right)

The duration consisted of 1 hour, 41 minutes, and 7 seconds for the background, and 14 minutes and 38 seconds for the containers.

Table 4.7: Overprediction analysis of the baseline tracker on Video 1. A significant surplus of container tracks is observed.

Class Name	GT Count	Pred. Count	Difference
container_yellow	40	127	+87
container_green	37	146	+109
container_default	139	275	+136
container_blue	38	137	+99
container_ash	8	6	-2
container_oil	8	33	+25
container_battery	6	15	+9
container_biodegradable	3	5	+2
Total	**279**	**744**	**+465**

In Table 4.7, the baseline tracker significantly overcounts containers across all major classes. The SAD results shown in Figure 4.26 reflect that the best results are achieved with H1 + H2, outperforming both individual heuristics and the full combination.

4.4.3.2 Video 2 (Left)

The duration consisted of 1 hour, 42 minutes, and 4 seconds for the background, and 12 minutes and 41 seconds for the containers.

In Table 4.8, we observe a consistent overcounting trend across all container classes, though the total overprediction is slightly less severe than in Video 1. However, the SAD results shown in Figure 4.27 reveal a divergent heuristic performance pattern. In contrast to Video 1, the best results are obtained by applying identity merging (H2) alone, followed closely by temporal filtering (H1). Interestingly, the combined heuristic H1 + H2, previously optimal in Video 1, ranks fourth in this setting.

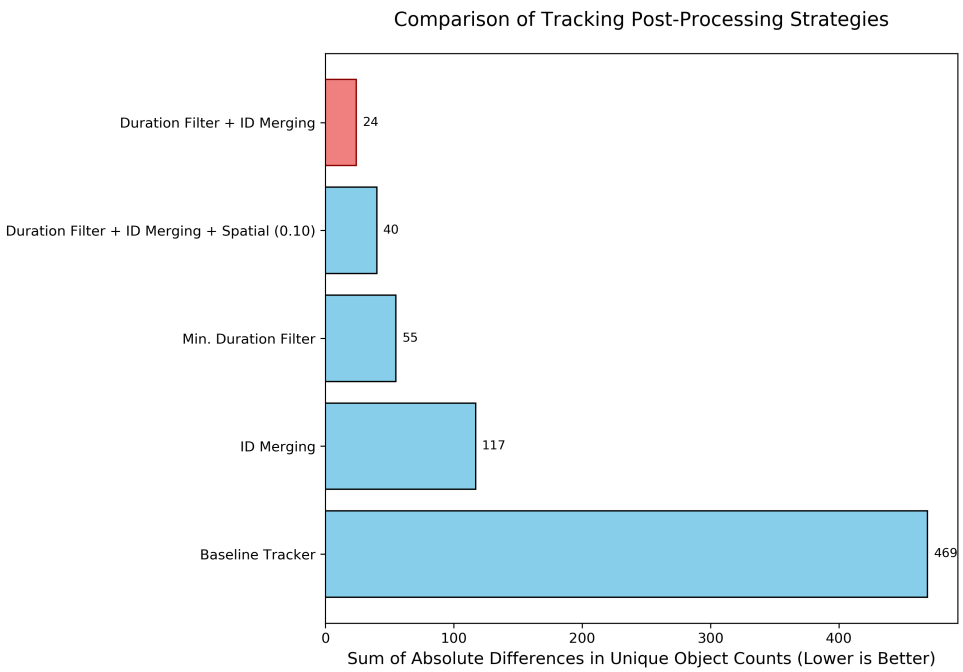

Figure 4.26: Impact of post-processing heuristics on SAD for Video 1 (Right). The combination H1 + H2 achieves the lowest SAD.

Table 4.8: Overprediction analysis of the baseline tracker on Video 2. While inflated predictions remain, the overcount is slightly lower than in Video 1.

Class Name	GT Count	Pred. Count	Difference
container_yellow	39	112	+73
container_green	30	135	+105
container_default	110	275	+165
container_blue	34	110	+76
container_ash	6	7	+1
container_oil	7	23	+16
container_battery	5	6	+1
container_biodegradable	2	4	+2
Total	**233**	**672**	**+439**

4.4.3.3 Identifying the Most Generalizable Heuristic Combination

No single heuristic consistently outperforms across all evaluations. In Video 1, the combination of temporal smoothing and identity unification (H1 + H2) ranks first, followed by the full combination (H1 + H2 + H3). In Video 2, identity merging alone (H2) achieves the best results, while the full combination ranks third.

In contrast, evaluation on the augmented second-stage dataset (Section 4.4.2) demonstrates that the full heuristic combination H1 + H2 + H3 delivers the best overall performance across both MAE and SAD metrics.

This divergence indicates that although H1 + H2 + H3 provides the most consistent improvements on the augmented dataset, its advantage does not fully extend to real-world videos, where simpler heuristics outperform the full combination.

Therefore, determining a universally optimal heuristic based solely on these results remains challenging. Nevertheless, H1 + H2 + H3 offers the most balanced configuration, making it

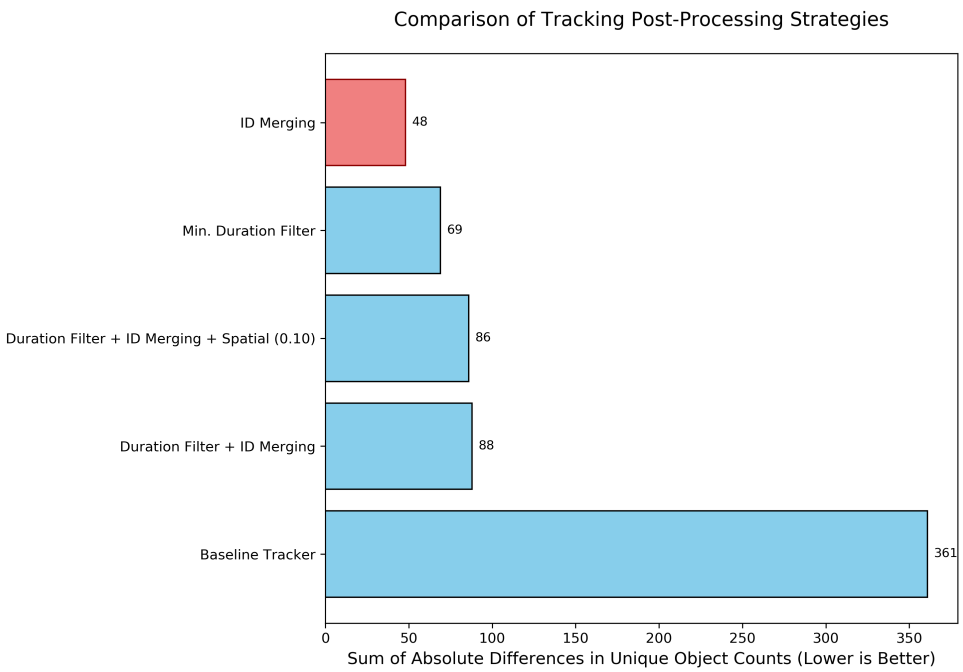

Figure 4.27: Impact of post-processing heuristics on SAD for Video 2 (Left). Here, H2 (identity merging) alone yields the lowest SAD, with H1 (temporal filtering) close behind.

the preferred default choice for deployment in realistic, varied scenarios.

4.5 Summary and Conclusions

This chapter presented an experimental evaluation of waste container detection and tracking using computer vision techniques, exploiting both single-frame and video-based models. Two detection approaches were analyzed in detail: YOLOv11 [10], a state-of-the-art image-based detector, and DiffusionVID [7], a diffusion-based video object detection VOD method. The evaluation began in a custom dataset, which includes 84 videos (33:02m) and over 22,000 annotated bounding boxes. YOLOv11 [10] achieved strong baseline performance, with a mean Average Precision (mAP)@0.5 of 0.912. It exhibited robustness across most container classes, particularly those with higher representation in the dataset. However, limitations were noted in detecting small or distant objects, especially the battery container class. Qualitative assessments also revealed challenges in over-detection and performance drops at greater object distances.

In contrast, DiffusionVID [7] demonstrated competitive accuracy. Notably, it outperformed YOLOv11 [10] in detecting the battery container class. Nevertheless, its overall detection performance was lower than that of YOLOv11 [10]. Augmenting the dataset led to improved results for both models, with YOLOv11 achieving a mAP@0.5 of 0.938 and DiffusionVID reaching 0.913. However, DiffusionVID did not gain the expected advantage from augmentation, likely due to persistent data diversity issues. As a result, YOLOv11 [10] was selected as the primary detection model due to its better balance between detection accuracy and computational efficiency—particularly relevant for deployment on edge devices such as the

Jetson Nano.

To improve temporal consistency and object tracking, a ByteTrack-based [12] pipeline was developed and extended with a series of post-processing heuristics. These included: (1) H1, for eliminating short-lived tracks; (2) H2, for merging fragmented tracks based on temporal continuity; and (3) H3, for enforcing spatial consistency between consecutive detections. These heuristics significantly improved tracking reliability. When applied to the base dataset, H1+H2 reduced MAE by over 80%. On the augmented dataset with additional videos, the combination H1+H2+H3 delivered the best overall performance. In real-world waste collection videos, heuristic effectiveness varied, but the full set of heuristics proved most robust across different contexts.

In summary, YOLOv11 [10] was confirmed as the most suitable detector for this application. DiffusionVID [7] remains a promising alternative for scenarios involving occlusions, with further potential to be unlocked through dataset expansion—particularly with the inclusion of varied conditions such as nighttime scenes. The tracking pipeline, complemented by heuristic strategies, demonstrated strong adaptability and robustness, paving the way for effective deployment in urban waste collection systems.

Chapter 5

Conclusion and Future Work

5.1 Main Contributions

This dissertation introduced a computer vision pipeline for automatic detection and tracking of urban waste containers, with the goal of supporting smart city waste management through scalable, real-time video analytics.

A major contribution of this work is the creation of a custom video dataset that reflects the challenges of waste container detection in real-world urban environments. The dataset comprises over 29,000 frames, including 10,022 fully annotated images containing 34,407 container instances (containers: 16:42m; background: 32:02; total: 48:44m).

Through a augmentation process that incorporated additional video sequences, the dataset was expanded to include more diverse scenarios involving containers, noncontainers, and rare container classes. This led to a 73,9% increase in the number of unique tracked objects from 218 to 379 and significantly improved overall model performance across all classes for both detectors: the video-based detector [7] and the image-based detector [10].

Two state-of-the-art object detection methods, the image-based detector [10] and the video-based detector [7], were trained and evaluated on both the base and augmented datasets. The image-based detector [10] outperformed the video-based detector [7] in terms of overall detection performance on both dataset versions, achieving a mAP@0.5 of 0.938 after training on the augmented dataset, compared to 0.913 for the video-based detector [7]. While the video-based detector [7] showed some advantages in detecting small containers (for example, batteries), the image-based detector [10] was ultimately selected due to its superior accuracy, efficiency, and suitability for resource-constrained applications (for example, Jetson Nano).

To support robust object tracking, the image-based detector [10] was integrated with ByteTrack [12] and further enhanced with a set of post-processing heuristics. These heuristics, namely temporal filtering of short-lived tracks (H1), temporal merging of fragmented identities (H2), and spatial consistency enforcement (H3), were evaluated. On the base dataset, the combination of H1 and H2 reduced the mean absolute error (MAE) in urban waste container counting from 2.00 to 0.38 and the sum of absolute differences (SAD) from 16 to 3. On the augmented dataset, the full heuristic pipeline (H1 plus H2 plus H3) reduced MAE from 3.41 to 0.77 and SAD from 75 to 17, representing relative improvements of 77.4% and 77.3%, respectively, in the accuracy of container counts.

The final system was validated on two real-world video sequences spanning approximately two hours each. Initial overpredictions by the baseline tracker were substantially mitigated through heuristic tuning. In Video 1, the combination of temporal filtering and identity merging heuristics (H1 plus H2) reduced the number of extra container tracks from 469 to 24, representing a reduction of approximately 94.9%. In Video 2, identity merging alone (H2)

decreased the overpredictions from 361 to 48, a reduction of about 86.7%.
Therefore, determining a universally optimal heuristic based solely on these results remains challenging. Nevertheless, the full combination H1 plus H2 plus H3 offers the most balanced configuration and is considered the preferred default choice for deployment in realistic, varied scenarios.
In summary, for the base dataset, the selected heuristic H1 plus H2 plus H3 reduces MAE from 2.00 to 0.50 and SAD from 16 to 4, corresponding to reductions of 75% and 75%, respectively. For the augmented dataset, the heuristic pipeline reduces MAE from 3.41 to 0.77 (a 77.4% reduction) and SAD from 75 to 17 (a 77.3% reduction).
Regarding the real-world videos, the chosen heuristic decreases the SAD by approximately 94.9% in Video 1 and 86.7% in Video 2, demonstrating its effectiveness in complete waste collection routes.
Altogether, this work delivers a deployable and modular solution for visual waste container detection and tracking. The system is lightweight and designed with edge deployment in mind, potentially allowing execution on devices such as the Jetson Nano. Its modular design also enables future integration with georeferenced mapping and route optimization tools. Through this contribution, the thesis advances the practical application of computer vision in municipal services and supports the broader goal of sustainable, smart urban infrastructure.

5.2 Future Work

Future work will focus on addressing the performance gap observed in underrepresented classes such as battery and ash containers. Although these classes achieve AP values above 0.93 using the image-based detector [10] on the augmented dataset (illustrated in Section 4.25), it remains uncertain whether further data augmentation or scaling of the training set will lead to meaningful improvements. To investigate this without additional data collection, we plan to adopt a progressive subsampling strategy. By training the model on increasingly larger subsets of the current dataset while keeping the test set fixed, we aim to generate learning curves that reveal whether model performance is saturating or still has room to improve.
On the tracking side, we will conduct a qualitative analysis of failure cases under the current heuristic configuration (H1 + H2 + H3). Through visual inspection of scenarios involving ID switches or fragmentations, we intend to identify recurring patterns and design targeted heuristics to address those specific issues.
While the current evaluation is based on two real-world video sequences, deploying the solution in the company's operational waste collection systems will ultimately yield a more reliable and realistic estimate of its performance under practical conditions.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2016, accessed: 2024-09-30. [Online]. Available: https://pjreddie.com/yolo/ xiii, xv, 5, 6, 7, 8, 12, 29

[2] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *arXiv preprint arXiv:2005.12872*, 2020. [Online]. Available: https://arxiv.org/abs/2005.12872 xiii, xv, 6, 9, 10, 18, 20

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *SSD: Single Shot MultiBox Detector*. Springer International Publishing, 2016, p. 21–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2 xiii, 5, 11, 29

[4] R. Girshick, "Fast r-cnn," 2015. [Online]. Available: https://arxiv.org/abs/1504.08083 xiii, 4, 11, 12, 13, 14, 15

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018. [Online]. Available: https://arxiv.org/abs/1703.06870 xiii, xv, 12, 15, 16, 17, 18, 29

[6] H. Wang, J. Tang, X. Liu, S. Guan, R. Xie, and L. Song, "Ptseformer: Progressive temporal-spatial enhanced transformer towards video object detection," 2022. [Online]. Available: https://arxiv.org/abs/2209.02242 xiii, xv, 4, 18, 19, 20, 21, 29

[7] S.-D. Roh and K.-S. Chung, "Diffusionvid: Denoising object boxes with spatio–temporal conditioning for video object detection," *IEEE Access*, vol. 11, pp. 121 434–121 444, 2023. xiii, xiv, xv, xvi, 3, 4, 21, 22, 24, 29, 32, 39, 44, 45, 46, 47, 48, 49, 51, 52, 56, 57, 59

[8] Y. Shi, N. Wang, and X. Guo, "Yolov: Making still image object detectors great at video object detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, pp. 2254–2262, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25320 xiii, xv, 24, 25, 26

[9] Y. Shi, T. Zhang, and X. Guo, "Practical video object detection via feature selection and aggregation," 2024. [Online]. Available: https://arxiv.org/abs/2407.19650 xiii, xv, 25, 26, 29

[10] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," 2024. [Online]. Available: https://arxiv.org/abs/2410.17725 xiv, xv, xvi, 3, 4, 7, 8, 9, 31, 32, 33, 34, 37, 39, 41, 42, 43, 44, 45, 48, 49, 51, 52, 56, 57, 59, 60

[11] W. Wang, "The development of face recognition in accuracy and speed: A review," in *2021 2nd International Conference on Computing and Data Science (CDS)*, 2021, pp. 79–89. xv, 13, 14

[12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," 2022. [Online]. Available: https://arxiv.org/abs/2110.06864 xv, 1, 3, 4, 27, 28, 29, 31, 33, 34, 37, 39, 57, 59

[13] Y. Gladiensyah Bihanda, C. Fatichah, and A. Yuniarti, "Multi-vehicle tracking and counting framework in average daily traffic survey using rt-detr and bytetrack," *IEEE Access*, vol. 12, pp. 121 723–121 737, 2024. xv, 27

[14] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sep. 2016. [Online]. Available: http://dx.doi.org/10.1109/ICIP.2016.7533003 3, 4, 27, 28, 29

[15] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649. 3, 4, 28

[16] D. Shyam, A. Kot, and C. Athalye, "Abandoned object detection using pixel-based finite state machine and single shot multibox detector," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6. 4

[17] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *ArXiv*, vol. abs/2206.14651, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250113384 4, 28

[18] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750. 5

[19] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: A simple and strong anchor-free object detector," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 4, pp. 1922–1933, 2020. 5

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018. [Online]. Available: https://arxiv.org/abs/1708.02002 5

[21] A. Tripathi, C. Srivastava, S. K. Pandey, M. K. Gupta, and P. Dixit, "Object detection using yolo: A survey," in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2022, p. 747. 6

[22] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016, accessed: 2024-09-30. [Online]. Available: http://pjreddie.com/yolo9000/ 7

[23] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018, accessed: 2024-09-30. [Online]. Available: https://arxiv.org/abs/1804.02767 7

[24] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020, accessed: 2024-09-30. 7

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016. [Online]. Available: https://arxiv.org/abs/1506.01497 11, 13, 14, 15, 17, 29

[26] T.-Y. Lin, M. Ma, S. Girdhar, P. Dollár, R. B. Girshick, K. He, B. Singh, S. Xie, and A. Farhadi, "Microsoft coco: Common objects in context," 2014, accessed: 2024-10-07. [Online]. Available: https://cocodataset.org/#explore 12

[27] G. Bhattrai, "Pascal voc 2012 dataset," https://www.kaggle.com/datasets/gopalbhattrai/pascal-voc-2012-dataset, 2017, accessed: 2025-01-27. 12

[28] R. Pandey and A. Malik, "Object detection and movement prediction for autonomous vehicle: A review," in *2021 Second International Conference on Secure Cyber Computing and Communication (ICSCCC)*. NIT Jalandhar, India: IEEE, 2021, pp. 60–65. 12

[29] A. Rosebrock, "Object detection: Bounding box regression with keras, tensorflow, and deep learning," https://pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/, 2020, accessed: 2024-09-30. 14

[30] W. Han, P. Khorrami, T. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. Huang, "Seq-nms for video object detection," in *arXiv preprint arXiv:1602.08465*, 2016. 18

[31] H. Belhassen, H. Zhang, V. Fresse, and E. Bourennane, "Improving video object detection by seq-bbox matching," in *VISIGRAPP (5: VISAPP)*, 2019. 18

[32] C. Deng, D. Chen, and Q. Wu, "Identity-consistent aggregation for video object detection," in *ICCV*, 2023. 18

[33] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *ICCV*, 2019. 18

[34] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei, "Relation distillation networks for video object detection," in *ICCV*, 2019. 18

[35] Q. Zhou, X. Li, L. He, Y. Yang, G. Cheng, Y. Tong, L. Ma, and D. Tao, "Transvod: End-to-end video object detection with spatial-temporal transformers," vol. 45, no. 6, 2023, pp. 7853–7869. 18

[36] G. Sun, Y. Hua, G. Hu, and N. Robertson, "Efficient one-stage video object detection by exploiting temporal consistency," in *ECCV*, 2022. 18

[37] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021. [Online]. Available: https://arxiv.org/abs/2107.08430 24, 25

[38] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552 27, 28

[39] Evox, “Evox - site oficial,” https://evox.pt/, accessed: 2025-01-11. 31, 39