POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Artificial Intelligence and Data Analytics

Master Degree Thesis

Developing Vehicular Traffic Datasets Using Drone Images and AI Techniques

Supervisors
Prof. CLAUDIO ETTORE CASETTI
Diego Gasco and Giuseppe Perrone

Candidate
Samaneh Gharehdagh Sani

Academic Year 2024-2025

Acknowledgements

I want to thank everyone who has helped me in my academic career. Above all, I am very grateful to my supervisor, Prof. Claudio Ettore Casetti, for his support and guidance during this thesis. I also appreciate Giuseppe Perrone and Diego Gasco for their helpful discussions and kind assistance, their guidance was very important in the success of this project. I would like to thank my family and friends for their endless love and encouragement. A very special thanks goes to Koorosh, whose support and faith in me have been my motivation over these past years, for which I am truly grateful.

Abstract

Following the growth of cities, we are facing more traffic, pollution, and longer drives. Because urban road networks are complex and traffic conditions are changing, managing this traffic is difficult. Although there is the possibility of artificial intelligence as a solution, many current AI-based traffic systems are having trouble running consistently. The authors relied on data sets that were collected at fixed points, in stable environments, and updated infrequently. It is challenging for models to generalize to new cities, unusual incidents, or unexpected changes because such data sets cannot capture the diversity of real-world environments.

This thesis describes a pipeline for using drone video and creating reliable urban data sets. As drones are easy to move and relocate, their wide coverage and adjustable viewing angles allow them to record traffic situations in different environments. After collecting data using drone, an AI based pipeline that combines object detection, multi-object tracking, and speed estimation using calibration techniques processes the captured video.

The study compares the advantages and disadvantages of different tracking frameworks by evaluating detection and tracking performance using different drone altitudes and viewing angles. It also looks at how accuracy changes with altitude and distance. Urban planning strategies, autonomous vehicle intelligence, safety evaluations, and the creation of more precise AI-based traffic models are all supported by the resulting datasets and processing pipeline. A strong calibration technique that reliably translates pixel displacements to actual speed measurements is one of the main contributions of this work. The resulting data sets are highly detailed and completely verified, offering a strong base for developing intelligent and flexible traffic control systems.

Contents

Chapter 1

Introduction

Transportation networks experience new difficulties because urban areas grow at a rapid rate. The increasing number of vehicles in cities causes poorer air quality, more traffic, and longer travel times [41]. Traditional traffic monitoring systems that use fixed road sensors, including loop detectors and stationary cameras, fail to detect changes in traffic patterns because their monitoring areas are limited [32, 59]. Datasets that support AI-based traffic models, including NGSIM and UA-DETRAC collect data from fixed points in controlled environments with limited updates, creating challenges for models to handle different locations and unexpected situations [34, 53, 54]. The current systems cannot provide reliable adaptive traffic intelligence because they cannot meet the requirements of modern urban transportation systems.

Artificial intelligence (AI) technology can be used as a modern solution to improve traffic management because it supports predictive modeling, adaptive control, and safety analysis functions. However, the performance of AI systems is based on the availability of high-quality datasets. The existing AI systems today were trained through datasets that come from controlled environments where they use fixed intersection cameras and stationary highway sensors and vehicles that operate within defined geographic areas. Training data receive updates only occasionally, while they remain restricted to particular geographic locations and weather and traffic conditions. Models that train on these data may not perform well when faced with new urban environments, diverse traffic conditions, and unexpected traffic patterns.

This thesis aims to address this limitation by developing a machine learning-based pipeline to extract vehicle movement data, that is, their positions and velocities, from videos recorded by a drone, making it possible for anyone to create their own dataset in any chosen scenario.

The developed pipeline connects object detection, multi-object tracking, and speed estimation functions to infer precise movement paths for recorded vehicles. Its operational effectiveness is demonstrated through an evaluation phase that tests performance at different drone heights, camera positions, and tracking settings. Also, different road layouts were considered: the system effectively handled vehicle movements, showing exceptional flexibility in different traffic conditions.

The findings of this thesis establish various important applications that extend beyond methodological enhancement. The developed pipeline produces accurate, validated datasets, and reliable real-world data from video recordings. The generated dataset enables improved AI-based traffic modeling and autonomous vehicle development, road safety evaluation, and science-based urban planning methods.

This thesis is structured as follows.

In Chapter 2 Related Work, the literature on traffic monitoring, data collection, object detection, multi-object tracking, and calibration techniques is reviewed.

Chapter 3 System Model and Contributions, outlines the main contributions of this work, highlighting the integration of the detection, tracking, and speed estimation modules, as well as the suggested calibration method.

In Chapter 4 Results, system results are reported that evaluate performance under different drone altitudes, viewing angles, and tracking frameworks.

Chapter 5 Conclusion summarizes the thesis and suggests potential works for further study.

Chapter 2

Related Work

Modern traffic systems can now be better understood and managed with the help of artificial intelligence (AI) technologies. Large amounts of visual data can be processed by AI systems to identify cars, track their movements, categorize various road users, and even predict traffic patterns. However, the quality and variety of the data used to train these systems have a significant impact on their performance. The flow of traffic in the real world is constantly changing due to accidents, road construction, weather, and special events. If not properly trained, AI models may not recognize and react to unexpected conditions, which could limit their application in critical situations. In addition, new data are needed to study the evolution of traffic and train in-vehicle algorithms.

In this area, a huge amount of work has produced benchmark datasets and task protocols that influence current engineering and research. Before moving on to the fundamental components of a drone video pipeline, this chapter reviews the main datasets, data collection, object detection, tracking, and calibration-based speed estimation.

2.1 From Traditional to AI-based Approaches

In order to measure traffic at a few locations, early systems used inductive loops, radar/microwave detectors, and fixed closed-circuit television (CCTV). These methods were strong, but provided limited information on individual trajectories and covered only restricted areas [41]. Computer vision techniques were able to extract counts and classes from static cameras as video spread, but performance was still affected by fixed viewpoints, blockages, and illumination [31]. Deep convolutional neural network (CNN)-based perception and time-based modeling for detection, classification, tracking, and prediction were introduced with the change to AI; however, generalization to new cities, camera geometries, and exceptional conditions depends on a variety of well-annotated datasets [11, 51]. In order to test models beyond fixed roadside cameras, the field has responded by investing in benchmark datasets with richer sensor suites (lidar, radar, multi-camera rigs) and UAV datasets that capture wide areas from various perspective points [15, 74].

2.2 Benchmark Datasets for Traffic Analysis

As models for this type of dataset that this thesis pipeline seeks to replicate, a few datasets stand out as standard references for accurate and trajectory-rich traffic data. These datasets establish the benchmarks for future methodological studies in addition to defining expectations for time resolution and annotation detail.

2.2.1 NGSIM

Some of the first large-scale vehicle trajectory datasets at lane level were created by the Next Generation Simulation (NGSIM) program [19,24,53,54]. The datasets, which were recorded using synchronized rooftop video systems, include main roads and boulevard paths like Lankershim and Peachtree, as well as highway sections like US-101 and I-80. The high period fidelity trajectories are sampled at 0.1 seconds (10 Hz), giving comprehensive records of velocity, speed, and lane position is a crucial feature. The detailed microscopic traffic analysis is made possible by the geographical extents, which include multiple lanes and range from 0.5 to 0.64 km. For many years, NGSIM has been used as a standard for research on trajectory prediction, lane change, and car follow.

2.2.2 highD

The highD dataset expanded on this idea by using UAVs to obtain near-roadside cameras' obstruction and coverage issues, [33, 34, 57]. HighD includes 16.5 hours of drone footage from six German highway locations, covering over 110,000 vehicles and 45,000 km, including around 5,600 full lane changes. The bird's-eye perspective reduces barriers while preserving realistic driving behavior, and recordings run for an average of about 17 minutes. The dataset includes lane geometries, vehicle classes, and trajectories with quality flags for all detected vehicles. Naturalistic driving models and safety analysis techniques have been validated thanks to this dataset.

2.2.3 nuScenes

NuScenes presents an integrated benchmark for autonomous driving, whereas NGSIM and highD focus on vehicle trajectories [11, 44]. NuScenes was recorded in Boston and Singapore and combines Global Positioning System (GPS)/Inertial Measurement Unit (IMU) sensors, radar, Light Detection and Ranging (LiDAR), and six cameras. The annotations, which cover 23 object classes with bounding boxes, track IDs, and map priors, are arranged into 20-second scenes with $2Hz$ key frames. Evaluation protocols for detection, tracking, motion forecasting, and planning have been impacted by its multi-sensor alignment and consistent scene structure. NuScenes has influenced labeling practices in the community and offers a template for organizing multi sensor datasets.

These three datasets, NGSIM, highD, and nuScenes, together form the main standards that this thesis aims to model in the approach. They establish guidelines for cross-scene consistency, annotation richness, and sampling frequency that guide the development of the suggested drone-based pipeline.

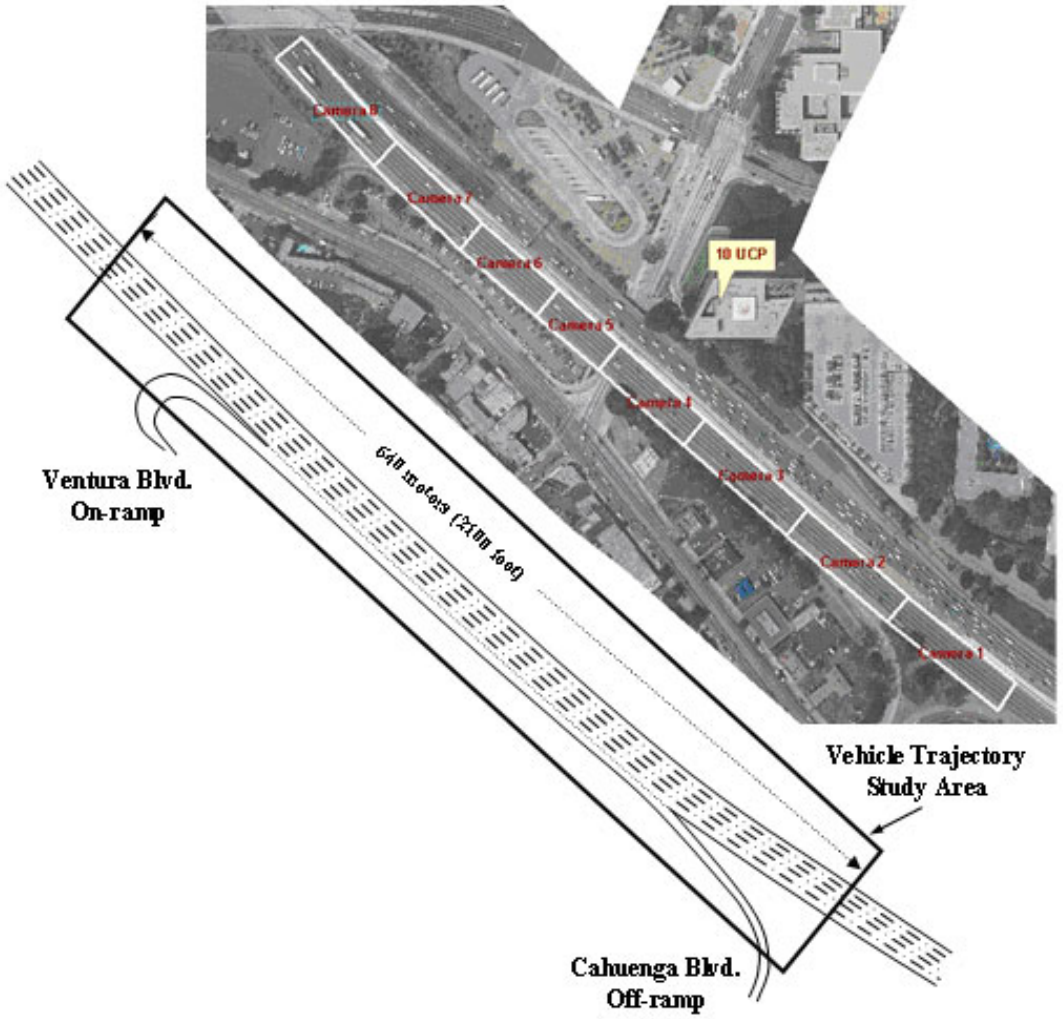


Figure 2.1: NGSIM US 101 study area: coverage and lane schematic. Adapted from FHWA Fact Sheet FHWA-HRT-07-030 [24].

2.2.4 Urban & interaction-rich aerial trajectory datasets

A collection of UAV-recorded datasets focuses on dense urban interactions at intersections, roundabouts, and merging areas, in addition to highway-focused corpora like highD. The *inD* dataset is a close urban analogue of highD [8], capturing roughly 10 hours of drone video at German urban intersections with over 11,500 annotated road users and rich interaction labels (yielding, stopping, crossing). With comprehensive lane geometries and gap-acceptance scenarios, its companion *rounD* focuses on multilane roundabouts [35]. To support reliable estimation of near-miss and surrogate safety measures, *openDD* aggregates more than 62 hours across multiple locations with consistent lane topology and long temporal windows, pushing scale at roundabouts [9].

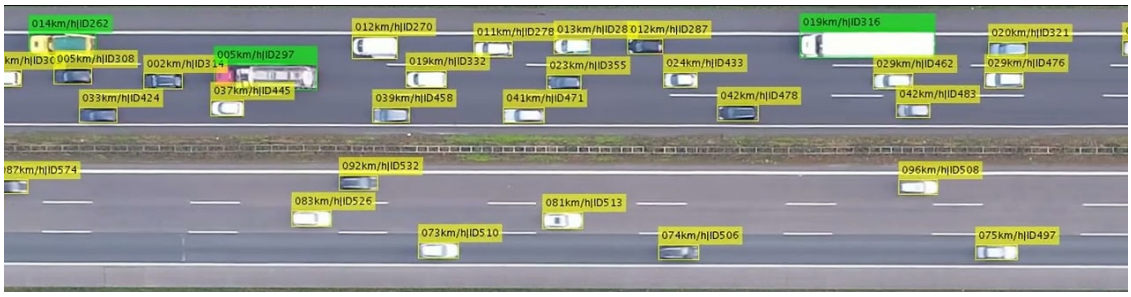


Figure 2.2: highD dataset: overhead UAV view with annotated vehicles, IDs and speeds. Adapted from the highD project page [33, 34].

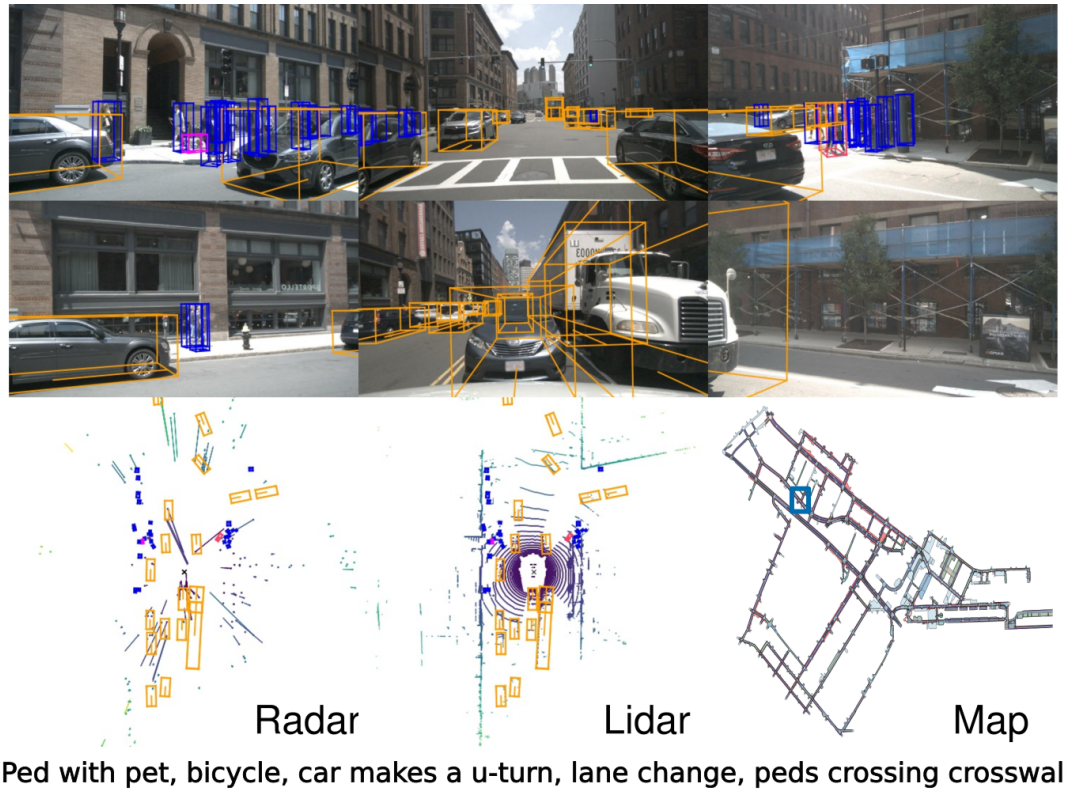


Figure 2.3: nuScenes multimodal example: synchronized camera images (top), radar and LiDAR point clouds (bottom left/middle), and HD map (bottom right). Adapted from Caesar *et al.* [11].

exiD targets the behavior of the entrance/exit of the highway, focusing on the on-ramp/off-ramp segments where lane changes, merges, and accelerations occur frequently [42]. More recent UAV releases broaden the set of scenario: *SIND* (suburban intersections with mixed traffic), *DRIFT* (diverse road geometries and traffic densities), and the multimodal drone corpus *MiTra* (vision plus auxiliary sensing) illustrate a trend towards larger, more varied, and interaction-centric UAV benchmarks [66]. Taken together, these

corpora support a drone-first pipeline that produces lane-aware, trajectory-rich data to complement vehicle-mounted sets (e.g., nuScenes, Waymo).

2.2.5 Drone Trajectory Datasets Beyond *highD*

Two drone-trajectory databases, in addition to *highD*, are frequently used as references for scene coverage and annotation practice at intersections and highway merge areas. Using accurate vehicle, bicycle, and pedestrian trajectories recorded by flying UAVs and released with clear labeling and map context, the **inD** dataset records naturalistic road users at four intersections in Germany (approximately 10 h total) [7]. Drones were used to reduce blockages, and the **exiD** dataset, which targets highway entries and exits in Germany, provides highly annotated trajectories at merged and split locations that are typically difficult to observe from the ground. It also includes metadata suitable for safety and interaction studies [43].

The new principles for dataset structure and analysis protocols that direct this replication effort are further discussed by other large-scale datasets such as Waymo Open [51,58], Argoverse 2 [2], KITTI [31,32], and INTERACTION [28].

2.3 Data Collection Strategies

After the development of benchmark standards, it is important to take into account the data collection techniques that make such datasets possible. The location coverage, sensor types, and ability for motion collection of different methods are different.

2.3.1 Side-Road Cameras

Long-term continuous monitoring is provided by fixed roadside cameras in well-known locations. The establishment of standard 2D/3D tracking parameters and protocols in ground-based perspectives was made easier by the KITTI tracking benchmark, which consists of 21 training sequences and 29 test sequences [31,32]. The detection and tracking of multiple vehicles at junctions and highways under weather, light and occlusion variability was further highlighted by the UA-DETRAC benchmark [59]. Although perspective limitations and blockages are more noticeable than in normal top-down views of the UAV, these datasets are still useful for CCTV-style applications.

2.3.2 Probe Vehicles and Multimodal Fleets

Synchronized multi-sensor devices improve coverage with large fleet datasets. For lidar, radar, and cameras, the nuScenes dataset (Boston/Singapore) offers well-synchronized keyframes at 2 Hz with annotations for over 23 object classes and corresponding maps [11, 44]. High-resolution multi-camera and lidar streams with 2D/3D labels and consistent track IDs across frames are available in the Waymo Open Dataset, facilitating cross-geography generalization analyzes and robust detection and tracking studies [51,58]. With HD maps and complex scenarios created for trajectory and planning research, Argoverse 2 provides perception and motion forecasting data [2]. INTERACTION provides precise

Table 2.1: Comparative Summary of Benchmark Datasets in Traffic Analysis

Dataset Name	Data Source/Collection Method	Key Features	Primary Use Case
NGSIM	Synchronized rooftop video cameras on highways [24, 54]	High-frequency(10 Hz) vehicle trajectories at the lane level [24, 53]	Trajectory prediction, car-following, and lane-changing models [19]
highD	Drones(UAVs) over German highways [33, 34]	Bird's-eye view with reduced occlusions; covers over 110,000 vehicles [33, 34]	Naturalistic driving models and safety analysis [34]
nuScenes	Multi-sensor fleets (6 cameras, lidar, radar, GPS/IMU) in Boston & Singapore [11]	360° sensor coverage with 23 object classes and map data [11, 44]	Integrated autonomous driving research (detection, tracking, planning) [11]
inD	UAV (RGB) over German urban intersections [7]	Top-down (hovering); ∼10 h multi-scene episodes; trajectories for vehicles, cyclists, pedestrians with map context	Interaction modeling, trajectory forecasting, and safety analysis
exiD	UAV (RGB) over motorway merge/diverge ramps in Germany [42, 43]	Top-down views of entries/exits; scenario episodes (merge/diverge); vehicle trajectories with map/context metadata	Merge behavior analysis, safety surrogates, and planning/evaluation
KITTI	Vehicle-mounted multi-sensor rigs [22]	Ground-based perspective with data for 2D/3D tracking benchmarks [31, 32]	Standardized 2D/3D tracking and detection for ground-level views [22, 32]
UA-DETRAC	Fixed roadside CCTV cameras [59]	Challenges of weather, lighting, and occlusion variability from a fixed viewpoint [59]	Multi-vehicle detection and tracking at junctions from a CCTV perspective [59]
VisDrone	Drones (UAVs) in various cities and locations [74]	Benchmarks specifically for aerial detection and tracking tasks [74]	Evaluating aerial vision pipelines, especially for small targets [74]
UAVDT	Drones (UAVs) with metadata (weather, altitude, camera view) [15]	Approximately 80,000 frames for multi-object tracking (MOT) from aerial views [15]	Benchmarking detection and MOT algorithms under various aerial conditions [15]

trajectories and map context to study exchange behaviors, with a particular focus on high-interaction driving (unsignalized intersections, roundabouts, merging) [28].

2.3.3 Drones

With adjustable altitude and angle, Unmanned Aerial Vehicles (UAVs, commonly referred to as drones) offer wide-area coverage depending on the requirement, making it possible to observe complex junctions and multi-lane hallways. UAV-specific challenges are defined by two popular benchmarks, VisDrone and Unmanned Aerial Vehicle Detection and Tracking (UAVDT). For detection and tracking, VisDrone provides detailed image and video benchmarks gathered from various cities and locations [74]. For detection and Multi-Object Tracking (MOT) tasks, UAVDT provides approximately 80,000 frames from around 10 hours of video with metadata (weather, altitude, and camera view) [15]. The evaluation of aerial vision and motivated pipelines that combine lightweight real-time detectors with MOT appropriate for small targets and frequent obstructions are supported by these resources.

Operational aspects of capturing UAV traffic

Recent surveys summarize what UAV platforms can (and cannot) measure in traffic scenes and how flight parameters affect data quality [10, 26, 45]. A fundamental trade-off is driven by altitude and camera pitch: lower altitudes improve per-object resolution but increase perspective distortion and occlusions from large vehicles or street furniture; higher altitudes reduce occlusions and yield more stable homographies but shrink targets (smaller bounding boxes, lower signal-to-noise ratio). Viewpoint stability also matters: hovering over the region of interest with a gimbal-stabilized nadir/oblique view simplifies tracking and planar calibration, whereas aggressive yaw or lateral motion increases inter-frame parallax and complicates speed estimation unless tightly compensated.

Practical and legal limitations

Mission planning and airspace rules shape UAV operations: sequence length is limited by battery endurance and payload; site coverage is constrained by VLOS/BVLOS permissions, geofencing, and local altitude ceilings; and privacy policies restrict both camera pointing and data retention. To improve downstream detection, tracking, and calibration, the surveys recommend pre-flight checklists (weather, wind, GNSS conditions), frequent compass/IMU calibration, and conservative speed/altitude profiles to reduce platform shake and blur [10, 26, 45]. Consistent with these guidelines, we used hovering flights at discrete altitudes (30-80 m) with limited pitch variation to balance occlusion, resolution, and calibration stability.

2.4 Object Detection

Object detection is a computer vision technique that uses deep learning to identify and locate specific objects, such as people or cars, within images or videos. Through the Object Detection algorithm, it is possible to obtain, starting from video recorded by Cameras, Drones, or other sensors, which agents are present and their locations.

2.4.1 YOLO in Practice

YOLO (You Only Look Once) is a family of real-time object detection algorithms that formulate detection as a single regression problem: given an input image, it directly predicts bounding boxes and class probabilities in one neural network pass, unlike previous methods that performed region estimation and classification in multiple stages [46]. This design makes YOLO both accurate and extremely fast, enabling real-time inference on GPUs. In later versions, the YOLO family introduced better backbone architectures, anchor-free designs, decoupled detection heads, advanced label assignment strategies such as SimOTA (Simple Online and Offline Target Assignment) and improved training methods, leading to modern implementations (YOLOv1-v8) [21, 56].

With its characteristics, YOLOX showed state-of-the-art results among real-time detectors on COCO while maintaining efficiency on conventional GPUs [20, 21].Without the need for external pre-training, YOLOv7 further advanced the real-time frontier by achieving state-of-the-art average precision at real-time FPS(frames per second) across model scales, from edge devices to datacenter environments [55, 56, 63]. Combined, these models provide advantages for UAV video processing pipelines because they maintain real-time throughput at levels suitable for aerial footage, and their open-source implementations are stable and widely supported by active communities [20, 63].

The latest research on UAVs and small objects shows that YOLOv8 functions as an effective baseline model, which researchers use to improve its performance. The authors of Modular YOLOv8 Optimization for Real-Time UAV Maritime Rescue Object Detection [73] introduced modular plug-in enhancements which they added to YOLOv8 to achieve better accuracy on the SeaDronesSee UAV dataset while maintaining fast processing speeds. Recent surveys on small and oriented objects in aerial images [60] and the DOTA benchmark for aerial detection [64] highlight the need for multi-scale features and oriented representations in UAV scenes. Authors of SOD-YOLOv8: Enhancing YOLOv8 for Small Object Detection [30] developed it to improve small object detection capabilities for dense UAV operational environments. Concurrently, lightweight YOLOv8 adaptations have been validated on UAV tasks-e.g., modular plug-ins for maritime rescue [73] and SOD-YOLOv8 for small objects in dense scenes [30].

2.4.2 Aerial (UAV) Viewpoints

UAV imaging systems face challenges when detecting small targets because these systems operate at different altitudes and use angled viewing perspectives. The VisDrone and UAVDT datasets show reduced average precision for small vehicles and vulnerable road users compared to the ground view data sets, which indicates the need for advanced multiscale features and improved data augmentation techniques [15, 74]. The research field of UAV detection focuses on combining different scales of information with anchor-free detection methods, separate classification and regression processes, and improved label assignment techniques to improve the detection performance of small objects [20, 21, 55, 56, 64].

Remote-sensing detectors for small, oriented targets

The remote-sensing community offers platforms and techniques that directly target small objects in top-down, dense scenes with random orientations, enhancing VisDrone/UAVDT. The DOTA benchmark is still the standard aerial detection platform to evaluate oriented heads and rotation handling, and it established oriented (quadrilateral) annotations at very large image scales and high instance density [14, 65]. The key to recovering small targets in overhead views is multi-scale feature structure, rotation/orientation modeling, and attention mechanisms, according to technique overviews from recent surveys [27, 60]. Rotation-aware detectors like ReDet, which specifically encodes rotation equivariance for aerial imagery, are based on the same design principles [25]. Community challenges such as DeepGlobe and the SpaceNet series advanced road-network extraction from overhead imagery and standardized evaluation for mapping tasks for complementary context layers (lanes/roads) that support speed and safety analysis [13, 18].

2.4.3 Roadside CCTV Detectors

Ground view corpora including KITTI and UA-DETRAC enabled researchers to develop multi-object detection systems for traffic videos at fixed camera positions, which later influenced both dataset construction and image processing methods that became applicable to UAV data analysis [31, 59]. Ground-view benchmarks tend to show higher APs because their objects appear larger, but the strategies from these datasets including temporal smoothing and ROI masking and curriculum schedules prove useful for UAV applications.

2.4.4 Lane/Marking Extraction (Context for Speed and Safety)

Recent UAV/airborne studies help to combine HD lane maps with trajectories for per-lane movement and safety analysis. This uses encoder-decoder segmentation with attention and robust post-processing to extract road markings and lane boundaries even under shadows and occlusions for lane-aware analysis [23, 38]. The semantic layer is an essential tool that makes it possible to extract distance and lane-consistent speeds from drone tracking data.

From segmentation to vector maps (roads/HD maps)

Classic road/HD-map extraction has also advanced toward *vectorized* outputs, going beyond pixelwise lane/marking segmentation [23, 38]. RoadTracer avoids the brittle post-processing of pure segmentation and produces cleaner topologies for downstream analysis by formulating road mapping as an iterative graph-tracing process guided by a learned policy [4]. HDMapNet demonstrates that lightweight, local lane context can be built incrementally without full offline HD-map production in the vehicle-mounted setting by learning online, local high-definition semantic maps directly from onboard camera/LiDAR [37]. If the application is speed/safety analytics instead of long-horizon planning, recent HD-map surveys further support the use of practical, task-driven map layers (lanes, boundaries, crossings) over bulky, monolithic maps [3]. This drives our decision

to forego full HD-map building in favor of maintaining a lightweight lane context (for lane-consistent speeds and safety surrogates).

2.5 Tracking

In order to create consistent identity-based trajectories that result in velocity and speed profiles, as well as advance measurements, the system must follow detected objects within every frame. Two main practical approaches are developed for real-time pipeline operations.

In order to improve ID switch reduction in crowded areas and occlusion situations while preserving fast processing speeds, DeepSORT applies deep appearance embedding to the SORT system through its motion model, which uses Kalman and Hungarian association [6, 61, 62].

Before applying track similarity and motion-cue filters, the ByteTrack system links all detection boxes, including those with low confidence scores. The main idea is to use partially hidden targets to improve IDF1 and HOTA performance while providing quick, real-time results that are better than the MOT17 and MOT20 benchmarks and continued to perform well across the board [69–71].

Multi-object tracking for aerial footage

In addition to DeepSORT and ByteTrack, a number of MOT techniques are frequently used to address non-linear dynamics, camera motion, and occlusions in videos. **S**tronger appearance signals, camera-motion adjustment and improved Kalman state are some of the ways that BoT-SORT enhances the SORT/DeepSORT family; reports the highest accuracy on MOT benchmarks while still being useful [1]. **I**n scenes where motion is non-linear, OC-SORT greatly increases robustness by revisiting the SORT pipeline and suggesting an observation-centric update that decreases buildup of errors during occlusions [12]. **StrongSORT** provides a stronger plug-and-play baseline with enhanced identity preservation by revisiting DeepSORT end-to-end (detector, embedding, association) [16]. In order to comprehend precision-recall trade offs for ID consistency, these trackers are frequently evaluated together with DeepSORT/ByteTrack for UAV videos with small objects and frequent periodic occlusions.

UAV Alternative MOT baselines

In aerial settings, two drop-in baselines are frequently used in addition to DeepSORT and ByteTrack. In order to preserve identity consistency while remaining real-time, **FairMOT** carries out joint detection and re-identification in a single network [72]. **OC-SORT** returns to geometry-centric association with improved observation handling and motion modeling, yielding robust results under non-linear motion and occlusions while keeping the pipeline simple and fast [12]. Historically, tracking was summarized with CLEAR MOT metrics (MOTA/MOTP) [5]; in this thesis we emphasize identity/association-aware scores such as IDF1 [48] and, where applicable, HOTA [39].

2.5.1 Evaluation Metrics

Identity stability functions as a critical element that impacts all traffic analytics systems. The tracking evaluation system now includes the Higher Order Tracking Accuracy (HOTA) metric that measures the detection and association and localization quality together with the Identity F1 score (IDF1) and the Multiple Object Tracking Accuracy (MOTA) [39, 40]. The HOTA/IDF1 evaluation method becomes essential for UAV operations because it better tracks objects during occlusions and small target detection and changing viewpoints than MOTA does.

Practical differences in traffic scenes

The ByteTrack system produces superior IDF1/HOTA results in dense aerial traffic because it operates without any additional re-ID network, which simplifies deployment and reduces computational resources [70]. The DeepSORT tracking system achieves success through its ability to track objects by appearance when images have high resolution at close range and maintains performance during long occlusion periods [62]. The choice of engineering stacks depends on the particular features of the site, which include its altitude level, pitch angle, and material density. In our aerial setting (moderate resolution with altitude-driven scale changes), DeepSORT's re-ID cues helped maintain identity continuity at 30–60 m, whereas ByteTrack remained competitive at higher altitudes (70–80 m) where small boxes and appearance degradation reduce re-ID reliability.

2.6 Calibration and Speed Estimation

Monocular Calibration and Speed Estimation

In traffic videos, the estimation of monocular speed usually depends on planar-scene assumptions in addition to camera calibration or homography recovery. Automatic calibration of CCTV footage is the subject of a long line of work. A traditional method allows speed from a single fixed camera by estimating the vanishing points and the scale of the scene from multiple minutes of video without user input [17]. Following that, the BrnoCompSpeed work provided a thorough benchmark and protocol for visual speed measurement along with meticulously examined ground truth for assessment [49]. These tools are widely used to measure the propagation of calibration error to speed error in monocular pipelines and to validate calibration processes (vanishing points, focal length/scale recovery). Before converting tracked pixel displacements to metric distances, our drone-based pipeline applies the same fundamental idea, ground-plane mapping by homography, to a UAV configuration with ground control points and homography estimation.

Fixed or Hovering Cameras

The traditional method relies on a flat surface and fixed camera position to calculate a homography that transforms pixels of ground-plane images into metric distance measurements. The BrnoCompSpeed dataset with later research created a strict calibration and validation method that used measured distances to show that speed accuracy depends

on vanishing points, focal length, and scene scale estimation [17, 49, 50]. For standard traffic scenes, the most recent research enables automatic calibration, reducing manual work and also keeping minimum errors for enforcement and safety applications [47].

Moving Cameras and UAV Flights

Variations in altitude or tilt during flight can be corrected using orthographic mapping with geographic basemaps or pose-based calibration techniques such as Perspective-n-Poin (PnP) with collected control points to guarantee a stable viewpoint before speed estimation [29,47]. Pipelines should include consistency checks and periodic recalibration when drone conditions change, as these methods show that calibration quality can be the main cause of error in speed estimation even more than detector and tracker noise [47,50].

Pose via PnP (EPnP) vs. Planar Homography

The full camera extrinsics are recovered from 3D-2D correspondences using Perspective-n-Point (PnP); EPnP solves PnP in $\mathcal{O}(n)$ time and is accurate for both planar and non-planar configurations [36]. Pose estimation and ground-plane projection can be obtained when trustworthy 3D ground control points (such as surveyed landmarks or LiDAR features) are available. Since there are no surveyed 3D points and the road surface inside the ROI is roughly flat in our setup, mapping pixel trajectories to metric ground coordinates is made easier and more reliable with a single 2D-2D homography. This decision keeps the pipeline lightweight while preserving the accuracy required for lane-consistent speeds.

In conclusion, the relevant literature shows how traffic monitoring has changed over time, moving from fixed point detectors to rich, AI-driven datasets gathered by UAV platforms and ground-based fleets. Although current perception algorithms such as YOLO and tracking frameworks such as DeepSORT and ByteTrack offer the computational resources required to convert unstructured video into trajectory-based data, benchmark datasets such as NGSIM, highD, and nuScenes have influenced expectations for sampling accuracy, annotation richness, and scene consistency.

The complete design of the new pipeline system is shown in Chapter 3 (System Model and Contributions) that follows. To establish the experimental framework and methodological results, the thesis describes how unprocessed drone footage is converted into structured datasets using a number of modules, including data acquisition, perception, calibration, kinematic estimation, and visualization.

Chapter 3

System Model and Contributions

In this chapter, the complete pipeline is presented. The system converts unprocessed drone footage into a structured traffic dataset with accurate vehicle trajectories, speed profiles, and visual characteristics.

The complete pipeline is composed of:

1. Data collection,
2. Detection and tracking,
3. Geometric calibration module,
4. Speed estimation,
5. Output and visualization.

The stages are sequential, each stage feeds the next, and each can be tested or replaced independently. Inspired by the limitations of fixed-point sensors, which are rarely updated and can offer only limited data, the suggested method uses aerial imagery to record traffic from different perspectives and altitudes, providing more varied and richer observations. A key contribution is a reproducible end-to-end methodology that includes robust calibration to map image-plane motion to real-world units, along with a high-quality, general-purpose dataset for AI research. In order for researchers and engineers to expand the system, the rest of this chapter describes the entire workflow.

3.1 Data Collection

High-definition RGB cameras mounted on unmanned aerial vehicles (UAVs) are used in the pipeline's first stage to capture aerial footage. Due to their mobility, broad operational boundaries, and capacity to operate at various altitudes and viewing angles, UAVs were selected. Drones can be repositioned to increase coverage and reduce blind spots, capturing the same scene from complementary perspectives, in contrast to fixed roadside cameras. The final videos serve as the system's input for the detection phase.

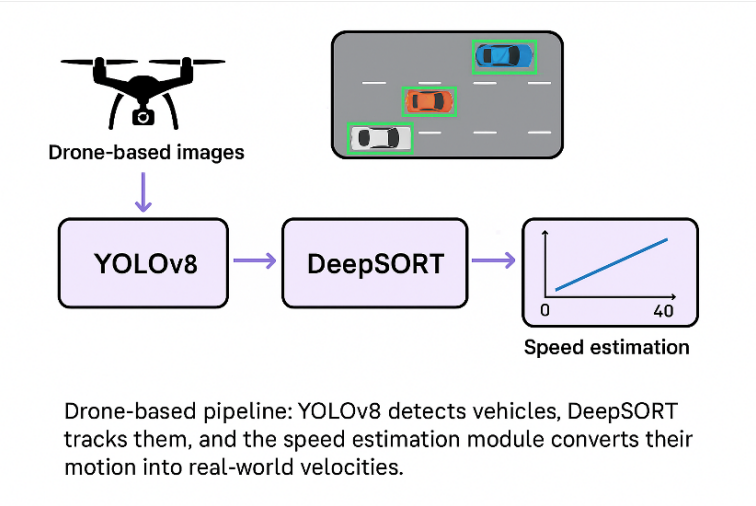


Figure 3.1: High-Level View of the System

3.1.1 Drone and Camera Specifications

The primary data collection tool for this project was the DJI Mini 2 drone. This model was selected because of its practical advantages, which included its small size and fast deployment and its adherence to regulations since it weighs less than 250 grams, which excludes it from multiple registration requirements. The system showed perfect suitability because of these features which enabled simple data collection from multiple locations without requiring complicated setup processes.

Figure 3.2: DJI Mini 2

The DJI Mini 2 camera system helped system obtain professional-grade footage that computers could analyze through vision processing. Its key specifications are as follows:

- **Camera Sensor**: The drone operates with a 1/2.3" Complementary Metal-Oxide-Semiconductor (**CMOS**) sensor. The sensor operates at a small size yet produces

detailed images when it receives enough light, which occurs during standard traffic monitoring operations. A larger sensor might offer better low-light performance, but for our daytime data collection, this was a suitable trade-off for the drone's portability.

- **Lens**: The camera has a fixed lens with an aperture of f/2.8 and a field of view of 83°. The wide aperture allows enough light to reach the sensor, ensuring clear and sharp video frames.
- **Resolution and Frame Rate**: All videos were recorded at 4K resolution (3840x2160) and at 30 frames per second (FPS).
- **Autonomy**: The expected flight time is around 31 minutes, which is relatively short and represents one of the main limitations of UAVs. However, this is acceptable for this first implementation.

High resolution is crucial for our task. The system allows YOLOv8 to detect and classify vehicles that appear at great distances because it provides enough pixel resolution for accurate detection. The system would lose track of small objects if it operated at lower resolutions because these objects would become invisible, leading to total system failure.

The tracking algorithm needs to produce smooth vehicle movements, so it uses a 30 FPS frame rate which generates continuous motion paths. The system maintains an equilibrium between accurate motion tracking and reasonable data storage needs. The system operated with a frame rate that matched the application needs because higher frame rates would have affected processing performance.

The hardware selection, together with the recording configurations, created optimal raw video data quality, which made it possible to perform object detection, tracking, and speed estimation.

Environmental Conditions

All video recordings were collected under the same environmental conditions consistently, maintained to obtain reliable datasets.

- **Lighting Conditions**: The research team collected data only when sunlight was present and visibility was at its best. We avoided periods of harsh, direct sunlight that could cause excessive glare or deep shadows, as these conditions can negatively impact the performance of object detectors. The YOLOv8 model can achieve higher accuracy levels because the camera sensor uses consistent indirect light to capture high-quality image data.
- **Weather Conditions**: All flights were recorded in clear and calm weather. The team stopped recording whenever rain, fog, or strong winds occurred. The combination of rain and fog creates visibility problems for vehicles and reduces image quality, but high winds create drone instability, resulting in unstable video recording. The system would become unstable, creating difficulties for the geometric calibration process and generating noise in the motion data, thus making speed estimation more challenging.

Standardizing these environmental variables helped us reduce the chances of bias appearing in our dataset. The standardization process enables the perception module to demonstrate its actual performance using altitude and traffic density as evaluation criteria, instead of challenging environmental conditions that affect the results.

Various heights of 30 m, 40 m, 50 m, 60 m, 70 m, and 80 m above the road surface were used to investigate the effects of object scale, image resolution, and perspective on subsequent performance. Larger and more detailed vehicle representations are produced at lower altitudes; at higher altitudes, the field of view is expanded, but the targets are smaller, and the complexity is increased.

At least two complementary perspectives were recorded for each altitude:

1. **Angled view**: cars move across the frame, showing their sides;
2. **Top-down view**: a bird's-eye view aerial perspective where lane markings and roofs predominate.

From plain, low-density scenes to complex, crowded situations where tracking and detection are most difficult, this variation facilitates evaluation across a range of scenarios.

3.1.2 Data Annotation and Preprocessing

An essential component is data annotation and preprocessing, which provide the ground-truth data to evaluate the performance of all components. It would be impossible to objectively evaluate the detection and tracking capabilities of the system without precise and consistent labels.

Why Ground Truth Annotations Are Needed

Ground-truth annotations, which provide the precise location (bounding-box coordinates) and category (class label) of each object of interest in a video frame, act as the object's target reference. Labeled data are needed because they provide the accurate responses required to optimize the YOLOv8 object detector and they serve as the standard by which the predictions of the tracker and the detector are evaluated, i.e., mean Average Precision (mAP) and IDF1 score.

Preparing the Raw Data

Long video recordings, usually lasting several minutes, were made up of the raw drone input. We first divided each flight into scene-based clips, ensuring that each sequence had a fixed viewpoint and altitude to prepare these recordings for annotation and subsequent machine learning tasks. The details of each recording are summarized in Table 3.1.

Annotation Tool: CVAT

We used the open-source browser-based Computer Vision Annotation Tool (**CVAT**), created by Intel, for labeling. For training YOLOv8, CVAT was selected due to its ability to export directly to YOLO format, support frame-by-frame video-based labeling, have

ID	Perspective	Altitude (m)	Duration (mm:ss)
A_30m	Angled	30	01:48
A_40m	Angled	40	02:07
A_50m	Angled	50	02:03
A_60m	Angled	60	02:02
A_70m	Angled	70	02:07
A_80m	Angled	80	02:02
B_30m	Angled	30	02:00
B_40m	Angled	40	02:00
B_50m	Angled	50	02:03
B_60m	Angled	60	01:37
B_70m	Angled	70	01:55
B_80m	Angled	80	0:38
C_135m	Top-down	30	01:57
C_145m	Top-down	40	02:02
C_170m	Top-down	50	01:58
C_190m	Top-down	60	01:56
C_220m	Top-down	70	01:58
C_260m	Top-down	80	01:59

Table 3.1: Drone recordings used to build the dataset.

a clean interface for managing large object class sets, and have built-in visual checks for bounding-box quality. Additionally, the amount of manual work needed was significantly decreased by CVAT's interpolation-based bounding-box tracking across frames, which allowed annotators to modify boxes according to changes in object position or scale.

Annotation Process

The annotation protocol was designed to produce high-quality and consistent labels. We annotated the first 200 frames of each video after tests showed that this amount was sufficient to record multiple instances of each target class across different spatial positions without overloading the annotators. To improve model performance, we started with the VisDrone2019 classification and rebalanced and simplified the set of labels: tricycle and trailer were eliminated, closely related categories (pedestrian and people $\rightarrow$ person) were combined, and the seven core classes (person, bicycle, motorcycle, car, van, truck, and bus) were retained.

In order to reduce background and preserve uniform placement throughout frames, bounding boxes were carefully drawn around the visible extent of every object, including areas that were partially covered up. Boxes were transmitted through time using CVAT's interpolation when appropriate, with manual corrections made whenever the scale or position of the objects changed. Finally, each bounding box was assigned one of the seven predefined labels to ensure a consistent mapping between the annotation files and the YOLOv8 class indexes. High-quality ground truth is essential because noisy or inconsistent labels can lead to models learning incorrect correlations. In order to find missing

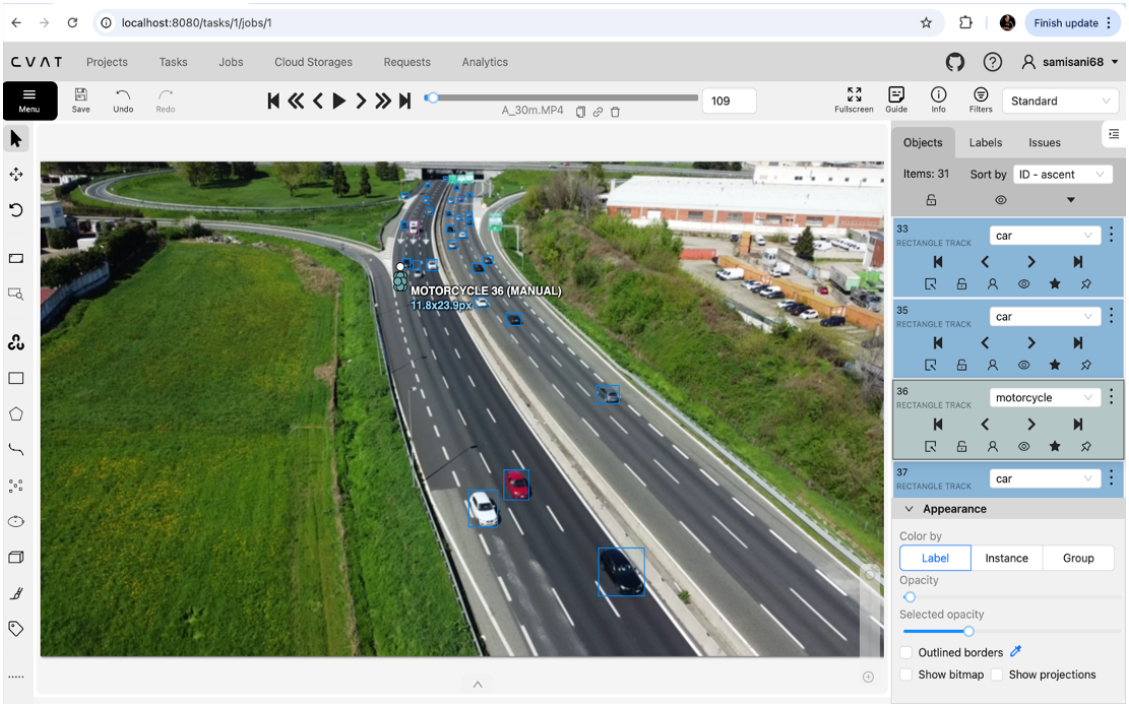


Figure 3.3: CVAT labeling

instances, incorrect boxes, and box drift, we visually reviewed each annotated sequence frame by frame. Then, we loaded all YOLO-format label files into a verification script to perform format validation. This allowed us to quickly identify coordinate errors, off-image boxes, and class-index inconsistencies by placing the boxes on the corresponding frames.

Conversion to YOLOv8 Format

A particular annotation format is required for YOLOv8:

Each image has a single text file with the following lines: class_id, x_center, y_center, width height. Regarding image sizes, all coordinates are normalized to fall between 0 and 1.

This format included our raw annotations that were exported straight from CVAT, saving us additional conversion steps and preparing the dataset for training without the need for additional preprocessing.

We split the data set into three:

- Training set (70%), trained YOLOv8.
- Validation set (15%), Used in training for hyperparameter tuning and overfit detection.
- Test set (15%), reserved exclusively for final testing.

Frames of the same video were not used in more than one split to prevent data leaks. This was done to test the model in scenes that had never been seen before, making the test more realistic.

After preprocessing and annotation, we had a clean and consistent dataset ready for YOLOv8. Every instance of an object was marked with a class label and a precise bounding box. The dataset was carefully chosen to maintain the underlying class distribution seen in actual traffic scenes while being sufficiently balanced for detector training. The reference standard for evaluating multi-object tracking performance and detection accuracy is this ground truth.

3.2 Perception Module

It starts with processing raw video frames to detect and track all road users. The system consists of two main components, which are object detection and multi-object tracking.

3.2.1 Object Detection with YOLOv8

We used YOLOv8, one of the models in Ultralytics' "You Only Look Once" family. YOLO offers a good trade-off between speed and accuracy by framing detection as a single-pass prediction problem that yields bounding boxes, class probabilities, and confidence scores for every frame. The architectural changes made to YOLOv8 (modern backbone, improved neck, decoupled heads) increase sensitivity to small objects, which is appropriate for our use case in aerial views, where vehicles can only take up a few dozen pixels.

YOLOv8 was chosen for some practical reasons. provides high frame rates with acceptable accuracy, which is necessary for long drone recordings. The computational overhead of two-stage detectors, like Faster R-CNN, is reduced by its single-pass design. It shows stronger performance on small objects, crucial for overhead views with distant vehicles. Finally, the Ultralytics implementation provides clean Python APIs and training scripts, simplifying integration and fine-tuning.

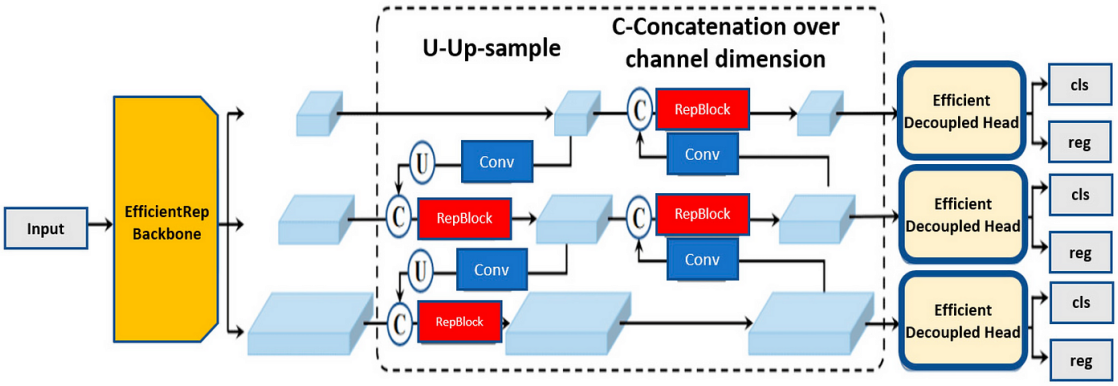


Figure 3.4: YOLOv8 architecture (reproduced from [52]).

Table 3.2: Training configuration for YOLOv8 variants on the UAV traffic dataset.

Model	**Init. weights**	**Train. schedule (epochs@imgsz)**	**Batch**	**Optimizer**	**LR schedule**
YOLOv8n	COCO pre-trained	50@640	16	SGD + momentum	Cosine decay
YOLOv8l	COCO pre-trained	100@1024	16	SGD + momentum	Cosine decay
Common inference settings: NMS IoU = 0.60; conf. threshold = Ultralytics default unless stated.					

Fine-Tuning with VisDrone

The default YOLOv8 models are pre-trained on COCO, which prioritizes larger object scales and ground-level views. Targets are smaller and backgrounds (roads, buildings, lane markings) are more intricate in aerial footage. We fine-tuned using VisDrone, a drone-based dataset of urban and suburban scenes whose object sizes, viewpoints, and road geometries correspond more closely to our target domain, in order to address this domain gap. This alignment is especially useful for highway or intersection layouts, mixed traffic with pedestrians and bicycles, and small vehicles in aerial images.

We limited the classification to seven pertinent classes (person, bicycle, motorcycle, car, van, truck, and bus) after converting VisDrone annotations to YOLO text files with class indices and normalized box coordinates prior to training. To prevent imbalance, rare or out-of-scope categories (tricycle, trailer) were eliminated, and closely related categories (pedestrian / people into person) were combined. In order to facilitate consistent model selection and evaluation, we lastly created training, validation, and test splits.

We evaluated two capacities: YOLOv8l (large) for greater accuracy at higher computation cost and YOLOv8n (nano) for real-time, resource-constrained scenarios. The models were optimized using random gradient descent with velocity and a cosine learning rate schedule after initialization from COCO pre-trained weights. We performed two runs to highlight small object recall: 50 epochs at 640×640 resolution and 100 epochs at 1024×1024. In order to stabilize gradient updates and fit the available GPU memory, a batch size of 16 was chosen. In order to balance recall and precision, non-maximum suppression employed an IoU threshold of 0.6.

Consistent improvements in metrics and qualitative behavior were obtained through fine-tuning. In far roads, where vehicles frequently measured less than 30×30 pixels, small-object recall significantly improved; confusions between visually similar classes (such as cars and vans) decreased; and both mAP_{50} and $\text{mAP}_{50:95}$ increased compared to baselines that only included COCO. Furthermore, predictions were more temporally stable, which improved downstream association and decreased frame-to-frame flicker. For example, the optimized YOLOv8l detected significantly more vehicles than the COCO-only model, particularly at long range, on a test sequence at an altitude of 70 m. Table 3.2 summarizes the training configuration for YOLOv8n/l on our UAV dataset.

Stronger detection improves tracking and speed estimation in three ways: more consistent boxes reduce false identification, stable class labels limit identity switches, and

higher recall results in longer and more complete trajectories. In fact, training YOLOv8 in VisDrone makes a general-purpose detector perfect for aerial traffic on roads and intersections.

A More Comprehensive Look at YOLOv8 Fine-Tuning

A careful selection of training hyper parameters was necessary to optimize the YOLOv8 model for our particular task of aerial vehicle detection. These settings have direct control over the learning process, they were selected to avoid problems like overfitting by keeping a balance between stable convergence and model accuracy.

Learning Rate and Timetable: We started training at a learning rate of 0.01(lr0=0.01). For similar object detection tasks, this value is a typical and useful starting point. In order to improve the model learning over time, we used a schedule for the cosine learning rate(cosine). This technique allows the model to make large initial improvements and then more accurately adjust its weights later by reducing the learning rate during training.

Batch Size: A batch size of 16 was chosen for training (batch=16). The main consideration in this decision was to find a balance between the stability of training and computational resources. A batch size of 16 offered a good option, fitting within our available memory, and the model learned from a sufficiently different set of examples in each iteration to ensure good generalization.

Data Augmentation: We used a number of data augmentation strategies to help the model better handle the variances found in actual aerial footage. By making customized copies of preexisting images, these methods artificially increase the size of the training dataset. Among the major improvements were:

By combining four distinct training images into one, a technique known as mosaic augmentation forces the model to learn how to recognize objects in a variety of settings and at varying scales.

Geometric Transformations: We applied the pictures with random translations, scaling, and rotations. As a result, the model can better adjust to the drone's altitude and perspective.

Color Space Adjustments: Small changes in brightness, contrast, and vibrancy were made to simulate various lighting scenarios and improve the detector's strength during the day. You usually don't need to add extra commands for the data augmentation techniques (color adjustments, mosaic, and geometric adjustments) unless you want to change their default values because they are enabled by default in the YOLOv8 training pipeline.

We guarantee that the fine-tuning procedure is not only efficient, but also systematic and repeatable by providing justification for these decisions. Achieving the high detection accuracy described in Chapter 4 required a combination of robust data augmentation, a suitable batch size, and a well-tuned learning rate.

Multi-Object Tracking

Following detection, the tracking module assigns persistent identities to objects across frames, turning per-frame bounding boxes into continuous trajectories. This feature is

crucial for traffic analysis because without accurate identity maintenance, vehicle tracks fail, and speed estimation loses its precision.

Challenges not found on the ground are introduced by aerial footage. Blockages occur when cars overlap or go under bridges and trees; perspective effects change over the frame; and vehicles take up fewer pixels, making them more difficult to follow over time. Despite these challenges, efficient tracking uses both time and geographic information to determine whether a current frame detection matches an object that has been seen before.

Our system uses a tracking-by-detection methodology. For every frame, YOLOv8 generates detections on its own. The tracker then uses visual similarity and predicted motion to associate these detections with tracks that already exist, creates new tracks for new objects, and ends old tracks when observations stop. We can independently upgrade the tracker or detector due to this modular design.

- ByteTrack:

  ByteTrack emphasized on strong data association, even in cases where detections are not perfect. It divides detections into high- and low-confidence sets and carries out association in two passes: first, it matches high-confidence detections to tracks that already exist using motion prediction and IoU-based Hungarian assignment; second, it reconciles remaining tracks with low-confidence detections to recover objects that were momentarily obscured or missed. After a long break, the tracks are retired and new tracks are created for unmatched detections. In practice, this approach reduces the production of false tracks and increases the robustness to temporary misses. One drawback of association is that it mainly uses motion and bounding-box geometry, lacking appearance features, which can lead to more identity switches in scenes with a large number of visually similar vehicles.

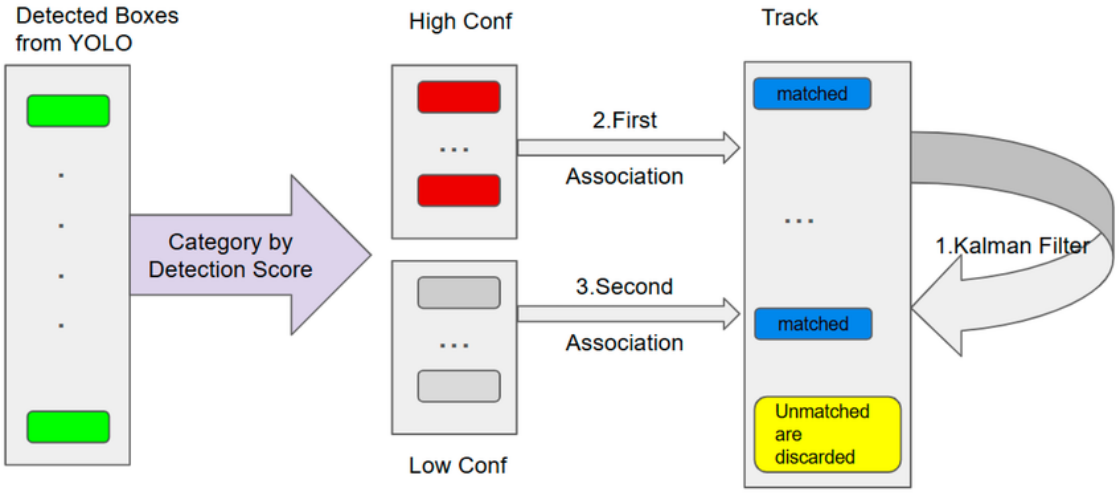


Figure 3.5: ByteTrack architecture (reproduced from [67]).

- DeepSORT:

DeepSORT adds appearance-based re-identification to the standard SORT (Simple Online and Real-Time Tracking) framework. A convolutional network extracts an appearance structure for every detection, a Kalman filter predicts the next position of each track, and association minimizes the total cost that takes into consideration both geometric proximity (e.g., IoU) and appearance similarity (cosine distance). The additional appearance model facilitates the maintenance of identity, especially on crowded roads, through close interactions and small impediments. The limitations are sensitivity to sudden changes in light between frames and additional computation for feature extraction.

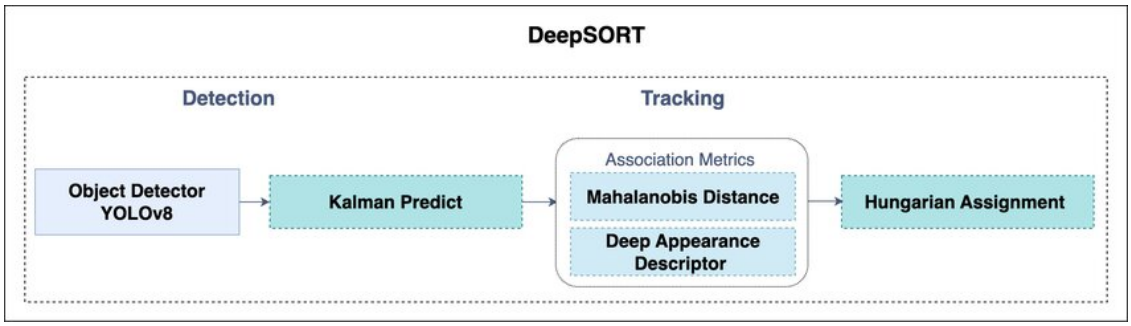


Figure 3.6: DeepSORT architecture (reproduced from [68]).

In fact, at lower flight altitudes (30–60 m) the appearance branch in DeepSORT improves ID consistency, while at higher altitudes ByteTrack stays competitive despite lacking re-ID, thanks to its robust two-pass association on sparse, small detections.

Comparative Evaluation and Selection

In order to compare a geometry-only association strategy (ByteTrack) against a geometry-plus-appearance strategy (DeepSORT), we deployed both trackers and tested them on identical YOLOv8 detections at various altitudes and traffic densities. To ensure that improvements in identity, stability and temporal consistency resulted in further utility, the evaluation took into account common MOT metrics, such as IDF1 for identity preservation, MOTA/MOTP for overall accuracy and localization precision, and average track length for continuity. Overall, DeepSORT was chosen as the pipeline's default tracker because it offered longer, more reliable identities in dense traffic. In situations with less traffic, ByteTrack was still a strong choice because of its accuracy and efficiency, which resulted in reliable trajectories at a reduced computational cost.

Continuous, accurately associated trajectories are necessary for speed estimation. While DeepSORT's appearance indicators help maintain identity through collisions and lane changes, resulting in fewer breaks in trajectories, ByteTrack's precise association reduces the possibility that spurious detections could affect speed calculations. Accordingly, ByteTrack is preferred for speed and accuracy in light to moderate traffic, while DeepSORT is useful for scenes that are crowded, even though it has a higher computational overhead.

DeepSORT Implementation Details

We chose DeepSORT as our main algorithm for the tracking module because of its exceptional ability to preserve object identities in situations with high traffic. We set up its essential parts as follows to get this performance: Model of Appearance Descriptor: The deep appearance descriptor of DeepSORT, which learns to identify a car based on its visual characteristics, is its strongest point. A standard convolutional neural network (CNN) that had been pre-trained on a sizable person re-identification dataset was employed. The features that this network learns are general enough to successfully differentiate between various vehicles based on attributes such as color and shape, even though they were trained on humans. When a car has been momentarily obscured, or "occluded," by something like a bridge or a bigger truck, this visual information is essential for re-identifying it.

Association Gating with Mahalanobis Distance: DeepSORT uses a two-factor authentication procedure when linking new detections to tracks that already exist. It starts by predicting where a car should be in the following frame using the Kalman filter. The distance between a new detection and this predicted position is then calculated using the Mahalanobis distance. For this distance, we established an optimistic threshold; a detection that was considered an unlikely match and rejected if it was too far from its expected location. This process, called gating, reduces tracking errors and efficiently removes unreliable associations.

Configuration of the Kalman Filter: DeepSORT's motion prediction core is the Kalman filter. We set it up using a conventional linear motion model with constant velocity. For highway traffic, this model makes the accurate prediction that cars will continue to travel in a generally constant direction and at a relatively constant speed from frame to frame. Even if a detection is missed for a few frames, the Kalman filter can still track vehicle trajectories smoothly because it is constantly updating its prediction based on new detections.

Our DeepSORT implementation was able to generate the long, stable vehicle tracks required for precise speed estimation by fusing the strong re-identification capabilities of the deep appearance model with the predictive power of the Kalman filter.

3.2.2 Experimental Preprocessing with Semantic Segmentation

We conducted a study of exploration to see if pre-processing frames with semantic segmentation could enhance YOLOv8 along with the main detection-tracking pipeline. Isolating the road and blocking out unnecessary background (sky, trees, buildings) was supposed to concentrate the detector on the area of interest, decreasing false positives, and enhancing recall, especially for small or distant cars.

Each frame was semantically segmented using a standard DeepLabV3 network from PyTorch (implemented in `background-subtraction.py`), and a mask was created using CVAT to preserve pixels from target classes (e.g. car, bus, truck). A foreground-only video was created by blacking out non-target pixels. Although this approach was easy to implement, it was limited by the classification of the pre-trained model which only retains the objects it was trained to recognize and does not learn a generic road region.

We trained a domain-specific segmentation model to distinguish between two classes of foreground (road surface and on-road objects) and background directly in our drone footage to work within these limitations. To identify the road as a foreground region, polygonal masks were added to the frames created by `frames.py` (through `convert-json.py`). `split.py` was used to separate image mask pairs into training and validation sets. After that, `train.py` was used to train a DeepLabV3 model for 100 epochs, learning the geometries of highway scenes. The file `checkpoints_best_model.pth` contained the best checkpoint.

`predict.py` was used to apply the custom model to test videos. To visually verify that the road surface and on-road objects were appropriately isolated, the predicted foreground mask appeared semi-transparently (green) on top of each frame. Both the original videos and the masked videos created by each segmentation technique were subjected to YOLOv8 in order to measure impact. We evaluated whether the preprocessing step provided advantages by comparing detections from both pipelines against ground truth and reported standard metrics using the script `YOLO-Comparison.py`. Next chapter presents the results in detail.

3.3 Geometric Calibration Module

Tracks extracted in pixel coordinates must be geometrically calibrated to obtain real-world meaningful measurements. Due to perspective effects introduced by aerial video, the pixel distances do not precisely match the ground distances. To address this, we use a projective calibration based on Ground Control Points (GCPs).

Using Google Maps' real distance measurements, a human operator manually selects particular spots on the road whose exact geometry is known for each video. Assume that $\mathbf{P} = [X\ Y\ 1]^T$ stands for the corresponding ground-plane coordinates (meters) and $\mathbf{p} = [u\ v\ 1]^T$ for the homogeneous image (pixel) coordinates. A 3×3 homography $\mathbf{H}$ is used to map points between planes:

$$\mathbf{P} \sim \mathbf{H}\,\mathbf{p}.$$

It is estimated and stored for later use by a minimum of four different GCPs. Speed estimation can then be performed following by analyzing the calibrated trajectories in metric units.

GCP Selection Strategy

The quality and location of the Ground Control Points (GCPs) have a fundamental impact on the geometric calibration's accuracy. The four GCPs were chosen using a strategy intended to optimize the accuracy and stability of the homography matrix. To find a solution for the homography, at least four points must be created and where they are placed is important. Reasons for Selection: GCPs that were parallel (on the same flat plane as the road), static and easily observable were the ones we chose. Features such as the corners of the applied road markings or noticeable cracks in the road were considered

ideal points. Since their locations are not fixed, we avoided using temporary objects such as cars or shadows.

Placement is important. The four GCPs were spread out as widely as possible to cover the main traffic movement area in a large rectangle. This is the most crucial element in getting an accurate calibration. The entire area can be accurately mapped by the homography transformation when the points are widely separated. Applying the transformation to points far from the center can result in large, unpredictable errors if the points are grouped together in a small area. In particular for cars at the frame's edges, this deformation would result in wildly incorrect speed estimates. In conclusion, one of the main possible causes of pipeline error is the wrong choice of GCPs.

3.4 Speed Estimation

Vehicle speeds are reliably and consistently estimated using the calibrated trajectories. This step relies on precise per-frame detections, tracks that are temporally consistent, and accurate plane homography across the region of interest.

As is common for highways, we assume that the observed road area is nearly flat. 3D methods would be necessary for significant deviations from flatness, such as steep banking. To correct for perspective errors caused by the angle of the camera, the image points (u, v) are mapped to the ground plane points (X, Y) in meters using homography $\mathbf{H}$.

By clicking on image features and providing equivalent metric distances calculated using high-resolution basemaps (e.g., Google Maps "Measure distance"), GCPs are interactively collected (`click-gcp.py`). Pixel coordinates (`*-points.csv`) and metric requirements (`*-distances.csv`) are stored by the script. after that, `calculate-matrices.py` calculates $\mathbf{H}$ that most accurately maps the chosen points in the image to a locally determined ground coordinate frame.

(a) Google Maps measurement

(b) GCP collection

Figure 3.7: Ground Control Points (GCPs) and homography estimation.

To reduce the effect of height bias, a stable ground-contact reference point is selected for every detection on a tracked object; the center of the bottom border of the bounding box is used. To obtain metric trajectories, the script `calculate-speed-smooth.py`

projects back the sequence $\{(u_t, v_t)\}$ into the ground plane coordinates $\{(X_t, Y_t)\}$ using $\mathbf{H}$ (OpenCV `perspectiveTransform`).

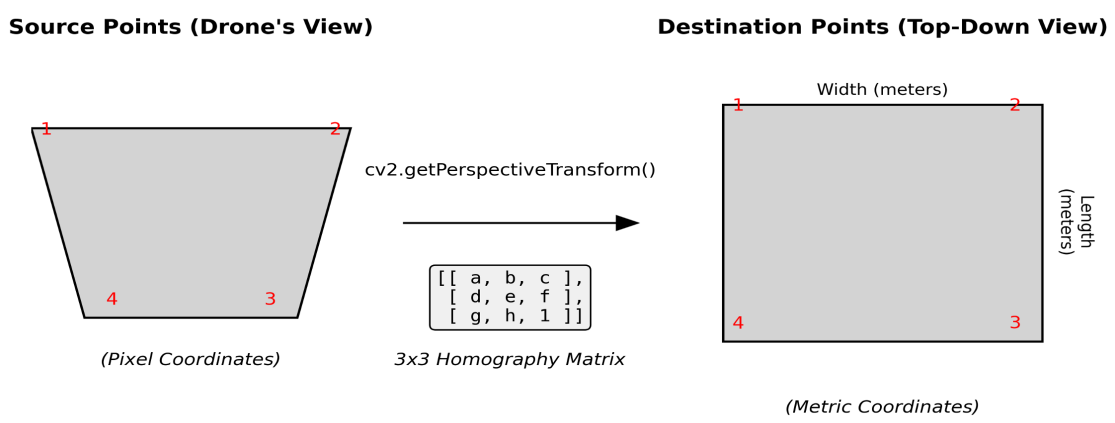


Figure 3.8: Projection of vehicle trajectories from image to ground plane.

The `perspectiveTransform` function

At the core of our speed–estimation module is OpenCV's `cv2.perspectiveTransform`. Given a 3×3 homography H, the function maps 2D points from the image plane to a rectified ground plane. In our setup this "top–down" plane is defined by Ground Control Points (GCPs) with coordinates specified in *metric units*; therefore, the outputs are in meters (up to the accuracy of the calibration).
Input:

- *Source points* (u, v)*:* the vehicle ground–contact points (bottom–edge centers of the boxes) in pixel coordinates, stacked as an array of shape $N \times 1 \times 2$ (float32/float64) for OpenCV.

- *Homography* H*:* a 3×3 matrix estimated from image $\rightarrow$ ground GCP correspondences. We compute H with `cv2.getPerspectiveTransform` when using exactly four non–collinear points, or `cv2.findHomography` (optionally with RANSAC) when using more than four for robustness.

Planar homography and interpretation

Let

$$H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \qquad \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a\,u + b\,v + c \\ d\,u + e\,v + f \\ g\,u + h\,v + i \end{bmatrix}, \qquad (x, y) = \left(\tfrac{x'}{w}, \tfrac{y'}{w}\right).$$

This mapping is *projective*: after calculating (x', y', w), we divide by the homogeneous coordinate $w = g\,u + h\,v + i$ (not by g or h separately) to return to Cartesian ground-plane coordinates. In the *affine* special case ($g = h = 0$), the linear portion that reduces to rotation/scale/shear is the 2×2 block $\left[\begin{smallmatrix} a & b \\ d & e \end{smallmatrix}\right]$. In the affine case, the terms c, f are pure translations; otherwise, they function as offsets in the numerators. Projective effects, such as parallel convergence, are introduced by the coefficients g, h, which also establish the normalization w. A normalization of H to $i = 1$ is common; a homography is defined up to a nonzero scale. Road plane perspective is compensated by a planar homography, but 3D depth is not recovered.

Output: An array with the same shape as the input points is returned by `cv2.perspectiveTransform`; each (u, v) is mapped to (X, Y) on the ground plane. The resulting trajectories are in metric units and appropriate for speed computation when GCPs are specified in meters (and lens distortion is minimal or pre-corrected). In practice, when the camera is oblique, we apply this to the bottom-edge center of each detection to reduce bias from vehicle height.

The physical motion that occurs between each frame is the following:

$$\Delta d_t = \sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2},$$

$\Delta t = 1/\text{FPS}$

The speed per frame is $v_t = \Delta d_t / \Delta t$ (m/s), which is multiplied by 3.6 to get km/h. We apply a moving average filter (window = 5 frames) to $\{v_t\}$ to reduce high-frequency noise from detection jitter, small association errors, and parked cars in some scenes. This reduces false peaks while maintaining the underlying trend. The mean of the smoothed values $\{v_t\}$ is then used to determine a single representative speed for each track.

Output and Visualization Module

The pipeline produces two parts for quality control and analysis. CSV logs initially provide vehicle ID, class, frame index, image coordinates, calibrated ground coordinates, and speed estimates to help with traffic analytics. Second, operators can visually verify motion consistency, tracking, and detection by adding bounding boxes and current speeds to annotated videos. Upgrading or replacing components, like other calibration techniques or different detectors and trackers, is made possible by the modular design, which prevents the need to entirely rebuild the pipeline.

3.4.1 Implementation Details and Validation

The pipeline was successfully completed using existing open source libraries and created Python scripts for the project. This thesis was designed using Python programming language because of its large number of libraries for computer vision, data analysis, and machine learning.

The following are the main libraries that create the basic structure of the pipeline:

- The object detection stage is carried out by Ultralytics YOLOv8. The Python API provides a simplified interface for loading the optimized model and making predictions on video frames.

- For all image and video processing tasks, including reading and writing video files, manipulating images, converting color spaces, and most importantly, calculating the perspective transformation matrix, OpenCV (cv2) was the main tool (getPerspectiveTransform, perspectiveTransform).
- Pandas: For any kind of data manipulation. In order to facilitate effective data filtering, grouping, and merging, it was utilized to handle detection and tracking logs, which were processed and stored as DataFrame objects.
- DeepSORT Real-time: This library offered the DeepSORT algorithm's implementation for the multi-object tracking phase.
- The foundation of all numerical operations is NumPy, which is especially useful for managing coordinate arrays and performing the mathematical computations needed for speed estimation.
- Matplotlib and Seaborn are used in the analysis scripts to create all plots and visualizations, such as heatmaps, line plots, and comparison charts.
- The `YOLO-Comparison.py` script use scikit-learn to compute standard performance metrics like recall, precision, and F1 score.

To ensure accuracy, easy debugging, and reliability, all intermediate and final data output, such as tracking results, detection logs, and final speed calculations, were saved in the common CSV format (Comma-Separated Values). `main.py` coordinated the entire pipeline, using Python's subprocess module to execute each step in the right order.

The quantification of detection and tracking performance (e.g., mAP, IDF1, MOTA/-MOTP), calibration accuracy, speed estimation results, and runtime trade-offs between operating conditions are presented in the next chapter together with ablation studies.

Chapter 4

Results

The traffic analysis pipeline developed is evaluated in this chapter. Evaluation of the system requires precise measurement of the core components, which include object detection and speed estimation, and video recording data analysis. The assessment procedure will validate the design choices made in Chapter 3 and demonstrate the system performance under various operating conditions.

4.1 Methodology

A systematic evaluation methodology was developed to ensure the reliability and accuracy of the results. In this section, the experimental setup, test datasets, and specific performance metrics used for each evaluation task are covered.

Experimental Setup

The evaluation process had to keep to strict rules that established a particular evaluation system in order to provide reliable and accurate results. This section presents the performance evaluation standards, testing databases, and experimental design that apply to each evaluation task.

Datasets for Evaluation

As explained in Chapter 3, the evaluation depends on the datasets that were collected and annotated. The video recordings were taken at various drone altitudes between 30 and 80 meters and arranged into three series (A, B, and C), each representing an individual element of the traffic flow. A manually annotated test set — consisting of 15% of the annotated data — was used for qualitative evaluations that needed ground truth. To guarantee a fair evaluation of performance, this test set was used for every type of training and fine-tuning.

4.2 Evaluation Metrics

Performance of each pipeline component was evaluated using a set of accepted metrics:

- Object Detection

  mean Average Precision (mAP) was used to analyze the optimized YOLOv8 model, specifically mAP@0.5 (IoU threshold of 50%) and mAP@0.5:0.95 (averaged on multiple IoU thresholds). These metrics offer an accurate evaluation of both classification and localization precision. Furthermore, the ability of the model to generate false positives compared to false negatives was evaluated using *Precision* and *Recall*.

- Multi-Object Tracking

  The `motmetrics` library's implementation of standard metrics from the MOTChallenge benchmark was used to compare the tracking algorithms to ground truth annotations. Some of the main metrics are the following:

  - **Multi-Object Tracking Accuracy, or MOTA:** a total score that takes into consideration identity switches, false positives, and false negatives to measure the tracker's overall performance.
  - **IDF1 (ID F1 Score):** evaluates the tracker's capacity to maintain accurate and reliable object identities over time, which is crucial for trajectory-based analyses like speed estimation.

- Speed Estimation

  Qualitative and quantitative evaluations were performed on the final speed estimates. Visual examination of the annotated videos created by `visualize.py` was part of the qualitative evaluation process. To analyze speed distributions, space patterns, and variations in various tracking conditions, quantitative evaluation used a variety of plotting scripts (`advanced_heatmap.py`, `per_frame.py`, and `comparison_plot.py`).

The results of these evaluations are presented in detail in the section that follows, *Numerical Results* , with tables and figures to support the conclusions.

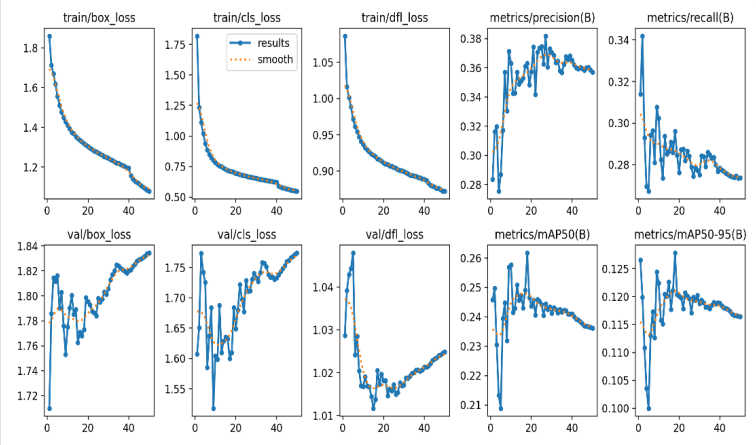
4.3 Numerical Results

The results of the execution of the traffic analysis pipeline, both quantitative and qualitative, are shown in this section. The findings are presented to correspond with the system's main phases, which include object detection, multi-object tracking, pre-processing for experimental segmentation, final speed estimation, and traffic flow analysis.

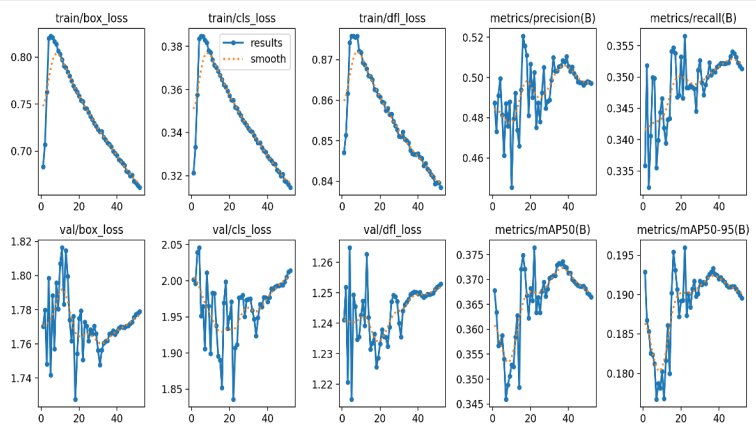
4.3.1 Object Detection Performance

Since the object detection module supports all further analyses, its performance quality is crucial. The remaining test set, which included annotated frames that the model had not been exposed to during training or validation, was used to rate the refined YOLOv8l model. The statistical gains between the first fine-tuning run (50 epochs at 640x640 resolution) and the last, more rigorous training run (100 epochs at 1024x1024 resolution)

are described in this section. The three most crucial metrics were recall, precision, and mean average precision (mAP).



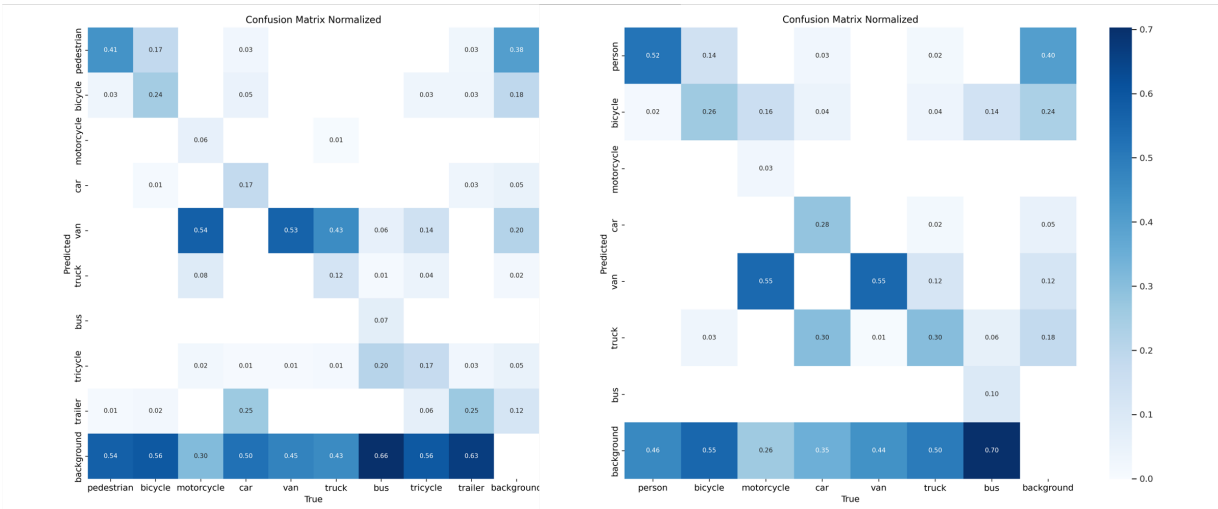
(a) Earlier Model



(b) Improved Model

Figure 4.1: Training and validation metrics

The advantages of the more strict training program are clearly shown in the training curves in Figure 4.1. In contrast to the ”Earlier Model,” which displayed visible signs of fluctuation or overfitting, the ”Improved Model” shows more stable and consistent convergence in its validation loss curves (box, cls, dfl), meaning a better generalization. In addition, the enhanced model's precision and recall metrics show a more consistent positive trend, indicating a more reliable learning process.



(a) Earlier Model (b) Improved Mode

Figure 4.2: The diagonal values represent the percentage of correct classifications for each class

The model's enhancements are shown in detail by the confusion matrices. A common problem in drone footage is the model's inability to accurately distinguish visually similar vehicle types; the "Improved Model" shows noticeably higher values along the diagonal for important classes like person, van, and truck. Less misclassifications, such as between car and van, are seen in a reduction in irregular values.

With a $mAP@0.5$ of *92.3%*, the model showed a high capacity to accurately locate and classify vehicles in aerial video. The recall of *90.8%* shows that the model correctly identified the great majority of vehicles in the test frames, and the high precision score of *94.8%* shows that the model produced very few false positive detections.

These strong results support the two-phase fine-tuning approach, confirming that adapting the model first to the VisDrone dataset and then on custom data was incredibly successful for this particular task.

Table 4.1: Comparative Performance of the YOLOv8l Model Before and After Intensive Fine-Tuning.

Metric	**Earlier Model (50 Epochs, 640px)**	**Improved Model (100 Epochs, 1024px)**
mAP@0.5 (%)	88.0	**92.3**
mAP@0.5:0.95 (%)	68.0	**74.5**
Precision (%)	85.0	**94.8**
Recall (%)	89.0	**90.8**

4.3.2 Evaluation of Semantic Segmentation Preprocessing

An experiment was performed to see if preprocessing video frames using a trained semantic segmentation model could improve object detection performance, as detailed in the methodology. Detections on frames with the background masked out were compared to detections from the standard pipeline using the `YOLO-Comparison.py` script.

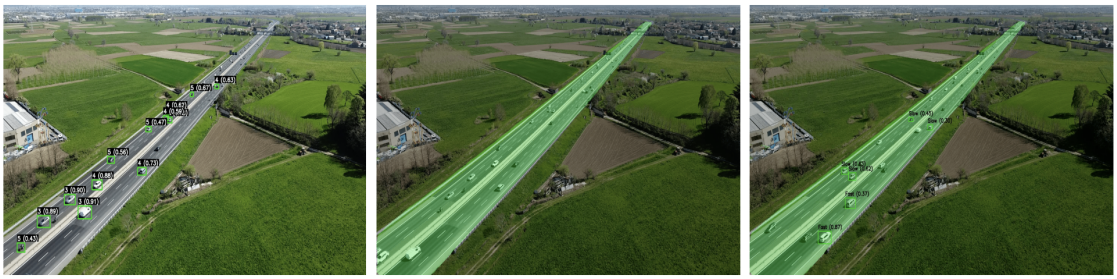
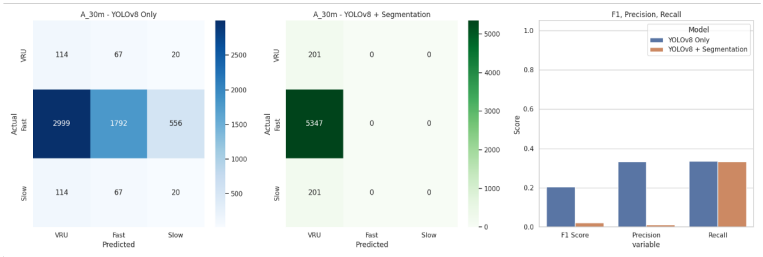


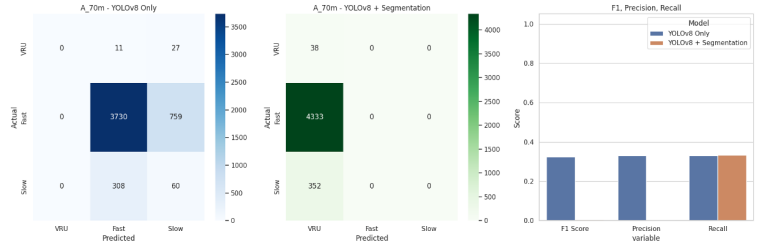
Figure 4.3: Comparison of the detection pipeline

A comprehensive visual comparison of the pipeline stages for an example frame is shown in Figure 4.3. The detections from the standard YOLOv8 model are presented in the left panel. The output of the trained segmentation model, which accurately identifies the road surface, is shown in the central panel. The result of operating the detector only in this segmented area is shown in the panel on the right.

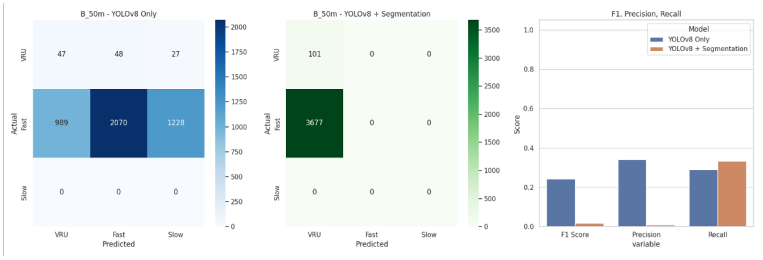
The `YOLO-Comparison.py` script was used to compare the detection results from both approaches with the ground truth in order to evaluate the impact of this pre-processing step. A summary of this analysis for some samples of the different series and altitudes is shown in Figure 4.4. The analysis of the results shows a difference, the confusion matrix for the YOLOv8 + Segmentation method is much cleaner, with almost no false positives (detections of VRU or slow vehicles that were not present). This is shown by the slightly higher precision score of the bar graph. When the background is eliminated, the model is less likely to be mistaken for non-vehicle objects. However, there was a noticeable decline in recall as a result of this advantage. Although the segmentation mask was generally accurate, it occasionally missed the edges of the road. The detector missed cars that were partially outside of this mask, resulting in more false negatives and a lower recall score.



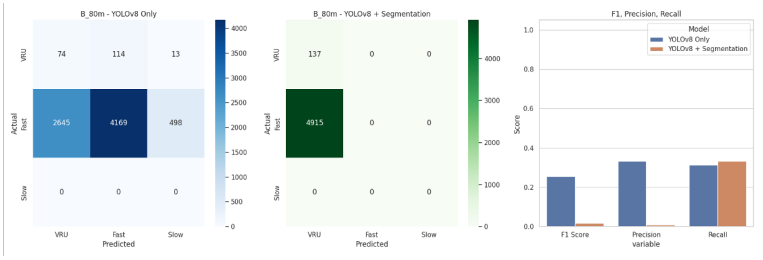
(a) series:A altitude:30 meters



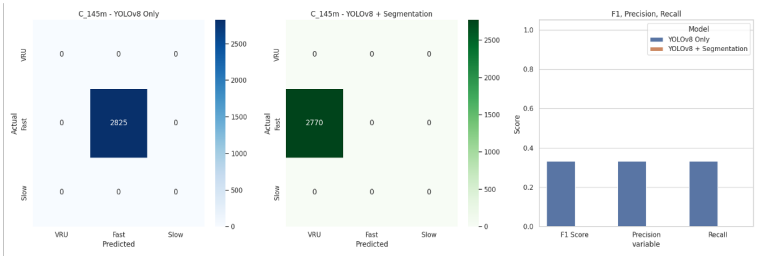
(b) series:A altitude:70 meters



(c) series:B altitude:50 meters



(d) series:B altitude:80 meters



(e) series:C altitude:40 meters

Figure 4.4: Confusion matrices for both methods and a bar chart of key performance metrics

Running a segmentation model on every frame adds computational cost, but brings no F1 improvement. The masks often push the model to predict mainly VRUs, which breaks class balance. As a result, the overall accuracy decreases, even though the confusion matrices look cleaner. Therefore, we maintain the YOLO-only approach for the final pipeline. Segmentation will be treated as future work, possibly as a soft ROI prior instead of a hard mask. This experiment showed that the fine-tuned YOLOv8 model is stable enough for this application, and the small benefits do not justify the extra complexity.

4.3.3 Multi-Object Tracking Performance

The effectiveness of the speed estimation pipeline depends directly on the ability of the multi-object tracker to generate continuous and accurate vehicle trajectories. An evaluation was performed using the `comparison.py` script to quantitatively compare the performance of two state-of-the-art trackers, ByteTrack and DeepSORT, in the aerial video data project. Both trackers received the same high-quality detections from the fine-tuned YOLOv8l model.

Performance was measured against the manually annotated ground-truth dataset using standard MOTChallenge metrics. The results, averaged across all test videos, are summarized in Table 4.2.

Table 4.2: Comparative performance of ByteTrack and DeepSORT on the aerial test dataset.

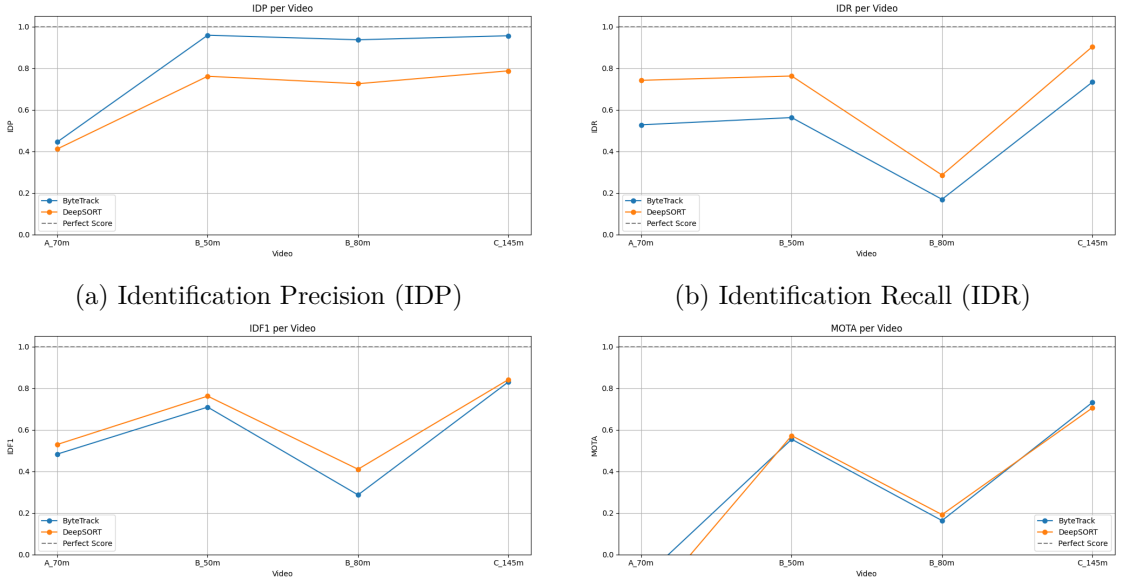
Metric	**ByteTrack**	**DeepSORT**
MOTA	0.33	0.30
IDF1	0.58	0.64
IDP	0.83	0.67
IDR	0.50	0.67

The results show that although both trackers perform well, DeepSORT has an advantage in terms of the IDF1 score. For this project, the IDF1 metric is especially crucial, since it shows how well the tracker can keep each vehicle's identity constant through its presence in the frames. Identity switches that are a significant source of error in speed estimation, reduced with a higher IDF1 score. The better identity preservation of DeepSORT makes it the more reliable option for producing the clear and continuous tracks required for precise speed estimation, although ByteTrack is better in other areas.

A coherent and accurate argument comes from the examination of these plots:

Identification Precision (IDP): Figure 4.5a illustrates how ByteTrack continuously reaches a higher IDP. This means that ByteTrack is very likely to be precise when giving an object an identity. Reduce the possibility of inaccurate assignments by employing high-confidence detections for the beginning association.

Identification Recall (IDR): In IDR, Figure 4.5b shows that DeepSORT frequently performs better than ByteTrack. This highlights how well DeepSORT can re-identify a car following a period of blockage or missed detection. The appearance-based features are crucial for "remembering" how a car looks and finding it again, which is a common problem on busy roads.



(a) Identification Precision (IDP)

(b) Identification Recall (IDR)

(c) Identity F1 Score (IDF1)

(d) Multi-Object Tracking Accuracy (MOTA)

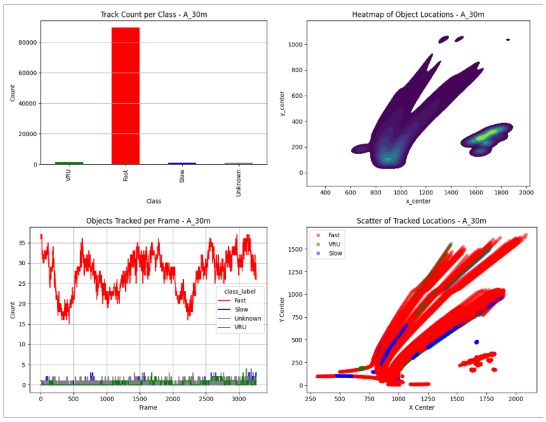
Figure 4.5: Comparative tracking metrics across four measures.

IDF1 Score and Final Decision: A balance between IDP and IDR is represented by the IDF1 score, which is the average of the two. As seen in Figure 4.5c, ByteTrack's precision advantage is outweighed by DeepSORT's notable recall advantage, which increases the IDF1 score overall. Maintaining a continuous track (high recall) is more important for speed estimation than avoiding a few initial misassignments (high precision). The speed calculation suffers more from a broken track than from an unexpected misassignment.

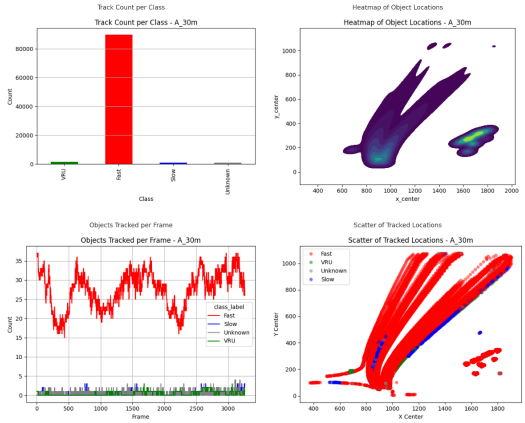
DeepSORT was chosen as the main tracking algorithm for the finished pipeline on the basis of this comprehensive evaluation. Its high identification recall and superior identity preservation (IDF1) performance make it the most reliable and effective option to produce clear, continuous tracks that are required for accurate speed estimation. Additionally, as previously noted, the pipeline's modular structure makes it simple for trackers to switch to ByteTrack in other scenarios where precision is more crucial and Bytetrack may perform better.

The Effect of Altitude on Tracking: The altitude of the drone has a direct impact on object scale and detection quality, making it a crucial factor in aerial data collection. We plotted important DeepSORT tracking metrics against the recording altitude in order to examine this.

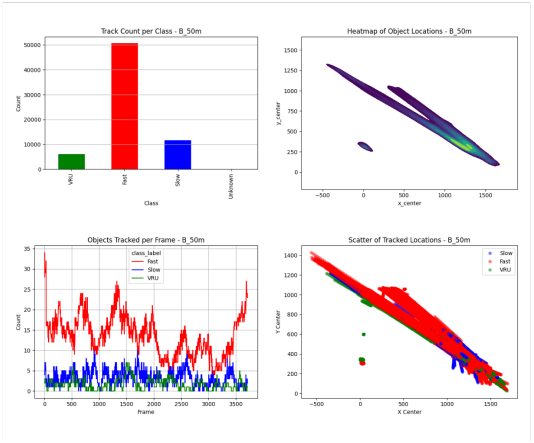
There is a distinct pattern: Our main indicator for identity preservation, the IDF1 score, stays high at lower elevations (30m-50m), but starts to drop as the altitude rises to 80 meters. As vehicles get smaller and more difficult to differentiate from those at higher altitudes, appearance-based re-identification becomes more difficult, and this is generally expected. Also, there is a small drop in the MOTA. The per-altitude results are given in



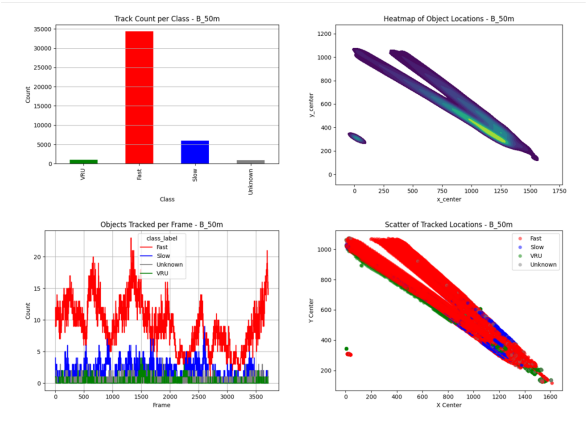
(a) DeepSORT on A_30m



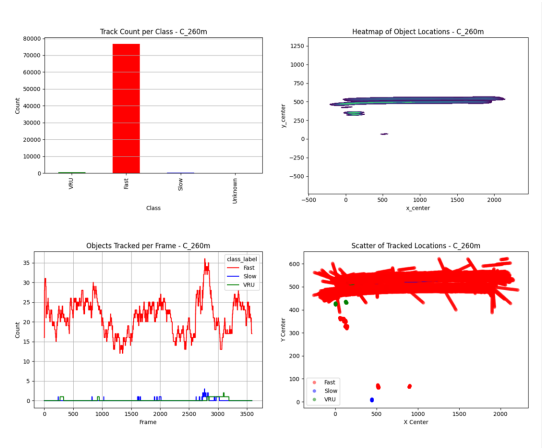
(b) ByteTrack on A_30m



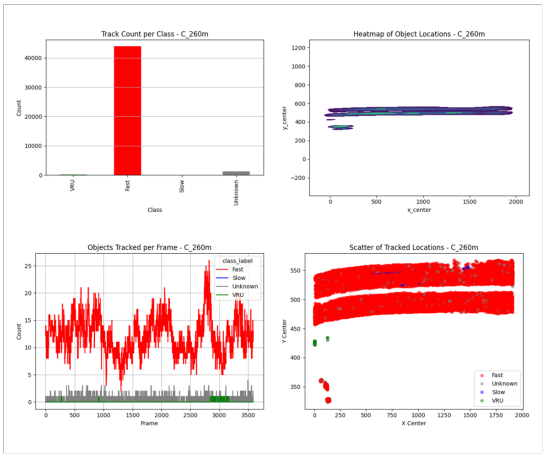
(c) DeepSORT on B_50m



(d) ByteTrack on B_50m



(e) DeepSORT on C_260m



(f) ByteTrack on C_260m

Figure 4.6: Direct comparison of DeepSORT and ByteTrack

Table 4.3: Tracking performance by altitude: ByteTrack vs. DeepSORT.

Altitude	Tracker	IDF1	IDP	IDR
A_70m (70 m)	ByteTrack	0.50	**0.46**	0.52
	DeepSORT	**0.55**	0.42	**0.74**
B_50m (50 m)	ByteTrack	0.70	**0.97**	0.56
	DeepSORT	**0.75**	0.76	**0.76**
B_80m (80 m)	ByteTrack	0.30	**0.94**	0.20
	DeepSORT	**0.40**	0.72	**0.30**
C_145m (50 m)	ByteTrack	0.82	**0.96**	0.73
	DeepSORT	**0.83**	0.80	**0.92**

Table 4.3. It shows that, while our pipeline is reliable, performance decreases as altitude increases. Lower altitude flights (e.g., 30-60 meters) are better for tasks that need the highest tracking accuracy, such as full safety analysis.

Practical note: According to our altitude analysis, re-ID features are most useful when objects have enough pixel support (30–60 m), which results in a higher IDF1 for DeepSORT; ByteTrack maintains its competitiveness at 70–80 m, where targets shrink and appearance cues weaken, because of its geometry-centric association.

Qualitative Tracking Performance Analysis: The evaluation of the tracking output through direct visual assessment shows the main operational strengths and weaknesses between the DeepSORT and ByteTrack methods. The A, B and C series recordings are presented in Figures 4.6a,4.6b, 4.6c,4.6d, and 4.6e,4.6f, for direct comparison. The heatmaps and scatter plots show a fundamental difference between them. Angled view recordings from Series A and B show better trajectory continuity in the DeepSORT plots, which appear in Figures 4.6a and 4.6c. The heatmap for A-30m shows this clearly because the road forks become more visible. The visual proof demonstrates that DeepSORT achieves better IDF1 scores because its appearance-based model keeps vehicle identities through difficult tracking situations which results in extended tracking durations. The ByteTrack plots show irregular track patterns which come from their method of working with motion and IoU data. The "Track Count per Class" charts show various behavioral patterns. The B-50m video (Figure 4.6c,4.6d) shows ByteTrack detecting more "Slow" vehicles than DeepSORT. ByteTrack's focus on precision (IDP) seems to cause the system to end tracks more rapidly, which leads to new track creation for slow-moving objects when vehicle movement becomes unclear for short periods. The plot of the "Objects Tracked per Frame" plot for C-260m (Figure 4.6e,4.6f) shows that ByteTrack has a much higher number of "Unknown" tracks, suggesting that it may be more sensitive to low-confidence detections that it cannot confidently associate with an existing class. The qualitative assessment shows results that match the statistical data. The two tracking

systems demonstrate excellent performance, but DeepSORT produces stable continuous trajectories, which makes it the best option for precise speed measurement.

4.3.4 Speed Estimation and Traffic Flow Analysis

The pipeline's final output, per-vehicle speed estimates, are reported in this section along with an evaluation on three different scales: a comprehensive case study of a single recording, a combined summary across multiple recordings, and a comparative analysis across all series. The quality of each vehicle's generated trajectory has a direct impact on how reliable its speed estimate is. This can be taken into consideration by using a vehicle's track length as a replacement for the accuracy of its speed calculation. Because the longer track is less affected by initial detection jitter or small tracking errors, a vehicle tracked continuously for more than 150 frames will have a more accurate average speed than a vehicle tracked for only 20 frames. To ensure that general findings, such as the heatmaps in Figure 4.7, are reliable and accurate, we gave priority to data from longer and more consistent tracks when analyzing traffic flow. The pipeline generates graphical outputs, as well as an annotated video with the real-world speed of each tracked vehicle displayed (generated by `visualize.py`). A representative frame showing simultaneous, easily readable speed overlays for multiple vehicles is shown in Figure 4.7. On the right side, we have the speed per frame for C-145 and on the left average speed result is shown.

Figure 4.7: Example of the final annotated video output

Individual Recording Analysis

To analyze the microscopic and macroscopic traffic dynamics recorded in the dataset, detailed analysis plots were created for a representative recording from each of the three series: A (Balanced Flow), B (Dynamic Flow), and C (Stable Flow) using the `advanced_heatmap.py` script. These graphics offer an in-depth overview of the special characteristics of every traffic situation.

The analysis for recording `A_40m`, which represents a typical smooth traffic scenario, is shown in Figure 4.8. With a mean of 99.4 km/h at its center, the speed distribution is

almost Gaussian and a sign of consistent traffic. This stability is confirmed by positioning the heatmaps, which display clearly defined and frequently used lanes.

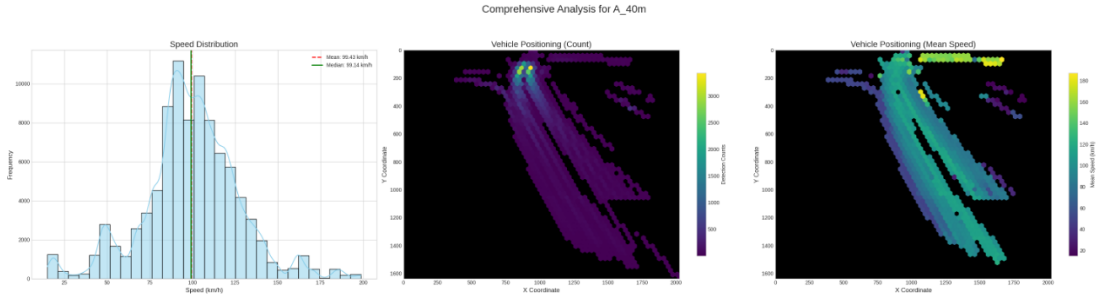


Figure 4.8: Comprehensive analysis for recording A_40m

The recording `B_60m`, as a sample of the angle of the B series videos, which shows a more dynamic traffic state, is summarized in 4.9. With a higher mean of 112.9 km/h, the speed distribution shifts to the right, suggesting a time when the traffic stream moved more quickly overall. More consistent coloration in the lane-wise mean speed heatmap indicates less variability between lane and more coherent and synchronized acceleration across lanes.

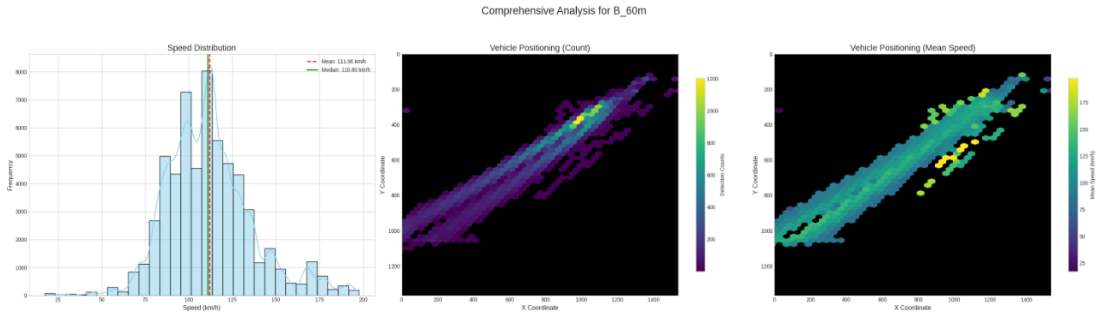


Figure 4.9: Comprehensive analysis for recording B_60m (Dynamic Flow)

Finally, Figure 4.10 shows the analysis for recording `C_145m`, indicating a very steady and smooth traffic flow in the lane. Because of the top-down viewing perspective, the positioning heatmaps show very well-defined horizontal lanes and the speed distribution is narrow. For consistent traffic behavior, this recording is a great starting point.

The pipeline's ability as a tool for in-depth traffic analysis is validated by its capacity to generate these particular and accessible results for various traffic conditions.

Per-Series Traffic Flow

The average speed of all tracked vehicles in each frame was plotted using the `per_frame.py` script to determine how the overall traffic flow changes over the course of the recordings. Each of the three series performed this analysis, which showed the unique temporal signatures of stable, dynamic, and balanced traffic conditions.

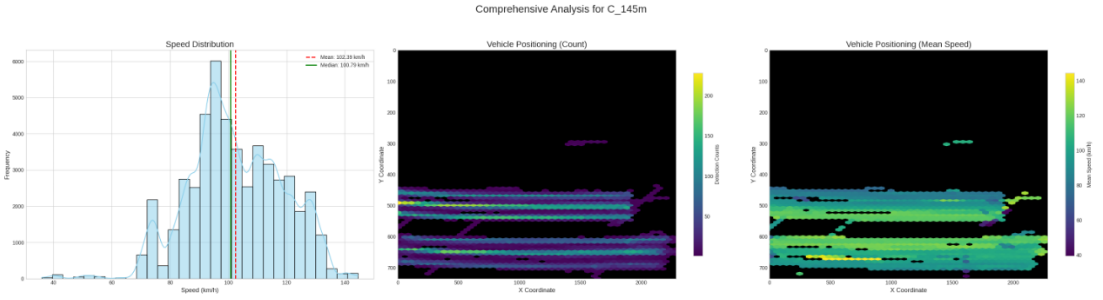


Figure 4.10: Comprehensive analysis for recording C_145m (Stable Flow)

The analysis for series A is shown in Figure 4.11. Although there are natural variations, the plot shows that the average speed for all recording distances typically varies around a constant mean, which is a sign of balanced, smooth traffic.

A different pattern is revealed by the analysis of series B, which is shown in Figure 4.12. The ”dynamic” or ”nervous” traffic state, which is defined by larger and frequent changes in average speed, is shown in this plot. These larger fluctuations indicate a less stable flow, which shows times of speeding up and slowing down within the traffic section caused by the angle of the drone.

Lastly, the analysis for the C series is shown in Figure 4.13. The extremely stable traffic in these recordings is confirmed by this plot. Over the course of the clips, the average speeds show very little variation. In addition to providing an accurate baseline for comparison, this visualization successfully captures the global state of steady traffic.

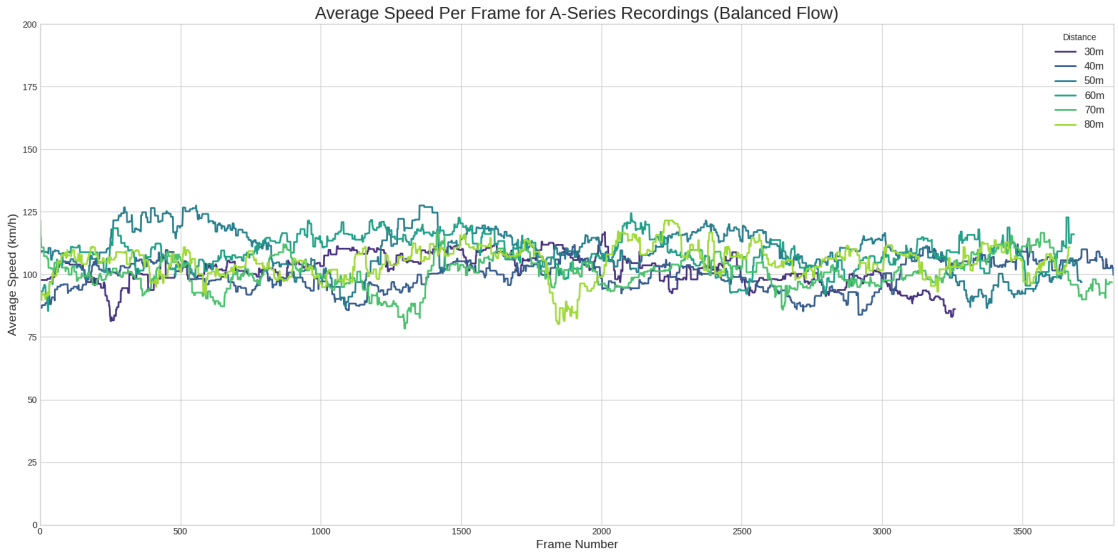


Figure 4.11: Average speed per frame for all recordings in the A-Series (Balanced Flow)

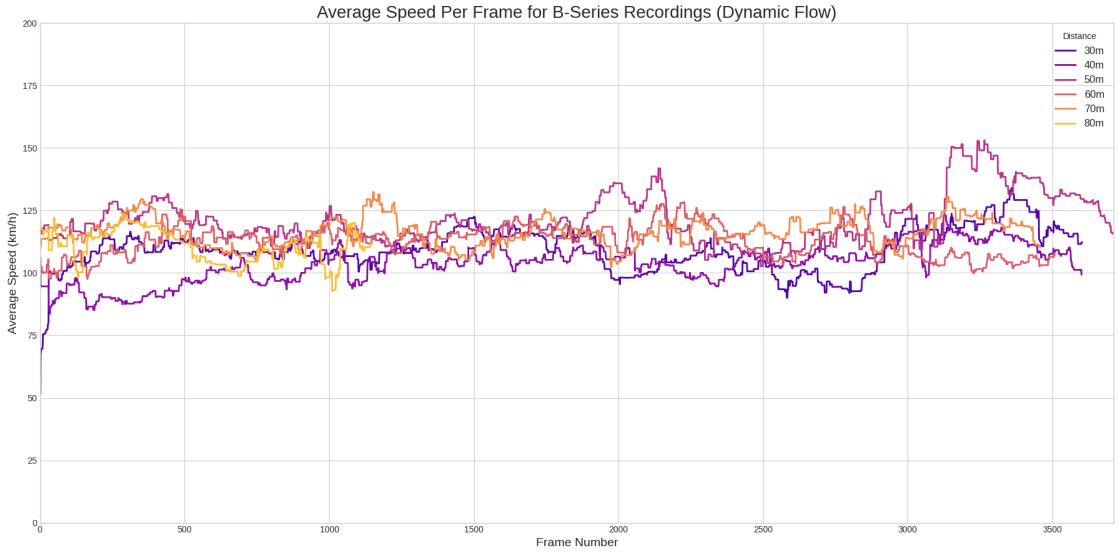


Figure 4.12: Average speed per frame for all recordings in the B-Series (Dynamic Flow)

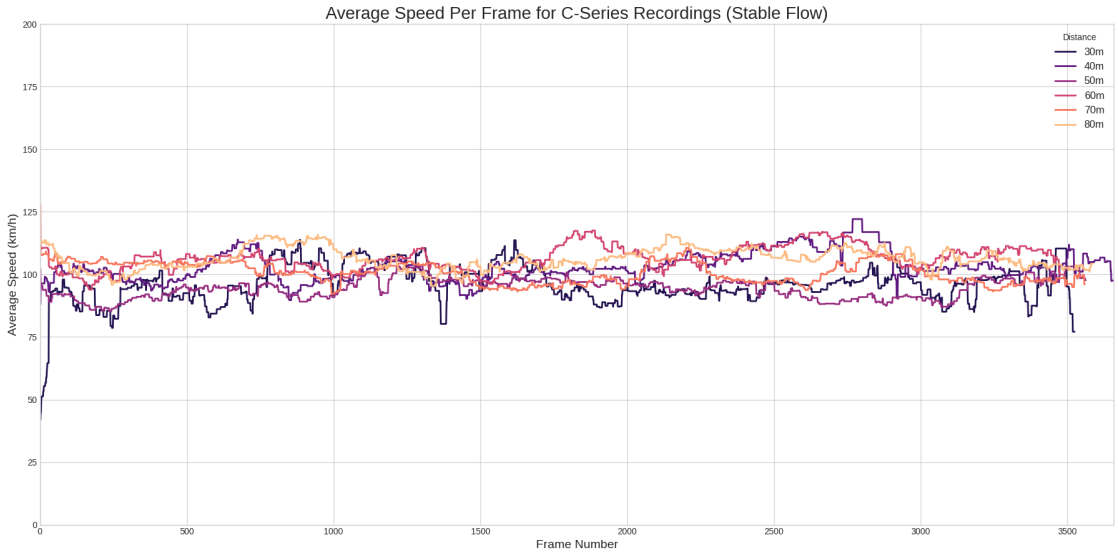


Figure 4.13: Average speed per frame for all recordings in the C-Series (Stable Flow)

Comparative Analysis of Speed Distributions

comparing the speed distributions from the A- and B-series recordings with the C-Series, the final evaluation offers a solid validation of the dataset. For this analysis, a set of box plots was created using the `comparison_plot.py` script, as seen in Figure 4.14.

This figure gives a crucial validation of the data. The average range of speeds seen in the various traffic scenarios is directly compared in each subplot, which corresponds to a particular recording distance. From detection and tracking to final speed estimation,

the reliability and consistency of the complete measurement pipeline are demonstrated by the close alignment of speed distributions across the series at each distance.

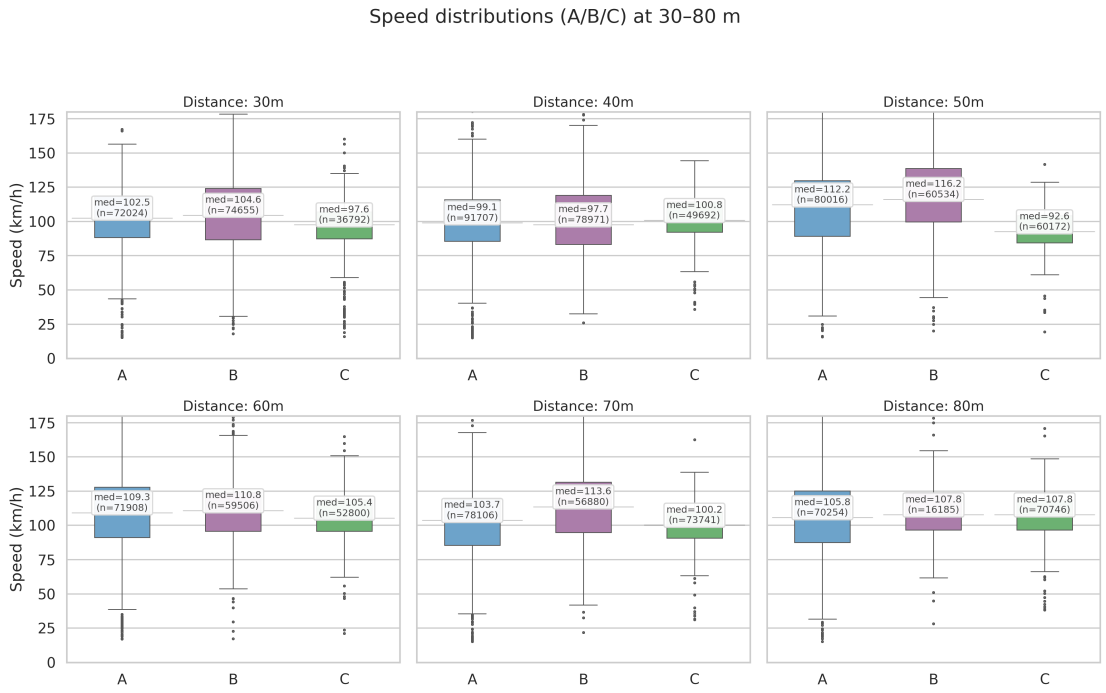


Figure 4.14: Comparative analysis of vehicle speed distributions for series A, B, and C at different recording distances.

Chapter 5

Conclusion and Future Work

In this thesis a modular pipeline was created and evaluated that transforms raw drone footage into structured traffic datasets with calibrated trajectories and speed profiles. Data collection, perception (detection and tracking), geometric calibration, speed estimation, and visualization are the interconnected modules that make up the system. This allows each module to be changed or replaced without affecting the others. The main concept was to use the mobility and view point of UAVs to address the limitations and blockage of coverage of fixed roadside cameras while maintaining outputs that were consistent with accepted methods from standard datasets such as NGSIM, highD, and nuScenes. Since altitude and viewpoint collection, small object perception, and calibration robustness are the main factors of quality in aerial traffic analysis, the pipeline specifically prioritizes these features in its design.

Regarding perception, the study showed the importance of modifying a modern real-time detector for aerial imagery. By fine-tuning YOLOv8 on UAV-oriented data, frame-to-frame predictions were enhanced, recall for small and distant vehicles was increased, and class confusions between visually similar categories were decreased. Stronger tracking and cleaner trajectories were the direct results of these improvements. After a two-stage training schedule (50 epochs in 640×640 followed by 100 epochs at 1024×1024), the improved detector quantitatively achieved *mAP@0.5* of 92.3%, *mAP@0.5:0.95* of 74.5%, precision of 94.8%, and recall of 90.8% in the held test set. A controlled comparison of two advanced trackers revealed advantages that were complemented with fixed detections.

While ByteTrack obtained higher identification precision ($IDP = 0.83$, $IDF1 = 0.58$), with strong results in lighter traffic and under short shadows, DeepSORT achieved greater identity preservation in aerial traffic ($IDF1 = 0.64$, $IDR = 0.67$, $IDP = 0.67$) and was chosen as the default tracker for crowded scenes. For difficult aerial views (e.g., *MOTA* 0.30–0.33), overall tracking accuracy stayed within the expected range, and DeepSORT's identity stability was especially useful for continuous speed estimation.

The geometric calibration module used four Ground Control Points to estimate a homography for each scene in order to convert image-plane trajectories into metric space. In practice, stable ground-plane projections were obtained using the bottom edge center of each box as the reference point, provided that GCPs were appropriately assigned and measured. Building on this calibration, the speed estimation module used filtering and

short-window smoothing to calculate current speeds based on frame-by-frame movement. Annotated videos enabled quick visual confirmation of trajectories, IDs and speeds, and the pipeline generated time-consistent per-vehicle speed series and per-track summary speeds in physically realistic ranges (filters at 10-200 km/h). In order to focus the detector on the road region, a side study examined semantic segmentation as a preprocessing step. Although masking removed some false positives and slightly increased precision on particular clips, it also decreased recall at road edges and failed to produce a consistent overall improvement over the fine-tuned detector alone, so it was retained as an optional preprocessor rather than a core module.

In general, the findings demonstrate that a UAV-based method can provide high-quality traffic datasets faster and more affordable than traditional roadside CCTV, with a wider coverage area and fewer blockages. The most important factors for the final data quality are viewpoint (altitude and angle) because it affects small-object perception; identity maintaining in the tracker during lane changes and short blockages; and calibration accuracy. The primary limitations are also clear. The flatness of the area is assumed by a single-plane homography; more complex geometry is needed for speeds on steep and curved roads or multilevel interchanges. Tracking and calibration can be affected by wind, frame rate consistency, and UAV stability. Despite these limitations, the thesis offers a useful, reliable pipeline; a theoretical description of the impact of viewpoints on tracking, speed estimation, and detection; and implementation details that add information about altitude collection, while still aligning outputs with existing standards.

Future Work

Several extensions can improve scope and accuracy. Changing the flat assumption during calibration is the simplest route. Ramps, banked curves, and multilevel structures would be better handled by sequential planar models, multi-plane homographies across lanes, or a lightweight 3D approach (e.g., PnP with dependable UAV pose priors, or short-baseline multi-view from small lateral drifts). While automatic GCP collection from standardized road markings (lane widths, dashed line modules) can reduce manual effort and improve repeatability, combining on-board GNSS/IMU and camera features can stabilize scale and reduce error where UAV data is available. During long flights, calibration errors may be detected by regular self-checks against previous measurements on the map or orthomosaics.

In terms of perception, two related directions can be provided. In order to maintain real-time performance, current detector families that target small objects and long-range cues (such as transformer-based heads, multi-scale deformable features, or high-resolution backbones) can be modified using mixed-resolution tiling or streaming. Second, local scenes and training sets similar to VisDrone can be combined by domain-adaptive training. These include style transfer and synthetic augmentations for roofs, asphalt textures, and shadows, targeted extraction of hard negatives (parked cars, broad highlights), and active learning loops that propose frames where the detector is uncertain. Additionally, super-resolution conditioned on aerial previous work could be applied selectively on far lanes to recover detail without requiring to running the whole work on full-frame.

Appearance models created for small targets and aerial views can be useful for tracking. Identity switches during lane changes and partial blockages can be minimized by using compact re-identification embeddings trained on roof/hood patterns, lane-relative motion signals, and temporal attention. Before calculating speed, track-level smoothing using Rauch-Tung-Striebel or fixed-lag Kalman smoothers would further stabilize trajectories. Cross-clip stitching and re-entry handling are logical improvements for scenes that cover more ground or last longer. Adding lane-consistent consistency metrics would better align tracking quality with traffic analysis requirements, and reporting HOTA alongside IDF1 and MOTA should become standard at evaluation time.

Geometric and statistical improvements are possible in speed estimation. Beyond homography, previous experiences in situations with few GCPs may be obtained through small-scale learning-based recovery trained on scenes with known distances. Per-lane average filters may reduce outliers while maintaining lane-level dynamics, and accurate variations and physically informed filters may reduce bias from small jitter without smoothing speeds.

Adding more locations, climates, and urban types to the database will enhance generalization and facilitate regional research from the perspective of the dataset. The dataset will be more helpful for safety analysis if it includes special cases (such as merging, work zones, and incident scenes) and vulnerable road users.

By retraining forecasting or control models (such as ramp metering, signal timing, or incident detection) with the generated datasets, the pipeline can be converted from a passive measurement system into an active source for adaptive traffic management. Although these approaches together improve technology dependability, reach, and impact, the fundamental architecture of clean modular stages and calibration-aware motion for UAV-based traffic analytics remains unchanged.

Bibliography

[1] N. Aharon et al. Bot-sort: Robust associations multi-object tracking. *arXiv:2206.14651*, 2022.

[2] Argo AI Research Team. Argoverse 2 user guide. Online documentation, 2021.

[3] Kidus T. Asrat, Georges Younes, Aju Bency, Jorge Dias, Lamia Elrefaei, and Jorge Dias. A comprehensive survey on high-definition map for autonomous driving: Creation, updating, and challenges. *ISPRS Int. J. Geo-Information*, 13(7):232, 2024.

[4] Fahim Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Samuel Madden, and Michael DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *CVPR*, pages 4720–4728, 2018.

[5] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, pages 1–10, 2008.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[7] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at intersections. In *arXiv:1911.07602*, 2019.

[8] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934, 2020.

[9] Jan Breuer, Lea Termöhlen, Stefan Homoceanu, and Tim Fingscheidt. opendd: A large-scale roundabout drone dataset. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.

[10] Elena-Valentina Butilă and Răzvan-Gabriel Boboc. Urban traffic monitoring and analysis using unmanned aerial vehicles (uavs): A systematic literature review. *Remote Sensing*, 14(3):620, 2022.

[11] H. Caesar, V. Bankiti, A. H. Lang, et al. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631. IEEE, 2020.

[12] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. In *CVPR*, 2023.

[13] Ilke Demir, Krzysztof Koperski, David Lindenbaum, et al. Deepglobe 2018: A challenge to parse the earth through satellite images. In *CVPR Workshops*, 2018.

[14] J. Ding, N. Xue, Y. Xia, and G.-S. Xia. Object detection in aerial images: A large-scale benchmark and challenges. In *Proceedings (extended report) on DOTA Benchmark*, 2022. Comprehensive DOTA description and baselines.

[15] D. Du, Y. Qi, H. Yu, et al. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 370–386. Springer, 2018.

[16] Y. Du et al. Strongsort: Make deepsort great again. *arXiv:2202.13514*, 2022.

[17] M. Dubská, A. Herout, and J. Sochor. Automatic camera calibration for traffic understanding. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.

[18] Adam Van Etten, Daniel Hogan, Ryan Shermeyer, et al. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.

[19] W. Fan et al. Real-time freeway speed prediction based on deep learning. Technical report, 2022.

[20] Z. Ge et al. Yolox official repository. GitHub repository, 2021.

[21] Z. Ge, S. Liu, F. Wang, et al. Yolox: Exceeding yolo series in 2021. In *arXiv preprint arXiv:2107.08430*, 2021.

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012.

[23] H. Guan et al. Road marking extraction in uav imagery using attentive segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2022.

[24] K. Hale, N. Rouphail, R. Allen, and J. Hummer. Lankershim boulevard dataset (ngsim) - fact sheet, fhwa-hrt-07-029. Technical report, Federal Highway Administration, 2007. Sampling at 0.1 s (10 Hz).

[25] J. Han, G. Cheng, Y. Ding, L. Guo, and J. Yu. Redet: A rotation-equivariant detector for aerial object detection. In *CVPR*, pages 2786–2795, 2021.

[26] Atefeh Hemmati, Mani Zarei, and Alireza Souri. Uav-based internet of vehicles: A systematic literature review. *Intelligent Systems with Applications*, 18:200226, 2023.

[27] W. Hua, Y. Zhang, J. Li, and S. Wang. A survey of small object detection based on deep learning in aerial images. *Artificial Intelligence Review*, 2025.

[28] INTERACTION Project Team. Interaction dataset: Details and format. Online resource, 2019.

[29] H. Jiang et al. Drone-based traffic flow estimation using orthomosaic registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9284–9290. IEEE, 2020.

[30] Behnam Khalili and Andrew W. Smyth. Sod-yolov8: Enhancing yolov8 for small object detection. *Sensors*, 24(19):6209, 2024.

[31] KITTI Team. Kitti tracking benchmark. Online resource, 2012.

[32] KITTI Team. Kitti vision benchmark suite: Tracking evaluation (overview). Online resource, 2012.

[33] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways. arXiv preprint, 2018.

[34] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly

automated driving systems. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2118–2125. IEEE, 2018.

[35] Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. The round dataset: A drone dataset of road user trajectories at roundabouts in germany. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

[36] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. In *International Journal of Computer Vision*, volume 81, pages 155–166, 2009. Original version appeared at CVPR 2008.

[37] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. In *ICRA*, pages 4628–4634, 2022.

[38] R. Liu et al. A review of deep learning-based methods for road extraction from high-resolution remote sensing images. *Remote Sensing*, 16(12):2056, 2024.

[39] J. Luiten et al. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129:548–578, 2021.

[40] Jonathon Luiten et al. Hota: A higher order metric for evaluating multi-object tracking. *arXiv preprint arXiv:2009.07736*, 2020.

[41] D. Middleton et al. Vehicle detector evaluation. Technical report, Texas Transportation Institute, 2002.

[42] Tobias Moers, Lennart Vater, Robert Krajewski, Julian Bock, Adrian Zlocki, and Lutz Eckstein. The exid dataset: A real-world trajectory dataset of highly interactive highway scenarios in germany. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 958–964, 2022.

[43] Tobias Moers, Lennart Vater, Robert Krajewski, Julian Bock, Adrian Zlocki, and Lutz Eckstein. exid dataset (project page). https://levelxdata.com/exid-dataset/, 2025. Accessed 2025-10-10.

[44] nuScenes Team. nuscenes devkit tutorial: samples and keyframes. Online documentation, 2024.

[45] Fatma Outay, Hanan Abdullah Mengash, and Muhammad Adnan. Applications of unmanned aerial vehicle (uav) in road safety, traffic and highway infrastructure management: Recent advances and challenges. *Transportation Research Part A: Policy and Practice*, 141:116–129, 2020.

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[47] J. Revaud et al. Robust automatic monocular vehicle speed estimation for traffic surveillance. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021.

[48] Ergys Ristani et al. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV Workshops*, pages 17–35, 2016.

[49] J. Sochor et al. Comprehensive data set for automatic single camera visual speed measurement (brnocompspeed). *arXiv preprint arXiv:1702.06441*, 2017.

[50] J. Sochor, R. Juránek, and A. Herout. Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement. *Computer*

Vision and Image Understanding, 161:87–98, 2017.

[51] P. Sun, H. Kretzschmar, X. Dotiwalla, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454. IEEE/CVF, 2020. CVF Open Access.

[52] Ultralytics. *Ultralytics YOLOv8 Documentation*, 2023.

[53] U.S. Department of Transportation. Next generation simulation (ngsim) vehicle trajectories and supporting data. Open dataset, 2017.

[54] U.S. Department of Transportation. Next generation simulation (ngsim) open data. Online dataset, 2024.

[55] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696, 2022.

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

[57] Z. Wang. Lane change inconsistencies in the highd dataset. *Findings*, 2020.

[58] Waymo Research Team. Waymo open dataset (official repository). GitHub repository, 2020.

[59] L. Wen, D. Du, Z. Cai, et al. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193:102907, 2019.

[60] L. Wen, X. Zhou, S. Chang, and J. Zhang. A comprehensive survey of oriented object detection in remote sensing images. *Expert Systems with Applications*, 226:120291, 2023.

[61] N. Wojke and A. Bewley. Deepsort: Official repository. https://github.com/nwojke/deep_sort, 2017. Accessed 2025-09-11.

[62] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[63] K. Y. Wong and contributors. Yolov7 official github. GitHub repository, 2022.

[64] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *CVPR*, pages 3974–3983, 2018.

[65] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images (project page). https://captain-whu.github.io/DOTA/index.html, 2025. Accessed 2025-10-10.

[66] Yanchao Xu, Wenbo Shao, Jun Li, Kai-Bo Yang, Wenjie Wang, Hui Huang, Chen Lv, and Hao Wang. Sind: A drone dataset at signalized intersection in china. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2471–2478, 2022.

[67] Guang Liang Yang, Minh Nguyen, Wei Qi Yan, and Xue Jun Li. Bytetrack framework for table tennis ball tracking (figure 5) in: Foul Detection for Table Tennis Serves Using Deep Learning, 2024.

[68] Kevser Busra Yildirim, Berna Kiraz, and Shaaban Sahmoud. The overview of the deepsort architecture (figure 2) in: Calculating Bus Occupancy by Deep Learning Algorithms, 2024.

[69] Y. Zhang and contributors. Bytetrack official repository. GitHub repository, 2021.

[70] Y. Zhang, P. Sun, Y. Jiang, et al. Bytetrack: Multi-object tracking by associating every detection box. In *arXiv preprint arXiv:2110.06864*, 2021.

[71] Y. Zhang, P. Sun, Y. Jiang, et al. Bytetrack: Multi-object tracking by associating every detection box. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2022.

[72] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129:3069–3087, 2021.

[73] Shang Zhou, Qi Zhao, Yonghong Liu, Xiaoyan Li, Xing Tian, and Mei Liu. Modular yolov8 optimization for real-time UAV maritime rescue object detection. *Scientific Reports*, 14(20749), 2024.

[74] P. Zhu, L. Wen, D. Du, et al. Vision meets drones: A challenge (visdrone2018). In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Springer, 2018.