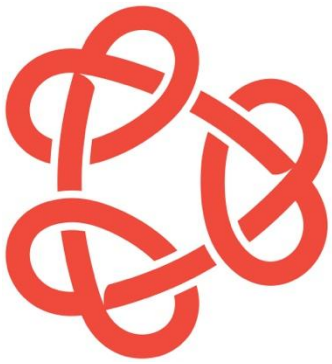


University of Staffordshire

FINAL YEAR PROJECT

COMP60011

VEHICLE SPEED ESTIMATION USING COMPUTER VISION AND DEEP LEARNING

DAMILOLA ADEDAYO

23021899

Supervisor: SAEED SHIRY GHIDARY

08/05/2026

TABLE OF CONTENTS

Contents

ABSTRACT

Accurate vehicle speed estimation is critical to intelligent transportation systems and road safety monitoring, yet traditional sensor-based approaches (radar, inductive loops, and LiDAR) are costly, infrastructure-dependent, and limited in scalability. This project investigates a calibration-free, vision-based alternative using monocular video footage captured from a fixed roadside camera.

The aim of the project is to design an algorithm capable of detecting, tracking, and estimating vehicle speed using visual data alone. The proposed system comprises four sequential stages: vehicle detection using YOLOv8n, multi-object tracking using DeepSORT, bounding box feature extraction over a one-second exit-anchored observation window, and speed regression using a tuned XGBoost model. No camera calibration parameters are used at any stage; instead, the model learns the implicit relationship between vehicle tracking behaviour and real-world speed directly from labelled training data.

The system is trained and evaluated on the VS13 dataset, comprising 400 video clips across 13 vehicle models. A folder-level train/held-out split ensures that evaluation vehicles are entirely unseen during training. On a held-out set of three vehicles representing a range of apparent sizes, the final system achieves a mean absolute error of 3.66 km/h, RMSE of 4.88 km/h, and R^2 of 0.949, with 93.4% of predictions falling within 10 km/h of ground truth. These results are broadly competitive with calibration-free published work on the same dataset.

Key findings include the importance of normalising width growth features to resolve small-vehicle generalisation failure, and the validation that feature importance reflects physically meaningful signal consistent with the near-POV camera geometry.

ACKNOWLEDGEMENTS

I would like to thank my project supervisor, Saeed Shiry Ghidary, for his guidance and support throughout this project. Their feedback during meetings was instrumental in shaping the direction of this work.

I would also like to express my sincere gratitude to my mother, whose encouragement and willingness to review drafts meant a great deal.

1.0 INTRODUCTION

1.1 Research Background

Accurate vehicle speed estimation plays a critical role in intelligent transportation systems, road safety monitoring, and automated traffic management (Djukanovic et al., 2022). Knowing the speed of vehicles in real time enables enforcement of speed limits, detection of dangerous driving behaviour, and the generation of traffic flow data that informs infrastructure planning. As road networks grow in complexity and traffic volumes increase, the demand for scalable, reliable speed monitoring solutions has grown accordingly.

Traditional methods for vehicle speed detection rely on physical sensing infrastructure such as radar, inductive loop detectors embedded in road surfaces, or LiDAR systems (Alwindi et al., 2024; Simbeye, 2022). While these approaches offer high measurement accuracy, they are costly to install and maintain, require significant physical infrastructure, and are inherently limited in spatial coverage. Their deployment at scale across road networks presents substantial logistical and economic challenges.

The growing accessibility of camera-based systems and advances in deep learning and computer vision have motivated research into vision-based alternatives for speed estimation. Cameras are comparatively inexpensive, widely deployed in existing traffic surveillance infrastructure, and capable of capturing rich spatial and temporal information about vehicle behaviour. This project investigates whether a purely software-based, calibration-free approach — one that requires no specialist hardware beyond a fixed roadside camera — can estimate vehicle speed with competitive accuracy. The system operates on monocular video footage and uses machine learning to learn the relationship between vehicle tracking behaviour and real-world speed directly from labelled data, without relying on explicit geometric modelling of the scene.

1.2 Hypothesis

A machine learning regressor trained on bounding box and trajectory features derived from a fixed camera can estimate vehicle speed with competitive accuracy without relying on explicit camera calibration

1.3 Aim

The project aims to design an algorithm capable of detecting, tracking, and estimating vehicle speed using visual data alone

1.4 Objectives

The objectives include:

- developing a dynamic object detection and tracking framework for vehicles;
- exploring deep learning approaches for improving accuracy over traditional computer vision methods;
- evaluating performance in terms of detection accuracy and computational efficiency.
- Evaluating a calibration-free approach to speed estimation that reduces reliance on geometric scene knowledge.

1.5 Deliverables

This project will produce the following deliverables:

- A Python-based vehicle detection, tracking and feature extraction pipeline, implementing YOLOv8n detection and DeepSORT tracking with an exit-anchored observation window
- Feature datasets in CSV format generated from the VS13 dataset, covering 400 video clips across 13 vehicle models
- A trained XGBoost regression model (v3) capable of estimating vehicle speed in km/h from bounding box tracking features, without camera calibration
- Evaluation results comparing cross-validation performance metrics on the training set and held-out evaluation on three unseen vehicle models
- A literature review examining camera calibration, learning-based speed estimation, object detection, and tracking methods
- This report, documenting the full research, design, implementation, and evaluation process

1.6 Structure of Report

The remainder of this report is organised as follows. Section 2 describes the research methodologies and methods adopted throughout the project. Section 3 presents the literature review, examining existing work in camera calibration, learning-based speed estimation, object detection, and tracking, and identifying the research gap that motivates this project's approach. Section 4 details the design and implementation of

the artefact, covering system architecture, dataset selection, the detection and tracking pipeline, feature extraction, and model training. Section 5 presents the testing and validation results, including cross-validation performance, held-out evaluation across unseen vehicles, feature importance analysis, and comparison against published benchmarks. Section 6 concludes the report by reviewing the aim, hypothesis, and objectives, summarising key findings, and identifying directions for future work.

2.0 RESEARCH METHODS

2.1 Research Methodologies

This research adopts a positivist research philosophy, as it is concerned with the objective measurement and evaluation of system performance using quantitative data. Vehicle speed estimation is treated as a measurable physical phenomenon, and the effectiveness of the proposed approach is assessed through observed experimentation rather than subjective interpretation. This stance is commonly adopted in computer vision and artificial intelligence research, where system performance can be evaluated using clearly defined metrics (Chatzichristos et al., 2025).

A deductive research approach is employed in this study. Existing theories and methods in vehicle speed estimation, object detection, and tracking are first examined through a review of available literature. Based on this foundation, a system is designed and evaluated to test whether a learning-based approach can estimate vehicle speed with acceptable accuracy. The outcomes of the experiments are then analysed to determine whether the observed results support the assumptions drawn from prior research.

In terms of research strategy, this project follows an experimental methodology. Controlled experiments are conducted using established datasets containing annotated vehicle speed information. By systematically training and evaluating models under defined conditions, the study aims to assess performance and identify factors that influence estimation accuracy. This strategy enables reproducibility and allows results to be compared with existing work in the field.

The research is conducted using a quantitative methodological choice, focusing on numerical performance indicators such as detection accuracy and speed estimation error. No qualitative data collection methods, such as surveys or interviews, are employed. The time horizon of the research is cross-sectional, as all experiments are

conducted over a fixed project duration using pre-existing datasets rather than longitudinal real-world deployments.

2.2 Research Methods

The research was conducted through a series of structured experimental stages. Initially, a comprehensive literature review was undertaken to identify existing approaches to vehicle speed estimation, highlighting calibration-based and learning-based methods. This informed the selection of appropriate techniques and datasets for the experimental phase.

The core method involved the use of offline experiments on publicly available traffic datasets containing ground-truth vehicle speed measurements. These datasets were used to train and evaluate a learning-based speed estimation system built upon object detection and tracking outputs. Using offline data allowed experiments to be conducted in a controlled and repeatable manner, facilitating detailed performance analysis.

Quantitative evaluation was performed by comparing estimated vehicle speeds against ground-truth annotations provided within the datasets. Performance was assessed using accuracy-based metrics, enabling objective comparison between experimental configurations. Detection performance was also evaluated, as inaccuracies at this stage directly affect subsequent tracking and speed estimation results.

All experiments were implemented using software-based tools, with models trained and evaluated using Python-based machine learning and computer vision libraries. Results were recorded and analysed to assess system performance and to identify limitations and potential areas for improvement.

This experimental methodology was chosen as it aligns with the aims of the project, allowing systematic evaluation of a learning-based approach to vehicle speed estimation while remaining feasible within the scope and constraints of an undergraduate research project.

3.0 LITERATURE REVIEW

3.1 Introduction

Vehicle speed estimation using visual data has been an active area of research within intelligent transportation systems and computer vision. The primary objective of such

systems is to estimate the real-world speed of vehicles using image or video data captured from roadside or surveillance cameras. Owing to the widespread availability and low cost of cameras, vision-based approaches have received considerable attention compared to sensor-based alternatives such as radar, LiDAR, or inductive loop detectors (Ansariyar, 2023; Cai et al., 2025; Simbeye, 2022).

Early vision-based methods predominantly relied on geometric modelling of the scene. These approaches typically involved explicit camera calibration to establish a mapping between image coordinates and real-world distances. By combining calibrated scene geometry with vehicle detection and tracking, speed could be computed by measuring the displacement of a vehicle over time. While effective in controlled environments, these methods often required manual intervention and were sensitive to camera placement and environmental changes (Fernández Llorca et al., 2021).

With advances in machine learning and deep learning, more recent approaches have incorporated data-driven techniques into the speed estimation pipeline. This pipeline involves camera calibration, object detection, and tracking (Macko et al., 2025). Certain implementations apply a learning-based approach to this pipeline. The following sections examine these approaches in greater detail, focusing on techniques, object detection methods, and tracking algorithms, to justify the project's methodology.

3.2 Camera calibration

This requires analysing the camera's extrinsic parameters such as its orientation, height, pitch angle, and relative position to the road plane, to accurately determine the corresponding real-world distance. This can be done manually using a fixed calibration or measuring the visible distance from the camera. Semi-automatic methods use automatic methods with at least one known metric such as parallel curves and vanishing points (Liao et al., 2025).

Vanishing point detection is the process of identifying the point(s) in an image where parallel lines in 3D space appear to converge due to perspective projection (Bharadwaj et al., 2024). It is a frequently used technique in computer vision for tasks like camera calibration.

Automatic methods don't require additional metric info. Some methods detect vanishing point and provide an estimated scale based on vehicle dimensions. Others use vehicles as landmarks and/or bounding boxes and work with their known shape.

For instance, Vuong, K., Tamburo, R. and Narasimhan, S.G., (2024) used Google Street View to make a 3D model of the scene, which was used to calibrate the camera.

Despite their effectiveness, calibration-based pipelines introduce some practical limitations. Their accuracy is highly sensitive to camera placement, scene geometry, and environmental conditions such as road curvature, occlusions, or changes in viewpoint. Manual and semi-automatic calibration methods often require prior knowledge of the scene or human intervention, reducing scalability and ease of deployment. Furthermore, calibration errors propagate directly into speed estimation, making these systems vulnerable to small inaccuracies in geometric modelling. These considerations have motivated recent research into learning-based approaches that aim to estimate vehicle speed directly from visual motion, reducing the reliance on explicit camera calibration.

3.3 Learning-Based Approaches

As discussed in Section 3.2, calibration-based pipelines introduce practical constraints that limit their scalability and robustness in real-world deployments. This has motivated a parallel body of research into learning-based approaches, which aim to estimate vehicle speed directly from visual data by exploiting patterns in image motion rather than through explicit geometric modelling. Rather than computing speed from calibrated scene coordinates, these approaches treat speed estimation as a regression or classification problem, learning the relationship between observed visual features and ground-truth speed from annotated training data (Fernández Llorca et al., 2021).

One of the earliest learning-based directions draws on classical optical flow estimation. Optical flow methods compute the apparent motion of pixels between consecutive frames, producing a dense or sparse field of velocity vectors that encodes how different regions of the image are moving. In the context of vehicle speed estimation, these motion fields can serve as inputs to statistical models or classifiers trained to associate flow magnitude with vehicle speed. Alwindi et al. (2024) demonstrate this approach using both a Lucas-Kanade optical flow system and a Gaussian Mixture Model variant, achieving mean percentage errors of 6.96% and 4.72%, respectively, at fixed reference speeds. While effective for relative speed ordering, optical flow methods face a fundamental limitation: translating pixel-level motion into absolute speed in km/h requires a known spatial scale, which itself depends on camera calibration (Wang & Wang, 2019). Optical flow, therefore, provides useful motion signal but does not, in isolation, resolve the calibration dependency.

More recent approaches apply deep learning architectures directly to the speed estimation task. Recurrent models (including Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), and Transformers) are well-suited to sequential video data, as they can capture temporal dependencies across frames without hand-engineered feature extraction. Mareddy et al. (2025) demonstrate this on the VS13 dataset, training LSTM, GRU, RNN, and Transformer architectures on sequences of bounding box coordinate differences extracted from tracked vehicles. Their best-performing model, an LSTM with an attention mechanism, achieves an RMSE of 3.96 km/h and a mean prediction accuracy of 94.25% on VS13, without explicit camera calibration. These results demonstrate that deep learning architectures can learn implicit speed-relevant structure from bounding box motion sequences, though they require substantially more model complexity and training data than classical approaches.

A third category involves extracting engineered features from tracked bounding box trajectories and training a conventional machine learning regressor on the resulting feature set. This approach occupies a middle ground between optical flow methods and end-to-end deep learning: it relies on human insight to identify informative features, such as apparent size change rate, and uses supervised learning to map these features to real-world speed. The approach is more interpretable than deep learning, less computationally demanding, and does not require dense pixel-level computation. Crucially, it can be implemented without any camera calibration if the model is trained on data from the target camera, as the geometric properties of the scene are implicitly encoded in the statistical relationship between bounding box features and labelled speed. It is within this category that the present project is situated.

It is important to note that many learning-based systems in the literature do not eliminate calibration entirely. They embed calibration geometry within a learned framework rather than removing the dependency. Macko et al. (2025), for example, use vanishing point detection and perspective transformation to rectify the scene before applying a modified object detector to estimate 3D bounding boxes, from which speed is derived. The learning component improves computational efficiency and detection quality, but the pipeline remains fundamentally calibration dependent. The degree to which calibration can be fully eliminated at the expense of accuracy remains an insufficiently explored question in the literature, and is the central approach evaluated in this project.

3.4 Object Detection

Following camera calibration, object detection forms a critical component of most vision-based vehicle speed estimation systems. Its primary role is to identify and localise vehicles within each video frame, typically through bounding boxes. Accurate detection is essential, as errors at this stage directly affect subsequent tracking and speed estimation accuracy.
Early approaches to vehicle detection relied on handcrafted features and classical machine learning techniques. However, these methods often struggled under varying lighting conditions, occlusions, and complex traffic scenes. The introduction of deep learning, convolutional neural networks (CNNs) especially, has led to significant improvements in detection robustness and generalisation (Adam et al., 2025).

Among modern object detectors, single-stage architectures have gained prominence due to their optimal balance between accuracy and computational efficiency. The YOLO (You Only Look Once) model is a notable example, widely adopted in traffic surveillance and intelligent transportation applications. YOLO-based detectors compute localised object detection and classification simultaneously, speeding up inference while maintaining competitive accuracy (Chen, 2025). Successive iterations of the YOLO architecture have introduced improvements in feature extraction, multi-scale detection, and network optimisation, further enhancing performance across diverse deployment scenarios (Ali & Zhang, 2024).

In the context of vehicle speed estimation, object detection is typically combined with calibrated scene geometry to associate pixel-level motion with real-world displacement. As discussed in Section 3.2, accurate calibration enables detected vehicle positions to be projected into a physical coordinate space. Consequently, detection accuracy and stability play a critical role in reducing error propagation throughout the speed estimation pipeline.
Recent studies have also explored optimising object detection models for efficiency, particularly in scenarios with computational constraints (Li et al., 2022; Macko et al., 2025).

Techniques such as model pruning, quantisation, and knowledge distillation have been investigated to reduce inference time and resource usage while preserving detection performance. These developments are especially relevant for real-time traffic analysis, although their applicability extends to offline processing when large-scale datasets are involved.

Overall, object detection serves as a foundational stage in vision-based speed estimation systems, providing the spatial information required for frame-to-frame analysis. The reliability of this stage strongly influences the effectiveness of tracking and speed estimation components, which are reviewed in the following section.

3.5 Tracking Methods

Tracking is a critical factor in vision-based vehicle speed estimation, as it enables the association of detected vehicles across consecutive frames. By maintaining consistent object identities over time, tracking allows the frame-to-frame displacement of vehicles to be measured, which is essential for estimating speed.

A widely adopted approach to object tracking in traffic surveillance is probability-based state estimation models, most notably the Kalman filter. The Kalman filter represents the state of a moving object using variables such as position and velocity and iteratively updates this estimate as new detections become available (Bai et al., 2023). By incorporating prediction and correction steps, the Kalman filter is effective at smoothing noisy measurements and handling temporary detection failures, making it well-suited to real-world traffic scenarios.

Building on this framework, Simple Online and Realtime Tracking (SORT) has become a popular tracking algorithm due to its simplicity and computational efficiency. SORT uses a Kalman filter to predict object positions in subsequent frames and applies the Hungarian algorithm to associate new detections with existing tracks. Its lightweight design allows it to operate at high frame rates, which has led to its widespread adoption in both offline analysis and real-time applications. However, SORT relies primarily on motion cues and may struggle in crowded scenes or under prolonged occlusion. (Wojke et al., 2017)

To address these limitations, extensions such as DeepSORT and StrongSORT incorporate appearance-based features learned through deep neural networks. By combining motion prediction with visual similarity, these methods improve identity preservation in complex environments. While such approaches increase robustness, they also introduce additional computational overhead, which may influence their suitability depending on application constraints.

More recent research has explored unified detection-and-tracking frameworks, in which both tasks are performed jointly within a single neural network. These end-to-

end approaches aim to reduce error propagation between detection and tracking stages and simplify the overall pipeline. However, their increased complexity and training requirements present additional challenges, particularly in data-limited or resource-constrained settings. (Meng et al., 2026)

In the context of vehicle speed estimation, the choice of tracking method directly affects the reliability of temporal motion measurements. As such, tracking accuracy and stability are key factors influencing overall system performance and are closely linked to the effectiveness of the detection stage discussed previously.

3.6 Summary and Identified Research Gap

This literature review has examined existing approaches to vehicle speed estimation using monocular camera systems, with particular focus on camera calibration techniques, object detection, and tracking methods. Traditional calibration-based approaches dominate much of the literature, relying on explicit geometric modelling of the scene through vanishing point detection, lane markings, or known object dimensions. While these methods can achieve high accuracy under controlled conditions, they typically require precise camera placement, manual intervention, or prior knowledge of scene geometry. As a result, their robustness and scalability in real-world deployments remain limited.

Recent advances in object detection, particularly the YOLO family of models, have significantly improved the feasibility of real-time traffic analysis using monocular video. These detectors offer a favourable balance between speed and accuracy, making them well-suited to downstream tasks such as tracking and speed estimation. However, detection alone is insufficient; reliable temporal tracking is essential to estimate vehicle motion over time. Lightweight tracking methods, such as Kalman filtering, are therefore commonly employed to maintain object identities and reduce the impact of noise and missed detections.

More recently, learning-based approaches have emerged as an alternative to explicit camera calibration. By exploiting temporal information and large annotated datasets, these methods aim to learn the relationship between image motion and real-world speed. Such approaches reduce reliance on manual calibration procedures and offer greater adaptability to diverse scenes. Nevertheless, many existing studies either continue to depend on some form of geometric calibration or prioritise real-time deployment on specialised hardware, leaving questions regarding accuracy-focused, software-only implementations insufficiently explored.

From the reviewed literature, a clear gap can be identified. There is a lack of work that systematically investigates a learning-based, calibration-free approach to vehicle speed estimation using monocular video, with primary emphasis on detection accuracy and feasibility within a purely software-based framework. Addressing this gap is particularly relevant for applications where ease of deployment and adaptability are prioritised over hardware-specific optimisation. This project seeks to contribute to this area by evaluating such an approach and assessing its effectiveness using established datasets.

4.0 DESIGN AND IMPLEMENTATION OF ARTEFACT

4.1 System Overview

The system designed in this project is a calibration-free, learning-based vehicle speed estimation pipeline operating on monocular video footage captured from a fixed roadside camera. The camera is positioned with a near-point-of-view (POV) perspective relative to oncoming vehicles: vehicles grow in apparent size as they approach, and exit the frame sharply off-axis after passing.

Figure 1 Sample frame showing camera POV

No camera calibration parameters, intrinsic or extrinsic, are used at any stage. Rather than computing speed through a geometric transformation from pixel coordinates to real-world

distance, which would require a known camera height, angle, and focal length, the system treats speed estimation as a supervised regression problem. A machine learning model learns the mapping between vehicle tracking behaviour and real-world speed directly from labelled data, implicitly absorbing the camera's geometric properties through the statistical structure of the training examples. This constitutes the primary design novelty of the system and directly addresses the gap identified in the literature review regarding calibration-free, software-only approaches to vision-based speed estimation (Fernández et al., 2021; Liao et al., 2025).

The pipeline comprises four sequential stages: vehicle detection, multi-object tracking, bounding box feature extraction, and speed regression. Each stage produces structured output consumed by the next, with the final stage outputting a speed estimate in km/h for each tracked vehicle. A schematic overview is shown in Figure 2.

Figure 2 Speed Estimation Pipeline

The system operates in an offline batch processing mode. Video clips are processed sequentially, with detection outputs, tracking trajectories, and extracted features saved as intermediate CSV files before model inference. This design reflects the project's priority of rigorous evaluation over real-time deployment, though the computational characteristics of the pipeline make real-time adaptation feasible, as discussed in Section 6.

4.2 Development Environment and Tools

The system was implemented entirely in Python, chosen for its extensive ecosystem of computer vision and machine learning libraries and its widespread adoption in academic research, both of which support reproducibility. The core libraries used are: Ultralytics for YOLOv8n detection, a Python port of DeepSORT for multi-object tracking, XGBoost for the primary regression model, and scikit-learn for the Random Forest baseline, cross-validation utilities, and hyperparameter search (Pedregosa et al., 2011; Chen & Guestrin, 2016). Pandas was used for feature CSV management and data manipulation, and OpenCV for video frame reading and preprocessing. Development was conducted on standard consumer hardware without GPU acceleration; all inference runs on CPU, keeping the system reproducible without specialist hardware.

4.3 Dataset

The VS13 dataset was selected as the primary data source (Djukanovic et al., 2022). The dataset consists of 400 short video clips drawn from 13 vehicle folders, each folder corresponding to a distinct vehicle model. Each clip is paired with an annotation file containing two values: the ground-truth speed in km/h and the pass-by timestamp at which the vehicle passes the camera. Vehicle speeds range from approximately 30 to 105 km/h, covering urban and semi-rural driving conditions.

VS13 was preferred over alternatives such as BrnoCompSpeed and KITTI for reasons of compatibility with the calibration-free design. BrnoCompSpeed is built around camera calibration workflows and is primarily used to benchmark calibration-dependent pipelines (Macko et al., 2025). KITTI is designed for autonomous vehicle perception and does not provide the per-clip speed ground truth required here (Liao et al., 2023). VS13's per-clip annotations make it directly compatible with the feature extraction and regression approach adopted in this project.

The dataset was loaded programmatically by iterating over all vehicle folders and pairing each .mp4 clip with its corresponding .txt annotation file by filename stem. Clips for which a matching annotation could not be found were skipped. Prior to detection, video frames are downscaled from their native 1080p resolution to 720p using OpenCV frame resizing, applied as a preprocessing step before each frame is passed to the detector.

frame = cv2.resize(frame, (1280, 720))

This resolution reduction meaningfully decreased inference time with no observed degradation in detection quality, as 720p retains sufficient spatial resolution for vehicle-scale bounding box estimation. The pass-by timestamp present in each annotation file is not used by the final system during inference. The design decision to eliminate this dependency is discussed in Section 4.7.

4.4 Vehicle Detection

YOLOv8n (the nano variant of the YOLOv8 architecture) was selected for vehicle detection (Ali & Zhang, 2024). The model is applied in inference-only mode using pretrained COCO weights, with no fine-tuning on VS13 data. This reflects the role of detection as a fixed pipeline component: the goal is to obtain stable, consistent bounding box detections as input to the tracking stage, not to optimise the detector itself.

The nano variant was chosen over larger YOLO configurations on the basis that speed estimation accuracy in this pipeline is more sensitive to temporal tracking consistency than to single-frame detection precision. This is consistent with findings in the literature, where

smaller detection models have been shown to produce equivalent or better downstream speed estimation compared to higher-precision alternatives (Macko et al., 2025). Detection confidence is thresholded at 0.4, with an IoU threshold of 0.5 for non-maximum suppression. Only vehicle-relevant COCO classes are retained: car (class 2), motorbike (class 3), bus (class 5), and truck (class 7).

The detector was applied frame-by-frame to each video clip, with output filtered to the above classes and a confidence threshold before being passed to the tracker. Initial testing was conducted on individual clips to verify stable, consistent detections before proceeding to full-dataset processing. Inference times averaged approximately 25–36ms per frame on CPU at 720p, confirming sufficient throughput for offline batch processing across the full dataset.

Figure 3 YOLOv8n Bounding Box Output on a VS13 clip

4.5 Vehicle Tracking

DeepSORT was adopted for multi-object tracking (Wojke et al., 2017). DeepSORT combines Kalman filter-based motion prediction with appearance features derived from a deep neural network, allowing it to maintain consistent vehicle identities across frames even under brief detection failures. The Kalman filter component additionally smooths bounding box trajectories between frames, reducing the impact of detection jitter on subsequently extracted features.

The selection of DeepSORT followed the practical rejection of SORT (Bewley et al., 2016). SORT uses a Kalman filter and the Hungarian algorithm for detection association, but relies solely on positional proximity, with no appearance matching. In initial testing on VS13 clips, SORT produced significant identity fragmentation — the same physical vehicle being assigned

multiple distinct track IDs across a clip due to brief detection gaps or low-confidence frames. Since feature extraction aggregates statistics over a single continuous trajectory, this fragmentation directly corrupts the extracted features, producing measurements that do not represent a coherent vehicle pass-by event. DeepSORT's appearance-based re-identification resolved this in practice, producing single continuous tracks for the majority of clips. StrongSORT and other enhanced variants were not adopted, as the additional complexity is not warranted given the single-vehicle, fixed-camera nature of the VS13 clips.

Figure 4.1 DeepSORT tracking at approach start Figure 4.2 Maintained track ID across frame

The tracker was configured with a maximum track age of 30 frames, a minimum initialisation count of 3 frames before a track is confirmed, and a maximum cosine distance of 0.2 for appearance-based matching. These parameters balance track persistence against the risk of incorrectly merging distinct detections in the rare cases where multiple vehicles appear simultaneously. The longest track within each clip was designated as the main vehicle — a heuristic that reliably identifies the primary pass-by event in VS13, where each clip is designed around a single vehicle.

4.6 Initial Speed Estimation Approach and Pivot

Before committing to the feature extraction and regression approach, direct pixel speed calculation was implemented as an exploratory step. Vehicle centroid displacement was computed between consecutive frames using Euclidean distance and multiplied by the frame rate to obtain a pixel-per-second velocity estimate. This produced plausible relative orderings (faster vehicles generated higher pixel velocities) but could not be translated into km/h without a pixel-to-metre conversion factor, which in turn requires camera calibration

parameters unavailable in this setting. The approach confirmed that bounding box motion carries speed-correlated signal, while also confirming that extracting absolute speed from pixel displacement without calibration is not tractable. This finding directly motivated the learning-based regression approach: rather than computing speed from first principles, a model would be trained to learn the implicit relationship between bounding box behaviour and real-world speed from labelled examples.

4.7 Feature Extraction

4.7.1 Observation Window and Anchor Design

A central design decision concerns how the observation window is positioned in time relative to the vehicle's pass-by event. Three elements define the window: the anchor point, the window duration, and whether a post-anchor component is included.

The initial implementation anchored the window to the annotated pass-by timestamp from the ground-truth file, extracting 2.5 seconds before and 0.5 seconds after this point. This produced a competitive training baseline: CV MAE of 3.86 km/h and R^2 of 0.940. This, however, introduced a fundamental deployment problem: the pass-by timestamp exists only in the dataset annotations and would not be available during real-world inference. A system trained using ground-truth timing but forced to approximate that timing at deployment creates a systematic train-inference mismatch that undermines practical validity. Early inference experiments (without relying on annotated pass-by time) confirmed this: approximating the pass-by timestamp using peak bounding box area produced timing drift of up to 26 frames, with correspondingly large prediction errors.

The final system resolves this by anchoring the observation window to the last confirmed frame of the main vehicle's track (the exit frame). For the VS13 camera geometry, where vehicles approach on-axis and exit sharply off-axis after passing, the bounding box exit event is geometrically guaranteed and can be identified directly from tracking output without any annotation. This anchor is consistent between training and inference, eliminates the annotation dependency, and makes the system genuinely feasible for deployment. Features are extracted from a window ending at the exit frame, with no post-window component; under exit anchoring, peak bounding box size falls naturally at the tail of the pre-window, making the post-window redundant.

Window duration was evaluated empirically across four configurations (1.0s, 1.5s, 2.0s, and 2.5s) using five-fold cross-validation on the full dataset. The 1.0s window produced the lowest RMSE and the most stable variance across folds and was selected for all subsequent modelling. Longer windows did not improve accuracy and introduced greater fold-to-fold variance, suggesting that additional early-approach frames added noise rather than signal

(See section 5.2). The extraction was implemented in feature_extraction_exit_anchor_v2.py, which processes each clip in a single pass, identifies the main track by longest duration, records the exit frame, and extracts bounding box features from the 1.0s window ending at that frame. (see Appendix 8.5)

4.7.2 Feature Set

Features are computed from the bounding box trajectory of the main track within the 1.0s extraction window and organised into three categories. The first covers apparent width dynamics — how the vehicle's bounding box width changes across the window. This is the most informative category for the near-POV geometry: a vehicle approaching straight-on grows primarily in width, with width growth rate encoding approach speed. The second covers height dynamics and aspect ratio changes, providing complementary perspective information. The third covers centroid displacement and pixel velocity statistics, which carry weaker signal in near-POV geometry because the vehicle moves primarily toward the camera, making lateral and vertical centroid displacement minimal.

The initial feature set comprised 17 features. An expanded set was developed in the third implementation version, adding: width_expansion_rate (absolute pixel width growth per second), std_width_delta (standard deviation of frame-by-frame width changes), max_width_delta (peak single-frame width expansion), TTC (time-to-contact estimate: mean_width / width_expansion_rate), aspect_ratio_change_rate, and std_height_delta. The YOLO-predicted vehicle class was also included as a candidate feature to provide explicit vehicle type information, with the expectation it might assist size-dependent generalisation. It was subsequently pruned after contributing zero importance across all cross-validation folds, suggesting the model absorbs vehicle type information implicitly through size-related features. (See Appendix 8.1)

The most consequential feature engineering change was the introduction of width_growth_norm, computed during training rather than extraction. The original width_growth feature — defined as (final_width − initial_width) / initial_width — is size-confounded: smaller vehicles with smaller initial widths produce disproportionately large ratios for identical absolute bounding box growth, causing the model to conflate high growth ratios with high speed regardless of vehicle size. Normalising by mean_width instead decouples the ratio from absolute starting size, anchoring it to the vehicle's typical size across the full observation window:

$$\text{df["width\_growth\_norm"]} = \text{df["width\_growth"]} / \text{df["mean\_width"]}$$

This normalisation resolved the model's generalisation failure on small vehicles — the root cause of a 32 km/h inference error on the Mazda3 in early held-out testing (See Appendix

8.3)— and became the dominant feature in the final model, accounting for approximately 70% of model decisions (See Section 5).

Four features were pruned after consistently contributing zero importance across cross-validation folds: track_length, height_expansion_rate, width_change_rate, and vehicle_class. All other features were retained; earlier experiments showed that pruning low-importance but non-zero features consistently degraded performance, so only features with confirmed zero importance were removed. The final training set comprises 23 features. A full summary of all features is given in Appendix 8.1.

4.8 Speed Regression Model and Training Pipeline

4.8.1 Model Selection

Speed estimation is formulated as a supervised regression task. A Random Forest regressor is trained as the baseline using scikit-learn defaults, providing a benchmark against which the primary model is compared (Breiman, 2001). XGBoost is adopted as the primary model (Chen & Guestrin, 2016). XGBoost's gradient boosting framework is well-suited to structured tabular regression with a moderate number of features and samples, and its regularisation mechanisms make it more robust to overfitting than Random Forest at this dataset scale.

A neural regression model was considered and rejected. The feature set is compact and structured, and 309 training samples make deep learning approaches likely to overfit without substantial regularisation effort. Tree-based ensemble methods offer comparable performance on tabular data of this scale, with the additional benefit of interpretable feature importance output (Grinsztajn et al., 2022). Interpretability was a practical consideration: both Random Forest and XGBoost expose per-feature importance scores, enabling post-hoc validation that the model is learning physically meaningful signals rather than spurious correlates. The dominance of width_growth_norm and mean_width in the final model's importance distribution is defensible given the camera geometry and is discussed further in Section 5.

4.8.2 Dataset Partitioning

The dataset was partitioned at the vehicle folder level rather than the sample level. Row-level random splitting would allow clips from the same vehicle model to appear in both training and test sets, introducing data leakage through shared bounding box geometry and artificially inflating held-out performance. Folder-level splitting ensures held-out vehicles are entirely unseen during training, providing a genuine test of generalisation.

Ten vehicle folders were assigned to training (309 samples) and three to a held-out evaluation set (91 samples): Mazda3 (small, mean bounding box width ~342px), NissanQashqai (mid, ~406px), and MercedesAMG550 (large, ~447px). These three were selected to represent a

range of apparent sizes, stress-testing the model across the size distribution. An initial configuration included Peugeot307 as the small-vehicle representative, but its mean bounding box width (336px) fell below the minimum in the training set, producing out-of-distribution extrapolation rather than generalisation. Peugeot307 was moved to training and replaced by Mazda3, which is small but within the training distribution boundary.

4.8.3 Cross-Validation and Hyperparameter Tuning

All model selection and hyperparameter decisions were made using five-fold cross-validation on the 309-sample training set. Single-split evaluation was found to be unreliable during development: an early experiment showed the 2.0s window outperforming 1.0s on a single random split (MAE 2.98 km/h, R^2 0.964), but cross-validation reversed this finding, correctly identifying 1.0s as the more stable configuration. All reported training performance metrics are CV-averaged unless otherwise stated.

Hyperparameter tuning was performed using RandomizedSearchCV with 60 iterations and five-fold cross-validation, optimising for RMSE. The tuned configuration is shown in Appendix 8.2. The tuned model uses shallower trees (max_depth: 4), high minimum child weight (min_child_weight: 10), and lower learning rate than the defaults — a pattern consistent with regularisation appropriate for a 309-sample dataset where the default configuration was over-complex. The tuned XGBoost model was retrained on the full 309-sample training set following hyperparameter selection, saved as xgb_1s_speed_model_final_v3.pkl, and used for all held-out evaluation and inference. The training pipeline is implemented in train_final_with_heldout_v2.py; single-video inference is handled by inference_final.py.

4.9 Model Development Summary

The implementation progressed through three substantive versions, each addressing a specific limitation identified in the previous. Appendix 8.3 summarises performance across versions.

Version 1 used a pass-by anchor with a 2.5s+0.5s window and default XGBoost, achieving a competitive training baseline (CV MAE 3.86 km/h, R^2 0.940) but remaining deployment-infeasible due to the annotation timing dependency. Version 2 introduced the exit anchor and 1.0s window, eliminating the annotation dependency and improving CV MAE to 3.16 km/h, but held-out evaluation revealed a catastrophic generalisation failure on the Mazda3 (inference error exceeding 30 km/h) caused by the size-confounded width_growth feature. Version 3 resolved this through the introduction of width_growth_norm, an expanded feature set, and hyperparameter tuning, achieving a CV MAE of 2.44 km/h and R^2 of 0.971 on the training set, and a held-out MAE of 3.66 km/h and R^2 of 0.949 on unseen vehicles. The Mazda3 MAE dropped from 5.09 km/h in v2 to 3.14 km/h in v3. These results form the basis for the evaluation presented in Section 5.

5.0 TESTING AND VALIDATION OF ARTEFACT

5.1 Evaluation Framework

Three metrics are used throughout this evaluation: mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2). MAE reports the average magnitude of prediction errors in km/h and is the primary reported metric, as it is directly interpretable in the context of speed enforcement tolerances. RMSE penalises larger errors more heavily than MAE due to the squaring of residuals, making it useful for diagnosing whether errors are evenly distributed or driven by a small number of large outliers. R^2 measures the proportion of variance in ground-truth speed explained by the model's predictions, providing a scale-independent view of fit quality. Together, these three metrics give a complete picture of accuracy, error distribution, and explanatory power.

All model selection and hyperparameter decisions were made using five-fold cross-validation (CV) on the 309-sample training set, rather than a single train/validation split. Single-split evaluation was found to be unreliable during development: an early experiment showed the 2.0s observation window outperforming the 1.0s window on a single random split (MAE 2.98 km/h, R^2 0.964), but cross-validation reversed this finding, correctly identifying 1.0s as the more stable and generalisable configuration. All training-phase performance figures reported in this section are five-fold CV averages unless explicitly stated otherwise.

The dataset was partitioned at the vehicle folder level, as described in Section 4.8.2. The held-out set of three unseen vehicle models — Mazda3, NissanQashqai, and MercedesAMG550 — was reserved entirely for final evaluation and was not used at any stage of model selection or tuning. This partition provides the most rigorous available test of the system's ability to generalise to vehicle sizes and bounding box geometries not seen during training.

5.2 Observation Window Comparison

Four observation window durations were evaluated under the exit anchor scheme — 1.0s, 1.5s, 2.0s, and 2.5s — using five-fold cross-validation on the full dataset. Results are shown in the table below.

Window (s)	CV MAE	CV RMSE	CV R^2	RMSE ±	R^2 ±
1.0	3.16	4.51	0.947	1.04	0.026

1.5	3.14	5.00	0.928	1.99	0.063
2.0	3.23	5.15	0.925	1.95	0.065
2.5	3.24	5.20	0.928	1.40	0.044

Table 1 Five fold cross-validation across various observation windows

The 1.0s window produced the lowest RMSE and the most stable fold-to-fold variance of all configurations tested. Longer windows did not improve accuracy and introduced substantially greater variance, as indicated by the RMSE standard deviation increasing from ±1.04 at 1.0s to ±1.95 at 2.0s. This suggests that frames extracted earlier in the vehicle's approach — more than one second before exit — add noise rather than useful signal. Under the exit anchor, the vehicle is at or near its largest apparent size at the tail of the 1.0s window, meaning the most geometrically informative frames are fully captured without the instability introduced by earlier, smaller-bounding-box frames.

It is worth noting that shorter windows performed worse under the earlier pass-by anchoring scheme. This is not a contradiction: under pass-by anchoring, the peak bounding box frames fell in the post-window segment, so pre-only windows missed the most informative data. The exit anchor resolves this by construction — peak size is structurally guaranteed to fall within the pre-window — and this is precisely why shorter windows become viable and indeed optimal under the revised design.

5.3 Model Comparison

A Random Forest regressor trained with scikit-learn defaults on the 23-feature set was used as the baseline. XGBoost was trained with default hyperparameters as the initial primary model, followed by the tuned configuration. The version history table in appendix 8.3 summarises performance across all stages of development.

The default XGBoost model outperformed the Random Forest baseline on CV metrics, consistent with the expectation that gradient boosting's sequential error correction provides better generalisation than Random Forest's parallel ensemble approach on structured tabular data at this dataset scale (Grinsztajn et al., 2022). The improvement is particularly visible in RMSE, where XGBoost's regularisation more effectively suppresses large individual errors. Feature importance interpretability was comparable between the two models, with both identifying width-related features as dominant, validating that this pattern is a genuine property of the data rather than an artefact of a particular algorithm's inductive bias.

5.4 Hyperparameter Tuning Results

Hyperparameter tuning was performed using RandomizedSearchCV with 60 iterations and five-fold cross-validation, optimising for RMSE. The tuned configuration is shown in the table below.

Parameter	Value
n_estimators	441
Learning_rate	0.0931
Max_depth	4
Min_child_weight	10
Subsample	0.7213
Colsample_bytree	0.8171
Reg_alpha	0.2503
Reg_lambda	0.6748

Tuning produced consistent improvements across all metrics. CV MAE improved from 2.86 ± 0.17 km/h (pruned baseline, pre-tuning) to 2.44 ± 0.26 km/h, and CV R^2 improved from 0.963 to 0.971. The tuned configuration is notably more regularised than the defaults: max_depth was reduced from 6 to 4, and min_child_weight was raised to 10, a high value that prevents the model from creating leaf nodes based on small clusters of training samples. Both changes reflect a dataset of 309 samples for which the default model was over-complex. The lower learning rate (0.0931 vs 0.1 default) combined with more estimators (441 vs 100 default) is a standard regularisation trade-off — slower learning with more iterations — that further reduces overfitting. Taken together, the tuned hyperparameters consistently indicate a model that needed to be constrained to the data's scale rather than expanded in capacity.

5.5 Evaluation

The tuned XGBoost v3 model was evaluated on 91 samples from the three held-out vehicles. Overall results are shown in the table below.

Overall Held out	

MAE	3.66 km/h
RMSE	4.88 km/h
R^2	0.949

Per-Vehicle Breakdown

Per-vehicle results are shown in the table below.

Vehicle	MAE	RMSE	R^2	Bias	Samples
Mazda3	3.14	5.18	0.948	-0.69	32
NissanQashqai	3.25	3.94	0.960	-1.68	29
MercedesAMG550	4.62	5.33	0.940	-3.47	30

All three held-out vehicles achieved R^2 above 0.94, indicating that the model successfully captures the variance structure of speed across unseen vehicle geometries. The NissanQashqai — a mid-sized vehicle well-centred within the training distribution — produced the lowest RMSE (3.94 km/h) and highest R^2 (0.960). The Mazda3, despite being the smallest held-out vehicle, achieved competitive MAE (3.14 km/h), representing a substantial improvement over v2 performance (5.09 km/h) and confirming that width_growth_norm resolved the small-vehicle generalisation failure identified during development. The MercedesAMG550 produced the highest MAE (4.62 km/h) and the largest systematic bias (-3.47 km/h), discussed further in Section 5.9.

Per-Speed-Bin Breakdown

Per-speed-bin results are shown in the table below.

Bin	MAE	Samples
30-50 km/h	2.99	22
50-70 km/h	3.47	25
70-90 km/h	3.45	24
90-110 km/h	4.91	20

MAE is relatively stable across the 30–90 km/h range (2.99–3.47 km/h) before degrading at the 90–110 km/h bin (4.91 km/h). Two factors contribute to this degradation. First, the training set contains fewer examples at the upper speed range, reducing the model's exposure to the bounding box dynamics of very fast vehicles. Second, the relationship between bounding box width growth and speed compresses at high speeds: the difference in apparent size change between a vehicle travelling at 90 km/h and one at 105 km/h is proportionally smaller than the equivalent difference at lower speeds, making fine-grained discrimination harder. This effect is inherent to the near-POV camera geometry and is not readily addressed through feature engineering alone.

Residual Summary

The residual distribution over the held-out set is summarised in the table below.

Metric	Value
Mean bias	−1.92 km/h
Std of residuals	4.48 km/h
Max overestimate	+15.54 km/h
Max underestimate	−15.66 km/h
Errors ≤ 5 km/h	75.8%
Errors ≤ 10 km/h	93.4%

The mean bias of -1.92 km/h indicates a systematic tendency to underestimate speed, consistent across all three held-out vehicles (See Appendix 8.6). The 93.4% of predictions within 10 km/h of ground truth and 75.8% within 5 km/h provide a practical characterisation of deployment-level reliability. The maximum errors of ±15.5 km/h occur at the extremes of the speed distribution, consistent with the high-speed degradation pattern identified above.

5.6 Feature Importance Analysis

CV-averaged feature importance for the tuned XGBoost v3 model is shown in Table X. **[TABLE: Feature | Importance — from v3 handoff. Top entries: width_growth_norm: 0.698, mean_width: 0.134, width_growth: 0.052, std_euclidean: 0.033, remaining features collectively ~0.083.]**

width_growth_norm accounts for approximately 70% of model decisions, with mean_width contributing a further 13%. Together, width-related features account for over 88% of

importance. This distribution is physically defensible for the near-POV camera geometry of VS13: a vehicle approaching roughly straight-on grows primarily in apparent width, and the rate of that growth — normalised to the vehicle's own typical size — is the strongest available visual signal of approach speed. The model is not taking an easy correlate; it is learning the correct physical signal for this geometry.

Centroid displacement features (mean_dx, mean_dy and related statistics) collectively contribute less than 2% of importance. This is consistent with the near-POV geometry: because the vehicle moves primarily toward the camera rather than across the frame, lateral and vertical centroid displacement is inherently low-variance and carries little information about speed. The model correctly underweights these features, providing a post-hoc validation that the importance distribution reflects genuine physical structure rather than spurious correlates.

The pruned features — track_length, height_expansion_rate, width_change_rate, and vehicle_class — each contributed zero importance consistently across all five folds, confirming that their removal was correct. The explicit inclusion of vehicle_class as a candidate feature is worth noting it was added with the expectation that explicit vehicle type information might assist size-dependent generalisation, but the model absorbed this information implicitly through size-related bounding box features, rendering the explicit label redundant.

5.7 Inference Examples

To illustrate system behaviour on individual clips, three held-out vehicle examples are presented. These examples were generated using inference_final.py on clips from the held-out set.

Table X shows inference results for one representative clip per vehicle. [TABLE: Vehicle | Ground Truth | Predicted | Error — NissanQashqai: 68.0 km/h, 64.53 km/h, 3.47 km/h | MercedesAMG550: 100.0 km/h, 91.42 km/h, 8.58 km/h | Mazda3 (v2): 67.0 km/h, 99.05 km/h, 32.05 km/h | Mazda3 (v3): within 3.14 km/h MAE.]

The NissanQashqai prediction (error 3.47 km/h) is representative of typical system performance on mid-sized vehicles within the training size distribution. The MercedesAMG550 prediction (error 8.58 km/h) is above the overall MAE and reflects the systematic underestimation bias that scales with vehicle size, discussed in Section 5.9. The Mazda3 result is presented across two versions to illustrate the effect of width_growth_norm: under v2, the size-confounded width_growth feature caused the model to catastrophically overestimate speed (predicted 99.05 km/h against a ground truth of 67.0 km/h, error 32.05 km/h). Under v3, with width_growth_norm as the primary feature, Mazda3 MAE dropped to

3.14 km/h, confirming that the normalisation fix resolved the failure mechanistically rather than through overfitting.

5.8 Comparison Against Published Benchmarks

Table X summarises the comparison of the proposed system against three published works.

System	Dataset	Error Metric	Calibration Required	Approach
Macko et al. (2025)	BrnoCompSpeed	Mean 0.71 km/h, Median 0.55 km/h	Yes — vanishing point detection + perspective transform	3D bounding box reconstruction, YOLOv6, edge hardware
Mareddy et al. (2025) — LSTM	VS13	RMSE 3.96 km/h, Accuracy 94.25%	No	Deep learning (LSTM) on sequential bounding box features
Alwindi et al. (2024) — GMM	Custom (fixed speeds)	Mean error 4.72%	Yes	Gaussian Mixture Model on optical flow
This work	**VS13**	**HO MAE 3.66 km/h, RMSE 4.88 km/h, R^2 0.949**	**No**	**XGBoost on bounding box tracking features, standard CPU**

Macko et al. (2025) represent the current state of the art in calibration-dependent vision-based speed estimation, achieving a mean error of 0.71 km/h and median error of 0.55 km/h on the BrnoCompSpeed dataset. Their pipeline requires explicit camera calibration via vanishing point detection, perspective image transformation, and 3D bounding box reconstruction using a modified YOLOv6 architecture — a substantially more complex system evaluated on specialised edge hardware. Direct numerical comparison with the present system is not straightforward as the two systems are evaluated on different datasets, but their results establish the accuracy ceiling achievable when full geometric information is available. The present system sacrifices accuracy in exchange for the complete elimination of calibration requirements, operating on standard CPU hardware without any scene geometry information.

Mareddy et al. (2025) provide the most directly comparable benchmark, as their system is evaluated on the VS13 dataset under a calibration-free design. Their best-performing model,

an LSTM with attention mechanism, achieves an RMSE of 3.96 km/h on VS13, alongside a mean per-prediction accuracy of 94.25% using the metric Accuracy (%) = (1 − |predicted − actual| / |actual|) × 100 (Mareddy et al., 2025). The present system achieves a held-out RMSE of 4.88 km/h on VS13, which is slightly higher. However, several contextual differences are relevant. Mareddy et al. employ deep learning architectures — LSTM, GRU, RNN, and Transformer — trained end-to-end on sequential bounding box features, representing considerably greater model complexity and computational demand. The present system uses a compact feature set fed to a gradient-boosted tree regressor, which is more interpretable and requires substantially less training data and compute. Additionally, the train/test partitioning strategy used by Mareddy et al. is not explicitly described; if clips from the same vehicle model appear in both training and test sets, their evaluation may benefit from shared bounding box geometry, whereas the present system's folder-level split ensures held-out vehicles are entirely unseen. Within these caveats, the two systems are broadly competitive on the same dataset under calibration-free conditions.

Alwindi et al. (2024) evaluate a Lucas-Kanade optical flow system and a Gaussian Mixture Model (GMM) approach at fixed reference speeds of 20, 40, 60, and 80 km/h. Their GMM system achieves an overall mean percentage error of 4.72%, and the Lucas-Kanade system 6.96%. Both approaches require camera calibration, which the present system explicitly avoids. Direct numerical comparison is limited by the different evaluation conditions — fixed reference speeds on a separate dataset rather than the variable-speed VS13 clips — but the percentage errors at their mid-range reference speeds (60–80 km/h) correspond roughly to 3–4 km/h absolute error, placing their GMM system at broadly similar accuracy to the present work while requiring calibration infrastructure the present system does not.

Taken together, the evaluation positions the proposed system as competitive with calibration-free published work on the same dataset, and demonstrates that bounding box feature regression can approach the accuracy of more complex deep learning pipelines without camera calibration, specialist hardware, or end-to-end sequence modelling.

5.9 Limitations

Systematic underestimation bias: The model exhibits a consistent negative bias that scales with vehicle size: Mazda3 −0.69 km/h, NissanQashqai −1.68 km/h, MercedesAMG550 −3.47 km/h. This is a direct consequence of the width_growth_norm normalisation. Dividing by mean_width means that a larger vehicle — which has a larger denominator — produces a proportionally smaller normalised growth ratio for the same absolute expansion. The model, trained primarily on mid-sized vehicles, interprets smaller normalised ratios as lower speeds. This trade-off was accepted because it resolved the catastrophic Mazda3 failure of v2 (32.05 km/h error) at the cost of a modest and mechanistically understood bias on larger vehicles.

The bias is predictable in direction and scales monotonically with vehicle size, meaning a calibration layer could in principle correct it given knowledge of vehicle class.

High-speed degradation: MAE at 90–110 km/h (4.91 km/h) is substantially higher than at 30–70 km/h (2.99–3.47 km/h). Two causes are identified: lower training sample density at the upper speed range, and the compression of the bounding box expansion signal at high speeds — the absolute difference in width growth rate between 90 km/h and 105 km/h vehicles is smaller than the equivalent difference at lower speeds. This is a fundamental geometric constraint of the near-POV camera setup.

Single camera geometry: The model implicitly learns the geometric properties of the VS13 camera through the statistical structure of the training data. No claim of generalisation to other camera positions, heights, or angles is made or tested. Deploying the system on a different camera would require retraining on labelled data from that specific setup.

Dataset scale: The training set comprises 309 samples across 10 vehicle types. Performance at the edges of the bounding box size distribution remains fragile, as demonstrated by the initial Peugeot307 failure (MAE 7.23 km/h) when a vehicle's mean width fell outside the training distribution. The current system is robust within the distribution boundaries established by the training set but should not be expected to extrapolate beyond them.

6.0 CONCLUSION

6.1 Aim

The aim of this project was to design an algorithm capable of detecting, tracking, and estimating vehicle speed using visual data alone. This aim has been met. The final system is a four-stage pipeline comprising YOLOv8n detection, DeepSORT tracking, bounding box feature extraction, and XGBoost regression. It produces speed estimates in km/h from monocular video footage without any camera calibration, geometric scene knowledge, or specialist hardware. On a held-out evaluation set of three unseen vehicle models totalling 91 samples, the final system achieves a MAE of 3.66 km/h, RMSE of 4.88 km/h, and R^2 of 0.949, with 93.4% of predictions falling within 10 km/h of ground truth.

6.2 Hypothesis

The hypothesis investigated is supported by the experimental results. The system achieves accuracy broadly competitive with calibration-free published work on the same dataset: Mareddy et al. (2025) report an LSTM RMSE of 3.96 km/h on VS13

using a substantially more complex deep learning architecture, while the present system achieves 4.88 km/h RMSE using a compact, interpretable feature set and a gradient-boosted tree regressor. The hypothesis is supported with the caveat that accuracy is constrained by the single-camera geometry and dataset scale, and that calibration-dependent systems, such as Macko et al. (2025) which operate with full geometric information, remain considerably more accurate at 0.71 km/h mean error.

6.3 Summary of Key Findings

Four findings from this project are worth drawing out explicitly.

The exit anchor design eliminates the annotation dependency that made the initial implementation not feasible for deployment. By anchoring the observation window to the last confirmed frame of the vehicle's track, the system requires no ground-truth timing information at inference. This architectural decision was validated empirically: the exit-anchored system matched and ultimately exceeded the accuracy of the pass-by-anchored baseline while being practically deployable.

The width_growth_norm normalisation was the most consequential feature engineering decision in the project. The original width_growth feature conflated vehicle size with speed, causing the model to catastrophically overestimate speed for the Mazda3 (32.05 km/h error in v2). Normalising by mean_width rather than initial_width separated growth rate from absolute starting size, reducing the Mazda3 MAE from 5.09 km/h to 3.14 km/h and confirming that the failure was mechanical rather than fundamental to the approach.

Feature importance analysis validates that the model is learning a physically meaningful signal. The dominance of width_growth_norm (approximately 70% of model decisions) is not a result of unfounded correlation, but rather it reflects the primary visual characteristic of a vehicle approaching a near-POV fixed camera, where apparent width growth rate is the strongest available proxy for approach speed. The underweighting of centroid displacement features is also consistent with the camera geometry: lateral and vertical centroid movement is minimal when a vehicle moves directly toward the camera. The importance distribution confirms that the system is operating on the correct physical signal.

Calibration-free speed estimation is feasible at competitive accuracy using a simple, interpretable pipeline. The system operates entirely on CPU at approximately 25–36ms per frame for the detection stage, requiring no GPU, no calibration equipment, and no scene geometry annotation. This positions it as a genuinely deployable option

for traffic monitoring contexts where calibration infrastructure is unavailable or impractical.

6.4 Limitations

The principal limitations of the system are documented in detail in Section 5.9 and are summarised briefly here. A systematic underestimation bias scales with vehicle size, reaching −3.47 km/h for the MercedesAMG550, as a direct consequence of the width_growth_norm normalisation. Accuracy degrades at high speeds (MAE 4.91 km/h at 90–110 km/h) due to lower training sample density and signal compression in the bounding box expansion relationship at high approach speeds. The model implicitly encodes the geometry of a single fixed camera and has not been tested on other camera positions or angles. The training set of 309 samples across 10 vehicle types limits robustness at the edges of the bounding box size distribution.

6.5 Future Work

Several directions follow naturally from the findings of this project. The systematic underestimation bias that scales with vehicle size is mechanically understood and could be addressed through a post-prediction bias correction layer dependent on bounding box width, or through augmenting the training set with more large-vehicle examples. High-speed degradation could be partially addressed by increasing training data density at higher speeds if the dataset permits.

The inference pipeline already operates at approximately 25–36ms per frame for the detection stage on standard CPU hardware, which approaches real-time throughput. Adapting the system for live video inference would require implementing a rolling window that updates continuously as new frames arrive and outputs a prediction at the exit event. This is a straightforward engineering step given the current architecture.

Finally, cross-camera generalisation remains untested. The current design has not been tested for generalisation, which limits deployment scope. A natural extension would be to evaluate whether a model trained on one VS13-style camera setup transfers to another, or whether a small amount of labelled data from a new camera is sufficient to fine-tune the existing model.

7.0 REFERENCE LIST

Adam, M.A.A. & Tapamo, J.R. (2025) 'Survey on image based vehicle detection methods', World Electric Vehicle Journal, 16(6), 303. Available at: https://www.mdpi.com/2032-6653/16/6/303

Ali, M.L. & Zhang, Z. (2024) 'The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection', Computers, 13(12), 336. doi: 10.3390/computers13120336

Alwindi, M., Tombokti, T. & Ganoun, A. (2024) 'Real-Time Vehicle Speed Estimation', 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Tripoli, Libya, pp. 510–514. doi: 10.1109/MI-STA61267.2024.10599656

Ansariyar, A. (2023) 'Providing a Comprehensive Traffic Safety Analysis Collected by Two LiDAR Sensors at a Signalized Intersection', Preprints, posted 23 October 2023. doi: 10.20944/preprints202310.1401.v1

Bai, Y., Yan, B., Zhou, C., Su, T. & Jin, X. (2023) 'State of art on state estimation: Kalman filter driven by machine learning', Annual Reviews in Control, 56, Article 100909. doi: 10.1016/j.arcontrol.2023.100909

Bewley, A., Ge, Z., Ott, L., Ramos, F. & Upcroft, B. (2016) Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP). IEEE.

Bharadwaj, S., Collins, R. & Liu, Y. (2024) 'R VPD: Recurrence Based Vanishing Point Detection', Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV 2025).

Bouncken, R.B., Czakon, W. & Schmitt, F. (2026) 'Purposeful sampling and saturation in qualitative research methodologies: recommendations and review', Review of Managerial Science, 20, pp. 579–615. https://doi.org/10.1007/s11846-025-00881-2

Breiman, L. (2001) 'Random forests', *Machine Learning*, 45, pp. 5–32. https://doi.org/10.1023/A:1010933404324

Cai, X., Li, Z., Qiao, W., Cheng, X., Peng, B. & Zhang, D. (2025) ‘Research on road traffic safety risk assessment based on the data of radar video integrated sensors’, Promet – Traffic & Transportation, 37(2). doi: 10.7307/ptt.v37i2.777

Chatzichristos, G. (2025) ‘Qualitative Research in the Era of AI: A Return to Positivism or a New Paradigm?’, International Journal of Qualitative Methods. https://doi.org/10.1177/16094069251337583

Chen, Q. (2025) ‘Traffic Object Detection Using YOLOv12’, Open Access Library Journal, 12, pp. 1–15. doi: 10.4236/oalib.1113991

Chen, T. & Guestrin, C. (2016) ‘XGBoost: A scalable tree boosting system’, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16), San Francisco, CA, 13–17 August. ACM. doi: 10.1145/2939672.2939785

Djukanović, S., Bulatović, N. & Čavor, I. (2022) ‘A dataset for audio-video based vehicle speed estimation’, 2022 30th Telecommunications Forum (TELFOR), Belgrade, Serbia, pp. 1–4. doi: 10.1109/TELFOR56187.2022.9983773

Fernández Llorca, D., Hernández Martínez, A. & García Daza, I. (2021) ‘Vision based vehicle speed estimation: A survey’, IET Intelligent Transport Systems, 15(8), pp. 987–1005. doi: 10.1049/itr2.12079

Geiger, A., Lenz, P., Stiller, C. & Urtasun, R. (2013) ‘Vision meets Robotics: The KITTI Dataset’, International Journal of Robotics Research. Available at: https://www.cvlibs.net/publications/Geiger2013IJRR.pdf (Accessed 5 Nov. 2025).

Grinsztajn, L., Oyallon, E. & Varoquaux, G. (2022) ‘Why do tree based models still outperform deep learning on typical tabular data?’, Advances in Neural Information Processing Systems 35 (NeurIPS 2022), Datasets and Benchmarks Track.

Han, Y., Kim, C., Jang, Y. & Kim, H.J. (2020) ‘Parametric analysis of KLT algorithm in autonomous driving’, 2020 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Korea (South), pp. 184–189. doi: 10.23919/ICCAS50221.2020.9268239

J.-H. Xu, J.-P. Li, Z.-R. Zhou, Q. Lv & J. Luo (2024) ‘A Survey of the Yolo Series of Object Detection Algorithms’, 2024 21st International Computer Conference on Wavelet Active Media

Technology and Information Processing (ICCWAMTIP), Chengdu, China, pp. 1–6. doi: 10.1109/ICCWAMTIP64812.2024.10873779

Karthika, B., Dharssinee, M., Reshma, V., Venkatesan, R. & Sujarani, R. (2024) 'Object Detection Using YOLO-V8', 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, pp. 1–4. doi: 10.1109/ICCCNT61001.2024.10724411

Li, C. et al. (2022) 'YOLOv6: A single-stage object detection framework for industrial applications', arXiv preprint, arXiv:2209.02976

Liao, K., Nie, L., Huang, S., Lin, C., Zhang, J., Zhao, Y., Gabbouj, M. & Tao, D. (2025) 'Deep Learning for Camera Calibration and Beyond: A Survey', arXiv preprint, arXiv:2303.10559v3. doi: 10.48550/arXiv.2303.10559

Liao, Y., Xie, J. & Geiger, A. (2023) 'KITTI 360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D', IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(3). doi: 10.1109/TPAMI.2022.3180408

Maity, M., Banerjee, S. & Sinha Chaudhuri, S. (2021) 'Faster R-CNN and YOLO based Vehicle detection: A Survey', 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, pp. 1442–1447. doi: 10.1109/ICCMC51019.2021.9418274

Macko, A., Gajdošech, L. & Kocur, V. (2025) 'Efficient Vision-based Vehicle Speed Estimation', arXiv preprint, arXiv:2505.01203. Available at: https://arxiv.org/abs/2505.01203 (Accessed 5 Nov. 2025).

Mareddy, S.K.R., Upplapati, D. & Antharam, D.K. (2025) 'Estimating Vehicle Speed on Roadways Using RNNs and Transformers: A Video based Approach', arXiv preprint, arXiv:2502.15545. Available at: https://arxiv.org/abs/2502.15545

Meng, Z., Xia, X., Xu, R., Liu, W. & Ma, J. (2026) 'HYDRO 3D: Hybrid Object Detection and Tracking for Cooperative Perception Using 3D LiDAR', IEEE Transactions on Intelligent Vehicles, 8(8). doi: 10.1109/TIV.2023.10148929

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine Learning in Python', Journal of Machine Learning Research, 12, pp. 2825–2830.

Simbeye, D.S. (2022) ‘Deployment of Inductive Loop Vehicle Traffic Counters Along Trunk Roads in Tanzania’, Tanzania Journal of Engineering and Technology, 41(4), pp. 119–132. doi: 10.52339/tjet.v41i2.796

Vuong, K., Tamburo, R. & Narasimhan, S.G. (2024) ‘Toward planet-wide traffic camera calibration’, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 8553–8562.

Wang, S. & Wang, Z. (2019) ‘Optical Flow Estimation with Occlusion Detection’, Algorithms, 12(5), 92. Available at: https://www.mdpi.com/1999-4893/12/5/92 (Accessed 5 Nov. 2025).

Wojke, N., Bewley, A. & Paulus, D. (2017) ‘Simple online and realtime tracking with a deep association metric’, 2017 IEEE International Conference on Image Processing (ICIP). IEEE. doi: 10.1109/ICIP.2017.8296962

Yang, B. (2023) ‘Research on Vehicle Detection and Recognition Technology Based on Artificial Intelligence’, Microprocessors and Microsystems, 104937. doi: 10.1016/j.micpro.2023.104937

8.0 APPENDICES

8.1 Full Feature Set

Feature	Category	Type	Notes
mean_width	Width dynamics	Existing	Mean bounding box width across window
width_growth	Width dynamics	Existing	(final_w − initial_w) / initial_w
width_growth_norm	Width dynamics	Derived (training)	width_growth / mean_width — not in CSV, computed at training time
width_expansion_rate	Width dynamics	New v3	(final_w − initial_w) / 1.0s — absolute px/s
std_width_delta	Width dynamics	New v3	Standard deviation of frame-by-frame width changes
max_width_delta	Width dynamics	New v3	Peak single-frame width expansion
width_change_rate	Width dynamics	New v3	Mean of frame-by-frame width deltas — **PRUNED**
mean_height	Height dynamics	Existing	Mean bounding box height across window
max_height	Height dynamics	Existing	Maximum bounding box height across window
height_growth	Height dynamics	Existing	(final_h − initial_h) / initial_h
height_change_rate	Height dynamics	Existing	Mean of frame-by-frame height deltas

std_height_delta	Height dynamics	New v3	Standard deviation of frame-by-frame height changes
height_expansion_rate	Height dynamics	New v3	(final_h − initial_h) / 1.0s — **PRUNED**
aspect_ratio	Aspect ratio	Existing	mean_width / mean_height
aspect_ratio_change_rate	Aspect ratio	New v3	(final_ar − initial_ar) / 1.0s
TTC	Derived	New v3	mean_width / width_expansion_rate — time-to-contact estimate
mean_dx	Centroid / motion	Existing	Mean horizontal centroid displacement per frame
max_dx	Centroid / motion	Existing	Maximum horizontal centroid displacement
std_dx	Centroid / motion	Existing	Standard deviation of horizontal displacement
mean_dy	Centroid / motion	Existing	Mean vertical centroid displacement per frame
max_dy	Centroid / motion	Existing	Maximum vertical centroid displacement
std_dy	Centroid / motion	Existing	Standard deviation of vertical displacement
std_euclidean	Centroid / motion	Existing	Variation in frame-to-frame Euclidean centroid distance
dx_dy_ratio	Centroid / motion	Existing	Ratio of horizontal to vertical motion

mean_pixel_speed	Centroid / motion	Existing	Mean Euclidean centroid velocity × fps
track_length	Meta	Existing	Total track duration in frames — **PRUNED**
vehicle_class	Meta	New v3	YOLO class ID (car, motorbike, bus, truck) — **PRUNED**

8.2 Tuned XGBoost Hyperparameters

Parameter	Default Value	Tuned Value	Interpretation
n_estimators	100	441	More trees to compensate for lower learning rate
learning_rate	0.1	0.0931	Slower learning — reduces overfitting
max_depth	6	4	Shallower trees — less complexity per tree
min_child_weight	1	10	Prevents splits on small sample clusters
subsample	1.0	0.7213	Row subsampling — introduces stochasticity
colsample_bytree	1.0	0.8171	Feature subsampling per tree
reg_alpha	0	0.2503	L1 regularisation
reg_lambda	1	0.6748	L2 regularisation

8.3 Model Version History

Version	CV MAE	CV RMSE	CV R^2	HO MAE	HO RMSE	HO R^2	Mazda3 MAE
v1 — pass-by anchor, 3s window, default XGBoost	3.86	4.89	0.940	—	—	—	—
v2 — exit anchor, 1s window, default XGBoost	3.16 ±0.22	4.93 ±1.01	0.937	4.16	6.58	0.906	5.09
v2 + expanded features (pre-tuning)	3.16 ±0.27	4.86 ±1.18	0.938	4.15	6.21	0.917	4.77
v3 — width_growth_norm, pruned, default XGBoost	2.86 ±0.17	3.87 ±0.54	0.963	—	—	—	—
v3 tuned — final submitted system	**2.44 ±0.26**	**3.36 ±0.68**	**0.971**	**3.66**	**4.88**	**0.949**	**3.14**

8.4 Sample YOLO Object Detection Inference

0: 384x640 (no detections), 28.5ms
Speed: 2.2ms preprocess, 28.5ms inference, 0.4ms postprocess per image at shape (1, 3, 384, 640)

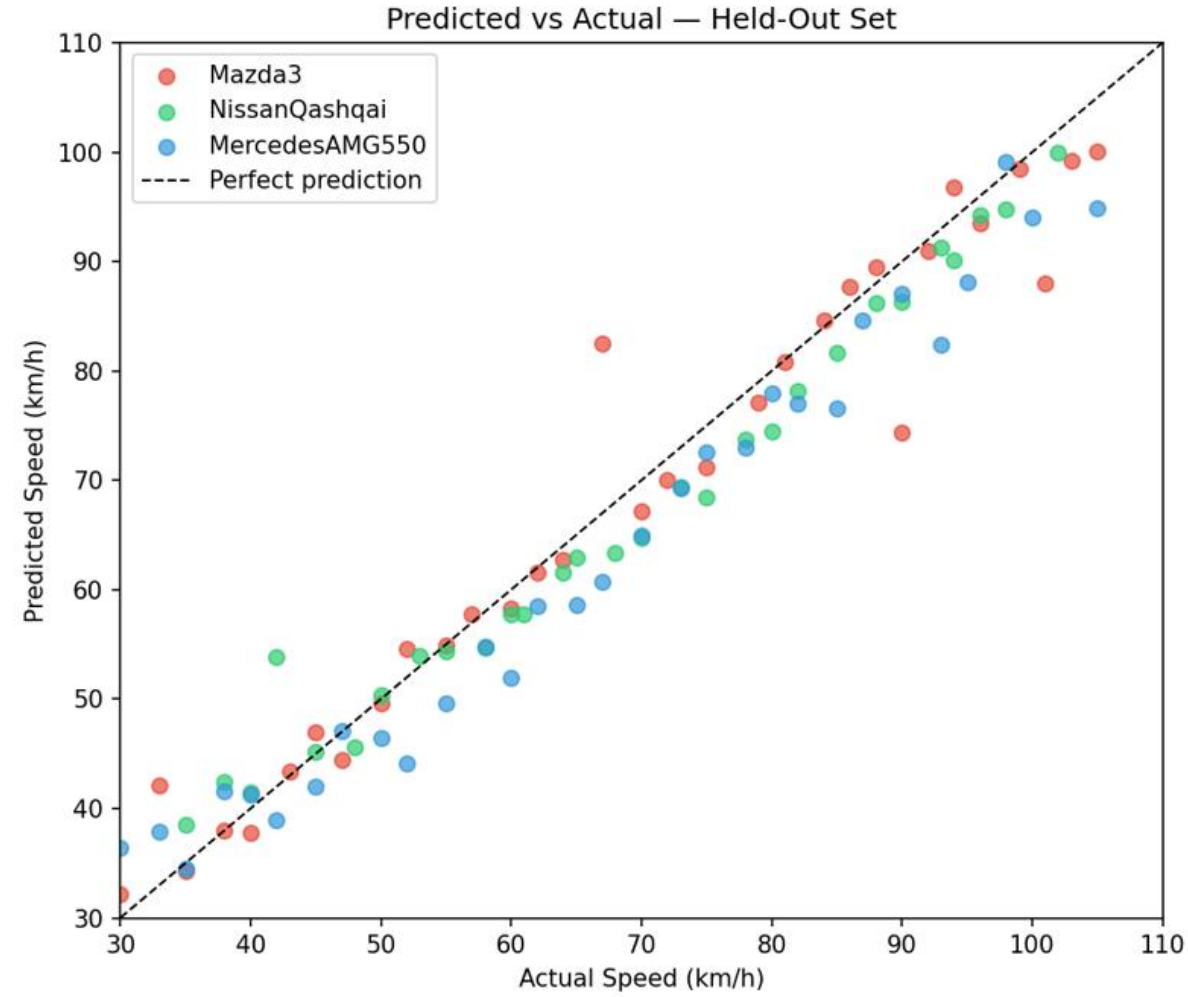
0: 384x640 (no detections), 26.9ms
Speed: 1.8ms preprocess, 26.9ms inference, 0.3ms postprocess per image at shape (1, 3, 384, 640)

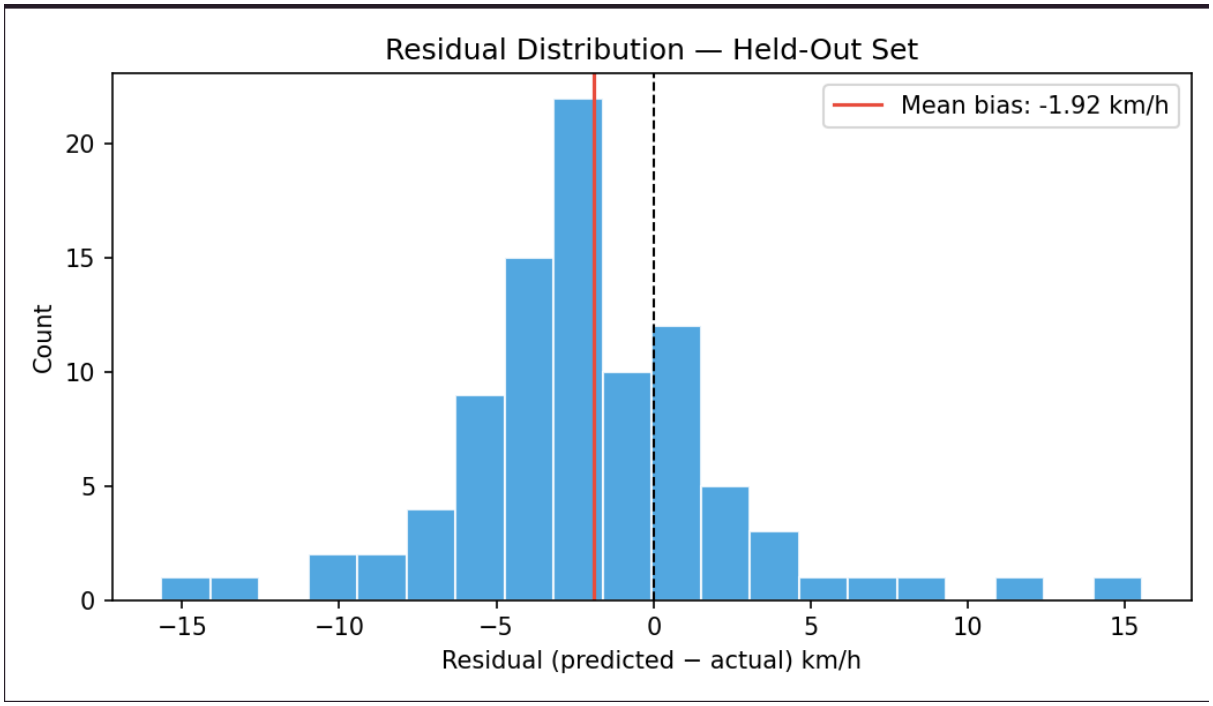
0: 384x640 1 car, 25.0ms
Speed: 1.9ms preprocess, 25.0ms inference, 0.8ms postprocess per image at shape (1, 3, 384, 640)

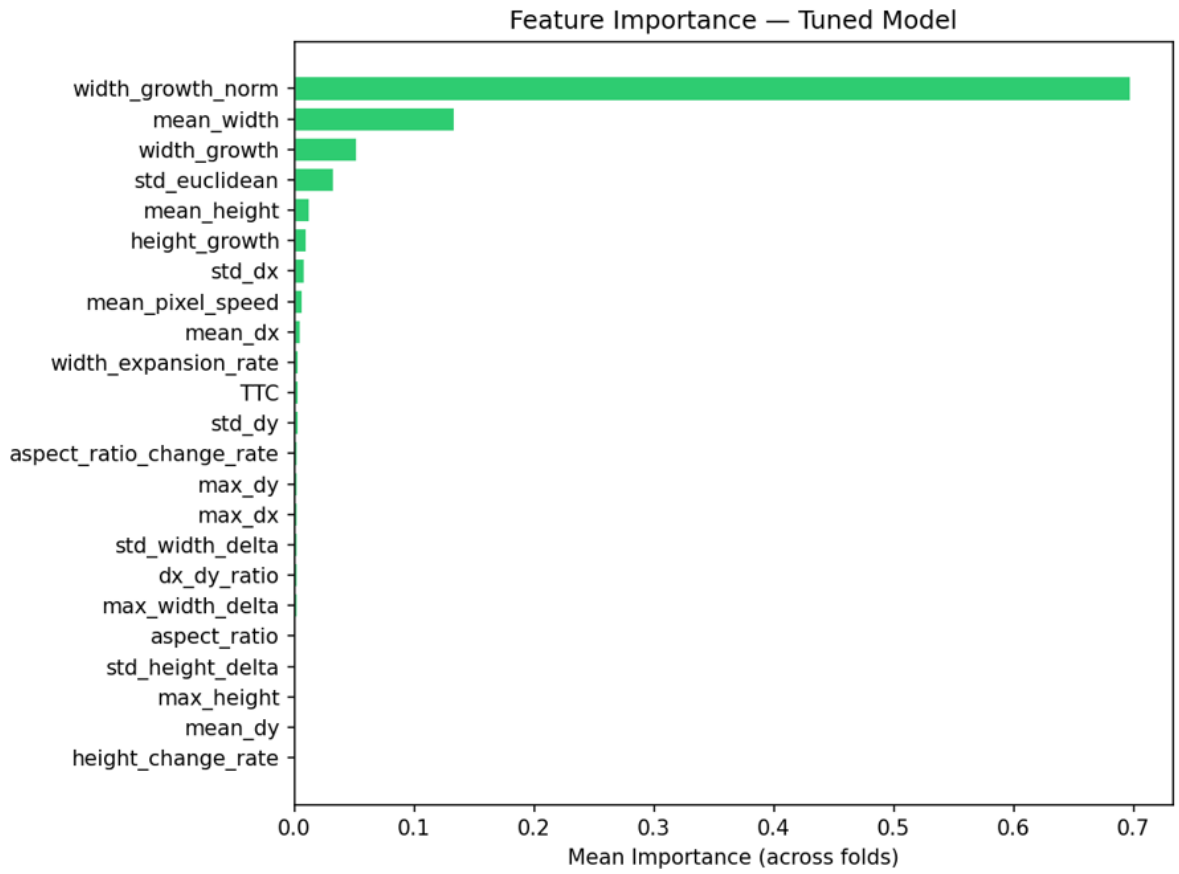
0: 384x640 1 car, 32.8ms
Speed: 1.4ms preprocess, 32.8ms inference, 0.9ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 1 car, 30.8ms
Speed: 2.3ms preprocess, 30.8ms inference, 0.6ms postprocess per image at shape (1, 3, 384, 640)

8.5 Results Charts


Predicted vs Actual — Held-Out Set
Mazda3
NissanQashqai
MercedesAMG550
Perfect prediction
Predicted Speed (km/h)
Actual Speed (km/h)
30
40
50
60
70
80
90
100
110


Residual Distribution — Held-Out Set
Mean bias: -1.92 km/h
Count
Residual (predicted − actual) km/h
0
5
10
15
20
−15
−10
−5
0
5
10
15


Feature Importance — Tuned Model
width_growth_norm
mean_width
width_growth
std_euclidean
mean_height
height_growth
std_dx
mean_pixel_speed
mean_dx
width_expansion_rate
TTC
std_dy
aspect_ratio_change_rate
max_dy
max_dx
std_width_delta
dx_dy_ratio
max_width_delta
aspect_ratio
std_height_delta
max_height
mean_dy
height_change_rate
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Mean Importance (across folds)