Monocamera-Based Traffic Analysis Using AI for Vehicle Position, Dimension and Speed Estimation
Monokamera-basierte Verkehrsanalyse durch KI zur Schätzung von Fahrzeugposition, -abmessungen und -geschwindigkeit

Master's Thesis

Oleg Porokhniak
Department of Electrical and Computer Engineering
Chair of Electromobility
Supervisors:
Prof. Dr.-Ing. Daniel Görges
Julian Hund, M.Sc.
Dzmitry Maladzenkau, M.Sc.

RPTU
Rheinland-Pfälzische Technische Universität
Kaiserslautern
Landau

February 26, 2025

Affidavit (Eidesstattliche Erklärung)

This thesis has been prepared on initiative and under guidance of my advisors. For the preparation I have not used others than the indicated aids.

Kaiserslautern, February 26, 2025

Oleg Porokhniak

Contents

List of Figures

List of Tables

List of Algorithms

Notation, Indices and Abbreviations

Notation

$\mathbf{A}$	**Fat** upper case letters are matrices
$\mathbf{v}$	**Fat** lower case letters are vectors
$\mathbf{u}, \mathbf{v}$	Image coordinates (in pixels)
$\mathbf{X}, \mathbf{Y}$	Real-world ground-plane coordinates (in meters)

Indices

$X_{\text{cam}}, Y_{\text{cam}}$	Fixed camera position in world coordinates
X', Y'	Transformed coordinates in homogeneous representation
$S_{\text{real},1}, S_{\text{real},2}$	Real-world bounding box widths at two different frames
α_1, α_2	Vehicle orientation angles at two different times
d_t	Displacement between consecutive frames
$v_{m/s}, v_{km/h}$	Instantaneous speed in meters per second and kilometers per hour

Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
FOV	Field of View
FPS	Frames Per Second
LiDAR	Light Detection and Ranging
MPC	Model Predictive Control
RANSAC	Random Sample Consensus
RHC	Receding Horizon Control
SSD	Single Shot MultiBox Detector
SVD	Singular Value Decomposition
YOLO	You Only Look Once (Object Detection Model)

1 Introduction

The growing demand for intelligent transportation systems has driven the need for efficient and cost-effective methods for monitoring and analyzing traffic. Traditional approaches to vehicle detection and tracking often rely on multiple sensors, such as LiDAR (Light Detection and Ranging), radar, and stereo cameras, which, while accurate, can be expensive and complex to integrate. With advancements in deep learning and computer vision, monocular camera-based systems have emerged as a promising alternative. These systems leverage powerful object detection models to extract valuable information from video streams, enabling real-time vehicle tracking, speed estimation, and size measurement. However, despite recent progress, achieving high accuracy in estimating vehicle dimensions and velocities from a single camera remains a significant challenge due to the inherent loss of depth information in two-dimensional images. This thesis explores a novel approach to addressing this limitation by combining deep learning-based detection with geometric transformations to infer real-world vehicle parameters from monocular footage.

1.1 Motivation

Road safety, traffic management, and infrastructure planning rely a lot on the accurate measurement of vehicle dynamics and physical parameters. Conventional techniques that rely on multiple hardware sensors (e. g. LiDAR, radar, stereo cameras) can offer high accuracy but can increase cost and complexity of the system. These downsides could be avoided by using a single monocular camera instead. Recent research has leveraged deep learning for robust real-time object detection, making it feasible to track individual vehicles on a video stream and compute metrics such as speed or bounding-box geometry [Sax+24; Naa+24]. Nevertheless, the consistent estimation of both speed and physical dimensions (length and width) from a single monocular view remains challenging because of the lack of depth information on a simple two-dimensional video. The proposed project addresses these challenges by developing a single-camera system that detects vehicles using only an object detection network with bounding-box outputs and leverages geometric transformations to estimate each vehicle's length and width. In contrast to recent multi-sensor strategies, the method described

here seeks to minimize hardware while retaining acceptable accuracy for on-road vehicle monitoring.

1.2 Introduction to Single-Camera Approaches

Video-based systems for vehicle speed detection have gained prominence due to their cost-effectiveness and flexibility [Naa+24; PJC21]. These approaches generally rely on object detection and tracking, where vehicles are identified in each frame using bounding boxes generated by neural networks such as You Only Look Once (YOLO). Once detected, geometric or depth estimation techniques are applied to infer real-world distances from pixel measurements, enabling the calculation of speed and other physical parameters. A detailed discussion of YOLO and its role in this study is provided in Section 2.1. In contrast, multi-camera or sensor fusion approaches, such as those incorporating radar or LiDAR, offer direct range measurements and the ability to fuse data from multiple perspectives. While these methods can improve accuracy, they also introduce increased setup costs and calibration complexity. This trade-off highlights the advantage of single-camera solutions, which aim to balance affordability and ease of deployment while maintaining acceptable levels of accuracy in real-world applications.

1.3 State of the Art

The field of vehicle detection and motion analysis has seen significant advancements due to the integration of various sensing technologies and machine learning techniques. Traditional methods rely on multi-sensor fusion, combining inputs from cameras, radar, and LiDAR to enhance accuracy in estimating vehicle position and speed. More recently, deep learning-driven monocular camera approaches have gained attention as a cost-effective alternative, leveraging object detection models and geometric transformations to infer real-world parameters from a single viewpoint.

1.3.1 Multi-Sensor and Radar Fusion

A variety of autonomous driving systems utilize camera–radar or camera–lidar fusion to estimate velocity with high accuracy [PJC21]. Radar provides precise velocity data but typically covers a narrow field of view (FOV), while the camera observes a wider scene but struggles with depth estimation. Combining the two allows one sensor's strengths to compensate for the other's weaknesses; however,

such an approach demands careful sensor-to-sensor calibration that can be fragile in dynamic, real-world conditions [PJC21; Oh+18].

Radar-camera calibration methods typically rely on affine transformations, homography, or extrinsic parameter estimation [Oh+18]. While these methods achieve high accuracy, they require frequent recalibration to maintain reliability. In contrast, this work eliminates the need for radar by leveraging a purely geometric approach to estimate vehicle speed and dimensions from a single monocular camera.

1.3.2 Deep Learning–Driven Single-Camera Methods

Several recent papers explore the exclusive use of one camera to estimate speed. For instance, [CD23] and [Naa+24] utilize YOLO for vehicle detection. In [CD23], 1D Convolutional Neural Networks (1D-CNNs) are employed to analyze the bounding-box area changes over time, while [Naa+24] applies Depth Estimation from single-view images to compensate for scale ambiguities. These methods achieve promising accuracy for speed detection, sometimes within a 2 km/h error [CD23]. Nevertheless, they do not provide direct measurements of the vehicle's physical length and width.

1.3.3 Object Detection and Dimension Estimation

While monocular methods exist for distance or velocity estimation [Wu+15], directly extracting vehicle dimensions (length and width) remains less common. Depth-based approaches [Naa+24] rely on advanced neural networks that infer scene depth, but they can be sensitive to training domain shifts or require large labeled datasets for accurate depth supervision. Stereo or multi-camera systems [Wu+15] can estimate width and length more directly by triangulating multiple viewpoints, albeit at an increased hardware cost and complexity. Single-view geometry models can estimate vehicle dimensions by recognizing common vehicle types and using their known real-world measurements as calibration references [KDA22]. However, this approach relies on prior knowledge of specific vehicle models and may not generalize well to unseen vehicle types.

1.3.4 Contributions of This Work

This work proposes a single-camera vehicle tracking system that minimizes hardware requirements while maintaining accurate speed and dimension estimation. A GoPro Hero 5 Session camera is positioned perpendicular to the road to cap-

ture passing vehicles, avoiding the need for multi-sensor setups such as radar or LiDAR. Vehicle detection is performed using YOLOv8, allowing bounding boxes to be extracted in a single neural network pass without requiring additional depth sensors or multiple camera viewpoints. A key contribution of this study is the introduction of a geometric model that estimates vehicle dimensions by accounting for perspective effects. Unlike conventional approaches that rely on depth estimation or predefined vehicle templates, this method decouples a vehicle's length and width from its bounding-box width by incorporating the vehicle's orientation relative to the camera. By explicitly considering the changing proportions of the bounding box as a vehicle moves through the field of view, the model provides a more adaptable approach to dimension estimation without requiring prior knowledge of specific vehicle types. Additionally, this approach avoids dependence on complex depth-learning models or optical-flow-based velocity estimation, opting instead for a geometry-driven technique that is interpretable and computationally efficient. The combination of neural network-based object detection with a mathematically grounded transformation model makes this method well-suited for cost-effective traffic monitoring applications.

1.4 Thesis Objective

The primary objective of this work is to develop and validate a cost-effective, single-camera traffic analysis system capable of estimating vehicle speed and physical dimensions, such as length and width, using only bounding-box detections. The system is designed to maintain robust performance under practical conditions, accounting for moderate variations in both the viewing angle and the distance between the camera and the observed vehicles. The subsequent chapters detail the camera calibration process, the implementation of YOLOv8 for object detection, and the methodologies used to calculate real-world vehicle size and speed. Proof-of-concept experiments are conducted to evaluate detection performance and the accuracy of the estimated dimensions and velocity across diverse traffic scenarios.

1.5 Thesis Outline

The organization of this thesis follows a logical progression from conceptual background to experimental validation:

Chapter 2 presents the object recognition model, discussing the fundamentals of object detection, the choice of YOLOv8, the dataset used for training, and evaluation of results.

Chapter 3 details the recording setup, including camera specifications, placement strategy, calibration process, and the experimental procedure used to validate the system.

Chapter 4 introduces the mathematical foundations required for accurate vehicle size and speed estimation, covering camera calibration, homography, and the estimation of vehicle dimensions.

Chapter 5 describes the software architecture, outlining the components that integrate camera calibration, neural network inference, coordinate transformations, and result processing.

Chapter 6 presents the results and evaluation of the system, analyzing speed estimation accuracy, vehicle size estimation, and error sources affecting performance.

Chapter 7 concludes the thesis by summarizing key findings, discussing limitations, and proposing potential directions for future work.

By placing emphasis on simplicity of hardware, robust deep learning detection, and a well-defined geometric model, the single-camera method proposed here has the potential to streamline on-road vehicle measurements at scale.

2 Object Recognition Model

Object detection is a fundamental task in computer vision that involves identifying and localizing objects within an image or video frame. Unlike simple classification, which determines the presence of an object in an image, object detection not only recognizes objects but also provides their spatial location using bounding boxes. This capability is essential for real-world applications such as autonomous driving, surveillance, and traffic monitoring. In the context of this study, object detection is used to identify and track vehicles across consecutive video frames. The bounding box data is then transformed into real-world coordinates for speed and size estimation. As described in Subsection 4.3.1, an extension of object detection known as keypoint detection was initially explored to identify wheel contact points, but due to inconsistencies in performance, the final implementation focused solely on bounding box tracking. Accurate vehicle detection and tracking are fundamental to this study, as they form the basis for speed and size estimation. The object recognition model is responsible for identifying vehicles in each frame and assigning unique tracking IDs to maintain continuity across multiple frames. This chapter describes the object recognition pipeline, the dataset used for training, the model architecture, the training process, and an evaluation of the results.

2.1 Object Detection Models

Several object detection models were considered for this system, each with distinct advantages and trade-offs. Faster R-CNN (Faster Region Convolutional Neural Network) is a two-stage detection model known for its high accuracy, but it is significantly slower compared to other architectures, making it unsuitable for real-time applications [IBM23]. SSD (Single Shot MultiBox Detector) is a lightweight model that offers faster inference than Faster R-CNN, though it generally performs worse in detecting small or overlapping objects [IBM23]. Detectron2, based on the Mask Region-Based Convolutional Neural Network (Mask R-CNN), is a powerful model from Facebook AI that extends object detection to instance segmentation. However, it is computationally expensive and slower than other options [Lab24]. Table 2.1 compares object detection models based on Frames Per Second (FPS), which indicates processing speed, and mean Average

Precision (mAP), which measures detection accuracy. FPS reflects how many frames the model processes per second, while mAP quantifies precision across different thresholds. The table also highlights whether each model supports keypoint detection.

Model	**Speed (FPS)**	**Accuracy (mAP)**	**Keypoint Detection**
YOLOv8	Fast (50+ FPS)	High	Yes
Faster R-CNN	Slow (5-10 FPS)	Very High	No
SSD	Medium (15-30 FPS)	Medium	No
Detectron2 (Mask R-CNN)	Slow (5-10 FPS)	Very High	Yes

Table 2.1: Comparison of Object Detection Models

Considering the trade-offs between accuracy, speed, ease of deployment, and the ability to detect keypoints, YOLOv8 was selected as the most suitable model for this system. The You Only Look Once (YOLO) family of models is widely regarded as one of the most efficient and accurate architectures for real-time object detection. YOLOv8, the latest iteration, offers significant improvements in detection accuracy, speed, and ease of deployment. In addition to standard bounding-box detection, YOLOv8 also supports pose estimation, allowing for the identification of key points on detected objects. YOLOv8 achieves high-speed inference, making it suitable for real-time vehicle tracking. The model balances speed and accuracy through efficient feature extraction and improved detection capabilities. Unlike traditional YOLO models that detect only bounding boxes, YOLOv8 Pose extends the architecture to include keypoint detection, enabling additional annotations such as wheel contact points. Furthermore, the Ultralytics YOLOv8 implementation provides built-in training utilities, streamlining the training and fine-tuning process.

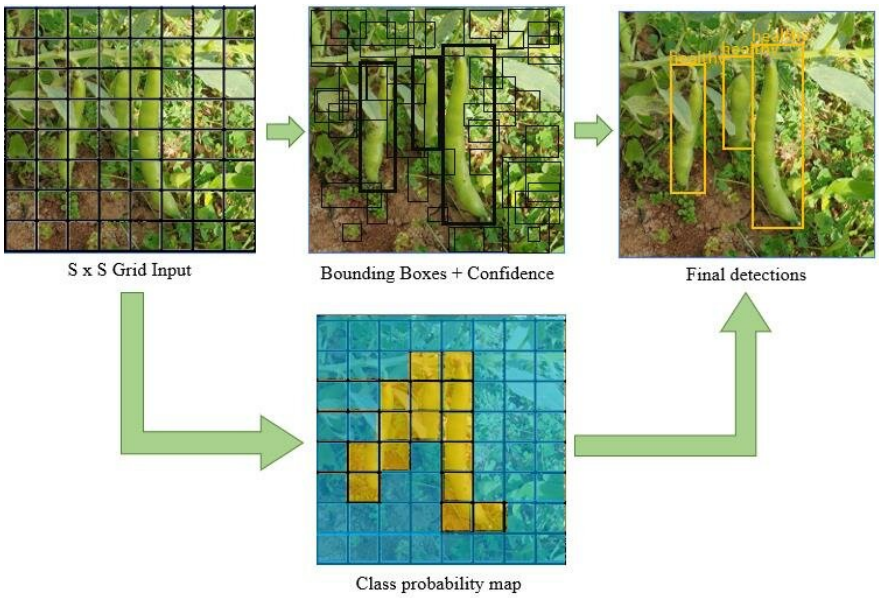

Figure 2.1: YOLO object detection pipeline.

Figure 2.1 illustrates the YOLO object detection process. The model divides the input image into an $S \times S$ grid, with each grid cell responsible for predicting bounding boxes and confidence scores for objects within its region. A class probability map is generated, indicating the likelihood of different objects in various grid cells. After non-maximum suppression (NMS) is applied, the final detections are obtained, highlighting objects with the highest confidence scores. Considering the trade-offs between accuracy, speed, ease of deployment, and the ability of YOLOv8 to detect keypoints, it was selected as the most suitable model for this system.

2.2 Detection Approach

The detection system was designed to achieve several objectives. It assigns a rectangular bounding box to each detected vehicle, ensuring precise localization. Additionally, it identifies points where the vehicle's wheels touch the ground, a feature initially intended to provide more accurate information about vehicle orientation. To maintain continuity, the system tracks detected vehicles across multiple frames using persistent track IDs. While the bounding box detection and tracking worked reliably, the wheel contact point detection faced challenges, particularly due to inconsistencies when applied to the final camera position. As a result, the focus shifted solely to bounding-box tracking.

2.3 Training Data and Annotation

The dataset used for training consisted of 283 images, manually annotated using Roboflow [Rob24], an online platform that facilitates dataset management, annotation, and augmentation for computer vision tasks. Roboflow provides tools for labeling, preprocessing, and exporting datasets in multiple formats, streamlining the training process for machine learning models. The images were sourced from various locations and perspectives to ensure generalizability. As shown in Figure 2.2, each image contains labeled vehicles with bounding box annotations and wheel contact points, which are marked as either visible or occluded.

Figure 2.2: Example of Annotated Training Image

2.4 Model Training

The model was trained using two different strategies to evaluate performance. Each approach had its' own method for annotating the wheel contact points: The training process was conducted using two different approaches. The first approach, Training with Visibility Labels, involved adding a custom visibility prediction head to YOLOv8 Pose. The model was trained to classify wheel contact points as either visible or occluded, incorporating a binary cross-entropy term in the loss function to optimize classification accuracy. In the second approach, Training Without Visibility Labels, the visibility classification layer was removed, and only visible keypoints were annotated and used for training. This modification led to a simplified loss function, which improved convergence and reduced inconsistencies in keypoint detection. Ultimately, this second approach was adopted as the final training method due to its better overall performance.

2.4.1 Training Configuration

The YOLOv8n-Pose model was trained using a configuration optimized for both performance and efficiency. As shown in Table 2.2, an image size of 640×640 px balances detection accuracy and processing speed. A batch size of 16 was chosen to optimize GPU memory usage, ensuring stable training. The model was trained for 100 epochs, providing sufficient time for convergence. The AdamW optimizer was used for improved weight decay handling, with a learning rate of 0.001 to maintain training stability. All training was conducted on a GPU to accelerate computation. The Ultralytics YOLOv8 framework was used for dataset loading, augmentation, and optimization.

Parameter	**Value**
Model	YOLOv8n-Pose
Image Size	640×640 px
Batch Size	16
Epochs	100
Optimizer	AdamW
Learning Rate	0.001
Hardware	GPU

Table 2.2: Training Configuration for Object Detection Model

The training scripts used the Ultralytics YOLOv8 framework to handle dataset loading, augmentation, and optimization.

2.5 Evaluation of Results

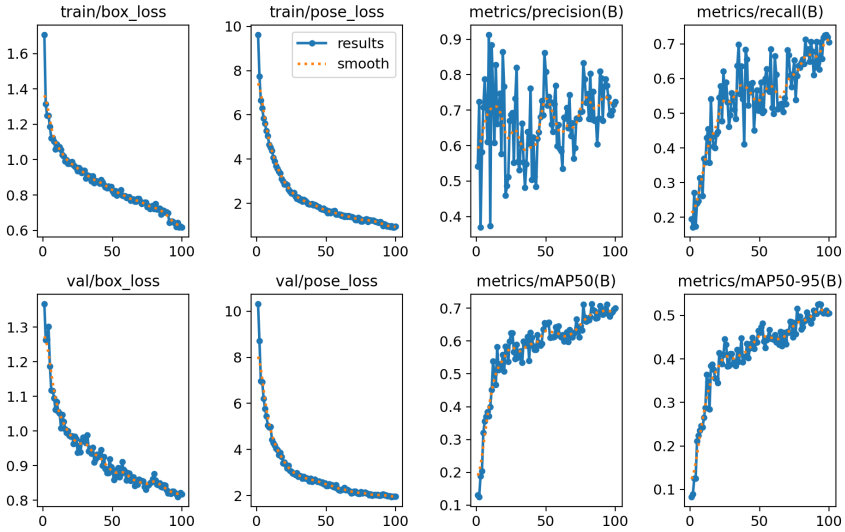

Figure 2.3: Training results

The trained model shows a steady decrease in box loss and pose loss, indicating effective learning with minimal overfitting (Figure 2.3). Precision fluctuates in early epochs but stabilizes later, while recall improves consistently, reaching around 0.7. mAP, a key detection metric, measures accuracy across different Intersection-over-Union (IoU) thresholds. mAP50, which evaluates detections at a fixed 50% IoU threshold, reaches approximately 0.7, while mAP50-95, which averages mAP over multiple IoU thresholds (50% to 95%), stabilizes near 0.5, indicating moderate overall accuracy. However, keypoint detection remains less precise, which may impact applications requiring fine-grained localization. Further improvements could be achieved through hyperparameter tuning or increasing dataset diversity.

2.5.1 Key Observations

Bounding box detection performed well, with the model successfully detecting vehicles with high precision and recall. However, wheel contact points showed inconsistencies. The visibility classification model struggled to differentiate between occluded and visible points, while a simplified model performed better but still exhibited inconsistencies.

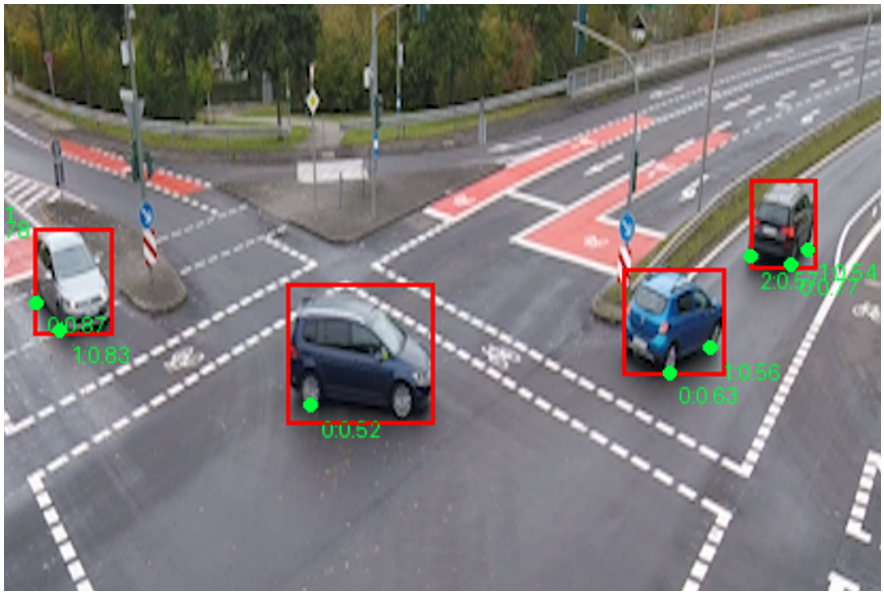

Figure 2.4: Wheel Contact Points inconsistency

Due to these limitations, wheel contact points were not used in the final system. However, if a dataset was specifically collected and labeled for the fixed camera perspective, future versions could reintroduce this feature for improved angle estimation.

2.6 Summary

This chapter described the object recognition model used in the study. Initially, a model was developed to detect both bounding boxes and wheel contact points using YOLOv8 Pose, but due to inconsistencies in keypoint detection, only bounding boxes were used in the final implementation. Despite strong performance in general vehicle detection, the dataset's diverse perspectives reduced the model's effectiveness for the fixed camera setup used in real-world testing. A dedicated dataset for the fixed viewpoint could improve future versions of the system. The following chapters explain how the detected bounding boxes are utilized for coordinate transformation, vehicle speed estimation, and size measurement.

3 Recording Setup

Accurate vehicle detection and measurement rely on a well-calibrated camera setup and a structured experimental approach. To achieve this, it is essential to consider the camera's specifications, including its field of view, resolution, and lens characteristics, as these factors directly impact the calibration process. This chapter first examines these specifications and their influence on system accuracy. It then discusses the placement strategy, designed to maximize road coverage while minimizing perspective distortions. Beyond camera setup, precise real-world coordinate mapping is crucial for accurate measurements. For this task, key reference points, such as the camera's position and road markers, were carefully recorded. Finally, the experimental procedure is outlined, explaining how vehicles with known dimensions were driven at controlled speeds to generate reference data for evaluating the accuracy of the system's speed and size estimations.

3.1 Camera Specifications

Figure 3.1: GoPro Hero 5 Session

A GoPro Hero 5 Session was selected as the primary camera for video recording due to its wide field of view and high-resolution capabilities. The wide-angle lens allows for coverage of a larger portion of the road, enabling vehicle detection across multiple lanes and providing more spatial context. However, this advantage comes with the drawback of barrel distortion, a common effect of fisheye lenses that causes straight lines to appear curved. This distortion must be carefully corrected through geometric calibration to ensure accurate coordinate transformations and vehicle measurements.

3.1.1 Recording Settings

To ensure both maximum field of view and high frame resolution for capturing sufficient details for object recognition, the videos were recorded using specific settings [GoP25]. The resolution was set to 4K (3840x2160 pixels) to capture fine details necessary for vehicle detection. A frame rate of 24 frames per second was selected to balance smooth motion capture with efficient storage. The widest field of view setting was used, resulting in an approximate horizontal field of view of 123°. To minimize vibrations and misalignment during setup, the camera was remotely operated via the GoPro mobile app. Table 3.1 summarizes the recording parameters used in this study.

Parameter	**Setting**
Resolution	4K (3840x2160px)
Frame Rate	24 FPS
Field of View	123°.
Control Method	Remote operation via the GoPro mobile app.

Table 3.1: Recording settings used for video capture

3.1.2 Camera Lens and Distortion Characteristics

The GoPro Hero 5 Session is equipped with a fixed-focus ultra-wide-angle lens that introduces severe barrel distortion, especially at the edges of the frame. This distortion is a characteristic of fisheye lenses, where straight lines appear curved, and objects further from the optical center experience greater geometric deformation. Given the extreme distortion present in raw GoPro footage, geometric transformations such as homography mapping and coordinate conversion would be inaccurate without proper calibration. The calibration steps, detailed in Section 3.5, ensure that bounding boxes represent real-world vehicle dimensions correctly.

3.2 Camera Placement and Road Alignment

To ensure a clear and stable view of passing vehicles, the GoPro Hero 5 Session was mounted on a tripod at a height of 3 meters. This elevation was chosen to provide visibility over all lanes while minimizing obstructions and perspective distortion. The camera was positioned adjacent to a straight section of the road, allowing a sufficiently large segment of the roadway to be captured within the frame. The tripod setup ensured stability and reduced vibrations that could impact image consistency. Additionally, the camera was remotely controlled via a mobile application to avoid unintended shifts in position after alignment. For accurate speed and size estimation, the camera's optical axis was carefully aligned perpendicular to the road to reduce perspective distortions. This orientation ensured that vehicles remained within a consistent field of view and that their bounding boxes could be mapped accurately to real-world coordinates. To ensure an effective measurement area, the camera was positioned at a distance that allowed it to capture a sufficiently large section of the roadway, enabling vehicle tracking over multiple frames. Proper placement and alignment ensured that the recorded footage provided a uniform and unobstructed view of the road across the entire field of view.

3.3 Coordinate Measurements

Accurate coordinate measurements are essential for geometric transformations, ensuring precise mapping between image space and real-world positions. The three-dimensional position of the camera was recorded by measuring its height above the road surface, which was determined to be 3 meters. The horizontal position was defined using a measuring tape relative to a fixed reference point on the ground, which was set as the origin $(0, 0)$ to maintain consistency in further calculations. The camera coordinates were determined to be $(-0.21, -8.37, 3.00)$ meters.

3.3.1 Road Reference Point Measurement

To facilitate homography-based coordinate mapping, which is explained in detail in Section 4.2, a LiDAR device was used to precisely measure at least four reference points on the road surface. These four points are necessary to compute the homography matrix, enabling the transformation of detected objects from pixel coordinates to real-world locations. While four points are the minimum required for this transformation, using additional reference points can improve the robustness of the mapping and reduce errors caused by measurement noise or

road surface irregularities. For the selection of reference points, two were carefully positioned on an axis parallel to the road, forming the X-axis for the coordinate system. Additional points were distributed across the field of interest to ensure that the homography transformation remains accurate throughout the measurement area. Furthermore, points were placed at varying distances from the camera to minimize extrapolation errors and improve mapping accuracy across different regions of the scene. By carefully selecting and measuring these reference points, the system maintains accurate real-world coordinate mapping, which is critical for reliable speed and size estimation.

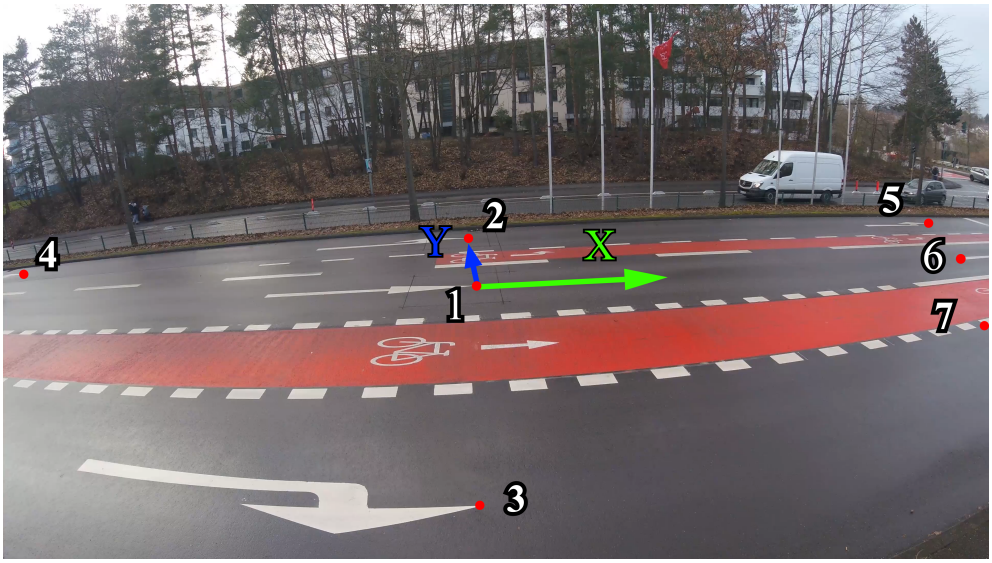

Figure 3.2: Reference points used for coordinate mapping.

Point	X (m)	Y (m)
1	0.0000	0.0000
2	-0.2587	4.8672
3	-0.1187	-4.3256
4	-20.2680	4.8034
5	20.7026	4.9524
6	16.2906	0.0000
7	11.3906	-4.2470
Camera	-0.21	-8.37

Table 3.2: Real-world coordinates of reference points and camera position.

Figure 3.2 shows the selected reference points used for coordinate mapping and the coordinate axes. These points were measured using a LiDAR device and

are distributed across the road to ensure accurate homography transformation. Table 3.2 provides their corresponding real-world coordinates, with Point 1 as the origin $(0, 0)$ and point 6 used for setting the direction of X-axis. These reference points enable precise conversion of detected vehicle positions from image space to real-world coordinates for speed and size estimation. Additionally, Figure 3.3 extends this representation by including the camera position, illustrating its placement relative to the reference points.

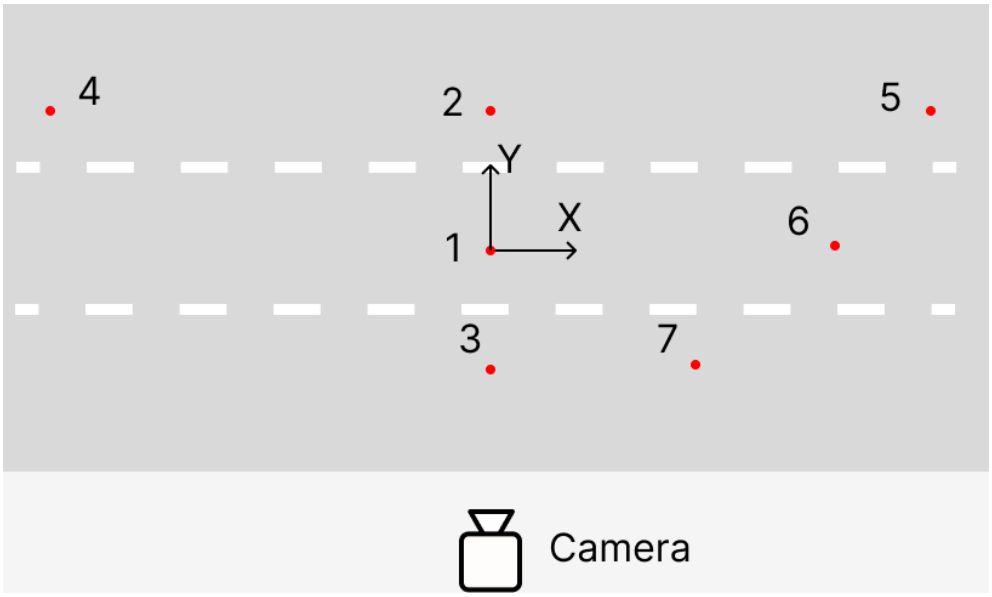

Figure 3.3: Camera position.

3.4 Experimental Procedure

To evaluate the accuracy of the proposed system, a controlled experiment was conducted using two vehicles of different sizes. Their specifications are summarized in Table 3.3.

Car Model	**Length (m)**	**Width (m)**
BMW 330e	4.71	1.83
Volkswagen Golf	4.25	1.76

Table 3.3: Specifications of the vehicles used in the experiment.

Each vehicle was driven along the test road section four times at controlled speeds of 20km/h, 30km/h, 40km/h, and 50km/h. The ground-truth velocity was estimated using an onboard speedometer, which provides a practical but not perfectly

precise reference, as the speed was manually maintained by the driver. During each run, the GoPro Hero 5 Session recorded the vehicles as they passed through the camera's field of view. Experiments were conducted during daytime under stable weather conditions to minimize illumination changes. The selected road section supports typical urban or suburban driving speeds (20–50km/h).

3.5 Calibration Process

To ensure accurate mapping between image coordinates and real-world positions, the geometric distortions introduced by the GoPro Hero 5 Session's fisheye lens must be corrected. The camera calibration process was conducted to estimate the intrinsic parameters and distortion coefficients as described in Section 3.5.1, which are subsequently used to undistort all recorded frames. This correction is crucial for minimizing errors in vehicle detection, tracking, and measurement.

3.5.1 Lens Distortion and Camera Parameters

The wide-angle lens of the GoPro Hero 5 Session introduces significant barrel distortion, a common effect in fisheye lenses where straight lines appear curved outward. This distortion is most pronounced at the edges of the image and must be accounted for to ensure accurate coordinate transformations. The calibration process determines two key sets of parameters: intrinsic camera parameters and distortion coefficients, both of which are essential for correcting lens-induced distortions. A detailed mathematical formulation of these parameters is provided in Chapter 4. Intrinsic parameters define the camera's internal characteristics, including focal length and the principal point. The focal length represents how strongly the lens converges light and determines the scaling factor between real-world distances and image coordinates. A longer focal length results in a narrower field of view, whereas a shorter focal length increases the captured scene width. The principal point refers to the optical center of the image, ideally located at the center of the camera sensor. Small deviations may occur due to lens misalignment, which is corrected during calibration. Distortion coefficients correct for the radial and tangential distortions caused by the fisheye lens. Radial distortion causes straight lines to appear curved, especially near the edges of the image. This effect is particularly strong in wide-angle lenses and must be compensated for to ensure accurate geometric transformations. Tangential distortion arises from minor misalignments between the camera lens and sensor, leading to slight perspective shifts. While less significant than radial distortion, it is By estimating these parameters, the calibration step enables the undistortion of video frames, ensuring that objects maintain their correct proportions and real-world distances

can be accurately measured.

3.5.2 Chessboard Calibration Process

To estimate the intrinsic parameters and distortion coefficients, a standard chessboard pattern calibration is performed [NK07]. This process involves capturing multiple images of a checkerboard pattern at various angles and distances to provide sufficient feature correspondences across the entire image plane. The calibration follows these steps:

1. Chessboard image capture: A set of 27 images of a checkerboard pattern is recorded, ensuring coverage across different regions of the frame, including the center, edges, and corners.
2. Feature detection: OpenCV's corner detection algorithm identifies the intersection points of the checkerboard squares.
3. Fisheye model fitting: The detected corner points are used to compute the intrinsic camera matrix $\mathbf{K}$ and distortion coefficients $\mathbf{D}$, optimizing for minimal reprojection error.
4. Undistortion validation: The calibration results are applied to an image, and straight lines are visually inspected to confirm the removal of barrel distortion.

3.5.3 Checkerboard Image Requirements

For accurate calibration, the checkerboard images must meet specific conditions. The pattern should be captured from multiple perspectives, including different tilts and rotations, to ensure robust feature detection. It should also appear at different locations in the frame to model lens distortion across the entire field of view. Excessive tilting should be avoided to prevent extreme perspective distortions. Additionally, shadows and reflections should be minimized to ensure reliable corner detection, and images must be high-resolution and in sharp focus, as blurry or low-quality images should be discarded to maintain precision in feature detection. The checkerboard should have a known and consistent square size, such as a 9×6 grid, with precisely measured dimensions for accurate scaling. By following this calibration process, the intrinsic parameters and distortion coefficients are accurately estimated, enabling precise undistortion of recorded frames. This ensures that all subsequent transformations, including homography mapping and real-world coordinate estimation, remain consistent and reliable.

(a) Original distorted image (b) Undistorted image after calibration

Figure 3.4: Comparison of distorted and undistorted images

The fisheye lens of the GoPro Hero 5 Session introduces significant barrel distortion, which causes straight lines, such as road markings and buildings, to appear curved outward in the recorded images. This effect is particularly pronounced at the edges of the frame, as shown in Figure 3.4a. After applying lens calibration and distortion correction, the undistorted image (Figure 3.4b) shows a rectified perspective where straight lines retain their actual shape.

3.6 Summary

The GoPro Hero 5 Session's compact design and wide-angle lens provide a flexible setup for capturing vehicle motion over a broad field of view. However, the significant barrel distortion introduced by its fisheye lens necessitates precise calibration to ensure accurate coordinate mapping. By positioning the camera on a tripod at a height of 3 meters and aligning it perpendicular to the road, a stable and well-framed recording was achieved. To enable reliable geometric transformations, key measurements were obtained using a measuring tape and LiDAR. A fisheye-specific calibration process was performed using a chessboard pattern to derive intrinsic parameters and distortion coefficients, correcting lens distortions for accurate object detection and tracking. In addition to the camera setup and calibration, an experiment was conducted to evaluate the system's performance. Two vehicles of different sizes were driven multiple times at controlled speeds, allowing a direct comparison between the system's estimated speed and dimensions and their real-world values. This structured approach ensures that the results obtained in later chapters are grounded in measurable, real-world data. With the camera's placement, calibration, and validation methodology established, the following chapters introduce the mathematical models and software framework that leverage these parameters for vehicle speed and size estimation.

4 Mathematical Foundations

Accurate real-world measurements of vehicle dimensions and speeds require careful geometric modeling. This chapter introduces the fundamental equations and transformations that map two-dimensional bounding-box detections to three-dimensional road-plane coordinates, culminating in a formula for vehicle width and length estimation. Additionally, a speed calculation method based on frame-to-frame displacement is described.

4.1 Camera Calibration and Undistortion

The fisheye nature of the GoPro Hero 5 Session lens necessitates calibration to compensate for radial and tangential distortions. Following the procedure discussed in Subsection 3.5.2, an intrinsic matrix $\mathbf{K}$ and distortion coefficients $\mathbf{D}$ are obtained, as shown in Equation 4.1:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix}. \tag{4.1}$$

In the matrix the coefficients $\mathbf{K}$ f_{x} and f_{y} are focal lengths in pixels along x and y axes, c_{x} and c_{y}. In the matrix $\mathbf{B}$ the coefficients k_1, k_2, k_3, k_4 are radial distortion coefficient, which correct for barrel distortion. All raw video frames are undistorted prior to further processing, ensuring that geometric assumptions remain valid when computing bounding-box widths and angles.

4.2 Homography and Ground-Plane Mapping

To map image coordinates (u, v) to world-plane coordinates (X, Y), we estimate a homography matrix $\mathbf{H}$. Given a set of at least four corresponding points in both the undistorted image and world coordinates, we solve for $\mathbf{H}$ in:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{4.2}$$

where $\mathbf{H}$ is a 3×3 transformation matrix:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{4.3}$$

A minimum of four reference points with known real-world positions are needed to solve for $\mathbf{H}$, as each point correspondence provides two linear equations. Since $\mathbf{H}$ has eight degrees of freedom and is determined up to a scale, four points provide sufficient constraints for computing the transformation [Ope24]. As described in Subsection 3.3.1, these points should be measured with high accuracy, e. g. using LiDAR or manual calibration. This procedure is performed once during the preparation stage.

4.2.1 Computation of the Homography Matrix

The homography matrix $\mathbf{H}$ is computed by solving a system of linear equations based on point correspondences. Given a set of $n \geq 4$ correspondences $(u_i, v_i) \leftrightarrow (X_i, Y_i)$, we formulate the linear system shown in Equation 4.4:

$$\mathbf{A}\mathbf{h} = 0, \tag{4.4}$$

where $\mathbf{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^\top$ is the vectorized form of $\mathbf{H}$. The constraint matrix $\mathbf{A}$, constructed from the known correspondences, is given in Equation 4.5:

$$\mathbf{A} = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -X_1u_1 & -X_1v_1 & -X_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -Y_1u_1 & -Y_1v_1 & -Y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n & v_n & 1 & 0 & 0 & 0 & -X_nu_n & -X_nv_n & -X_n \\ 0 & 0 & 0 & u_n & v_n & 1 & -Y_nu_n & -Y_nv_n & -Y_n \end{bmatrix}. \tag{4.5}$$

Since this system is overdetermined, we solve for $\mathbf{H}$ using Singular Value Decomposition (SVD). The decomposition is expressed in Equation 4.6, where the last column of $\mathbf{V}$ provides the solution:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top} \tag{4.6}$$

In implementation, $\mathbf{H}$ is estimated using OpenCV's `cv2.findHomography()` function [Ope24], which supports both the least squares (LS) method and RANdom SAmple Consensus (RANSAC) for robust estimation. The least squares method computes $\mathbf{H}$ by minimizing algebraic error via SVD but is sensitive to outliers. To improve robustness, RANSAC selects subsets of correspondences, iteratively refines $\mathbf{H}$, and rejects mismatched points based on a reprojection error threshold.

4.2.2 Application of Homography in Real-Time Processing

During the system's operation, after each frame is undistorted, a homography $\mathbf{H}$ is used to map 2D pixel coordinates $\big(u, v\big)$ to world-plane coordinates $\big(X, Y\big)$ in meters, as shown in Equation 4.7:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{4.7}$$

Since the homography transformation operates in homogeneous coordinates, the resulting vector must be normalized. As shown in Equation 4.8, the output from the matrix multiplication is:

$$\begin{bmatrix} X' \\ Y' \\ w \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{4.8}$$

To obtain the final world coordinates (X, Y), the homogeneous scaling factor w must be applied, as described in Equation 4.9:

$$X = \frac{X'}{w}, \quad Y = \frac{Y'}{w}. \tag{4.9}$$

In real-time tracking, homography is applied frame-by-frame to map detected objects from image coordinates to the world plane. The bottom-center of each detected bounding box, (u, v), is transformed using $\mathbf{H}$ to obtain the real-world coordinates (X, Y)) of the object.

4.2.3 Practical Considerations

The accuracy of the transformation depends on the planarity assumption; if the ground is uneven, a single homography may not be sufficient. Errors in homography estimation, caused by imprecise calibration points or imperfect undistortion, can lead to inaccuracies in world coordinate mapping. To improve robustness against noisy correspondences, outlier rejection methods such as RANSAC can be applied.

4.3 Vehicle Size Estimation

The core idea of size estimation is based on determining the proportion of a vehicle's length l and width m relative to the bounding box width S using trigonometric functions, as described in Subsection 4.3.3. This approach relies on two key assumptions. First, the vehicle travels along a straight road whose centerline is perpendicular to the camera's optical axis. Second, vehicles remain on the ground plane, meaning elevation changes are negligible. To estimate both l and m, we analyze the vehicle's motion across multiple frames. If the car is observed in at least two different positions, its size can be determined using two independent width measurements. Given more than two observations, a least squares approach can be applied for robustness. To perform this estimation, we first need to determine the real bounding box width in meters S_{real} and the angle α of the vehicle relative to the camera. These values allow us to establish the relationship between the vehicle's projected width in the image and its actual dimensions.

4.3.1 Determining the Vehicle Orientation Angle

The angle α represents the orientation of the vehicle relative to the camera's optical axis. It is defined as the angle between the line connecting the vehicle's bottom-center point to the camera and the perpendicular line from the camera to the road. Figure 4.1 illustrates the geometric relationship between the camera, the road, and the detected vehicle. The camera is placed at a fixed position perpendicular to the road, and α is measured as the deviation of the detected vehicle from this perpendicular axis.

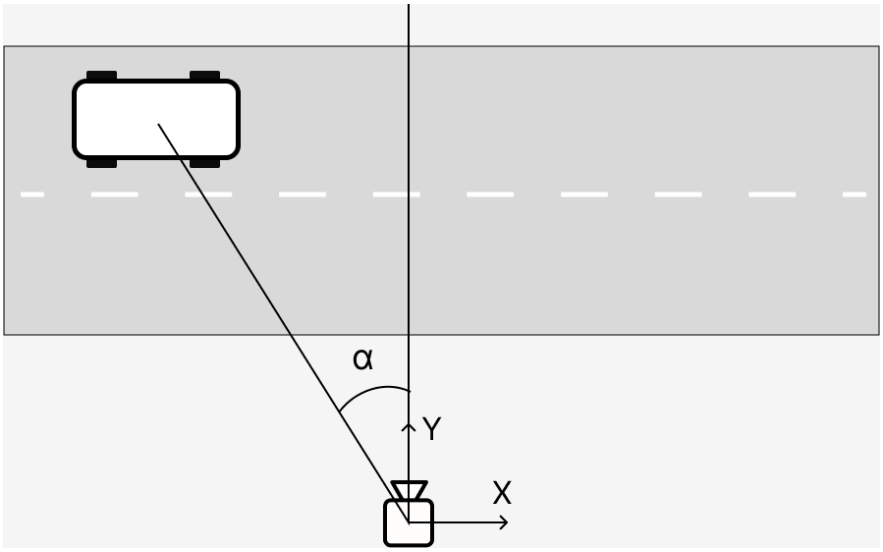

Figure 4.1: Illustration of the vehicle's orientation angle α.

To compute α, we use the transformed real-world coordinates obtained via homography (Section 4.2). Let the vehicle's bottom-center coordinate at time t be denoted as (X_t, Y_t), and let the camera's horizontal coordinate be fixed at X_{cam}. Then, as shown in Equation 4.10, α is given by:

$$\alpha = \arctan\left(\frac{Y_t - Y_{\text{cam}}}{X_t - X_{\text{cam}}}\right). \tag{4.10}$$

Here, (X_t, Y_t) represents the transformed coordinate of the detected vehicle, while X_{cam} denotes the fixed camera position along the horizontal axis. The term Y_{cam} corresponds to the camera's reference coordinate on the road plane, which is typically set to zero. This angle is crucial for accurate vehicle size estimation since the observed bounding box width depends on the vehicle's orientation relative to the camera. By knowing α, we can correctly interpret the bounding box dimensions and estimate the vehicle's true width and length. Initially, an alternative approach for estimating α more accurately involved tracking the vehicle's wheel contact points using YOLOv8 Pose. By constructing a vector between one front and one rear wheel from one side, the orientation of the vehicle relative to the camera could be directly computed. This method would be particularly useful during lane changes, where the car's orientation is not parallel to the road. The orientation angle α was derived from the angle of this wheel vector with respect to the camera's coordinate system, as shown in Equation 4.11:

$$\alpha = \arctan\left(\frac{Y_{\text{rear}} - Y_{\text{front}}}{X_{\text{rear}} - X_{\text{front}}}\right). \tag{4.11}$$

where $(X_{\text{front}}, Y_{\text{front}})$ and $(X_{\text{rear}}, Y_{\text{rear}})$ are the transformed real-world coordinates of the front and rear wheel contact points, respectively. However, this approach presented challenges in terms of robustly detecting wheel contact points under varying lighting conditions, occlusions, and different vehicle types. As a result, a homography-based transformation method was adopted, providing a more stable and consistent estimation of α.

4.3.2 Determining the Real Bounding Box Width

The bounding box width detected in the image space, denoted as S, does not directly correspond to the real-world width of the vehicle due to perspective distortion. To obtain the actual projected width of the vehicle in real-world coordinates, we need to map the bounding box dimensions from image space to the ground plane using the homography transformation. The real-world bounding box width S_{real} is computed by projecting the left and right edges of the bounding box from image coordinates $(u_{\text{left}}, v_{\text{left}})$ and $(u_{\text{right}}, v_{\text{right}})$ onto the ground plane, and then calculating the distance between these two points on the plane. As shown in Equation 4.12, this distance is given by:

$$S_{\text{real}} = \sqrt{(X_{\text{right}} - X_{\text{left}})^2 + (Y_{\text{right}} - Y_{\text{left}})^2}. \tag{4.12}$$

This transformation ensures that the vehicle's detected width accounts for perspective distortions and is expressed in the same real-world coordinate system as the camera calibration and homography mapping.

4.3.3 Calculation Process

By determining S_{real} for different frames, we can incorporate multiple observations of the vehicle's dimensions to improve the robustness of the size estimation process.

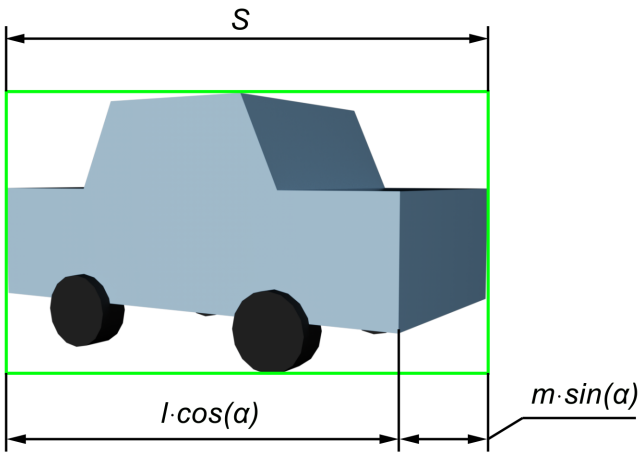

Figure 4.2: Illustration of vehicle size estimation principle.

As illustrated in Figure 4.2, at different orientations α, the projected bounding-box width $S_{\text{real}}(\alpha)$ satisfies the relationship given in Equation 4.13:

$$S_{\text{real}}(\alpha) = l\cos(\alpha) + m\sin(\alpha). \tag{4.13}$$

Given measurements $S_{\text{real}}(\alpha_i)$ from N different frames, we solve the system of equations presented in Equation 4.14:

$$\begin{bmatrix} \cos(\alpha_1) & \sin(\alpha_1) \\ \cos(\alpha_2) & \sin(\alpha_2) \\ \vdots & \vdots \\ \cos(\alpha_N) & \sin(\alpha_N) \end{bmatrix} \begin{bmatrix} l \\ m \end{bmatrix} = \begin{bmatrix} S_{\text{real}}(\alpha_1) \\ S_{\text{real}}(\alpha_2) \\ \vdots \\ S_{\text{real}}(\alpha_N) \end{bmatrix}. \tag{4.14}$$

For $N = 2$, the system is directly solvable. If more observations are available, a least squares solution minimizes noise and improves accuracy. This approach ensures

4.3.4 Choosing Measurement Points

In practice, if the camera calibration and object recognition are sufficiently precise, the use of only two frames is adequate for vehicle size estimation. However, to enhance measurement accuracy, the difference in S_{real} between the selected frames should be as large as possible. The following considerations guide the selection of optimal frames:

1. When the vehicle is positioned directly in front of the camera ($\alpha = 0$), one of the terms in Equation (4.13) reduces to zero, leaving:

$$S_{\text{real}}(\alpha) = l, \tag{4.15}$$

from which the vehicle length l can be directly determined.

2. To maximize the difference in S_{real} between two frames, the point of maximum variation in S_{real} during the vehicle's passage in front of the camera should be identified. This requires taking the derivative of Equation (4.13), resulting in Equation (4.16):

$$\frac{dS_{\text{real}}(\alpha)}{d\alpha} = -l\sin(\alpha) + m\cos(\alpha). \tag{4.16}$$

Since Equation (4.16) depends on l and m, which vary for different vehicle models, determining a universal maximum is infeasible. Instead, an approximation can be obtained by using statistically averaged vehicle dimensions [Nim24] and solving for α when Equation (4.16) equals zero:

$$-4.399\sin(\alpha) + 1.821\cos(\alpha) = 0. \tag{4.17}$$

Solving for α, as shown in Equation (4.18):

$$\alpha = \tan^{-1}\left(\frac{1821}{4399}\right) \approx 0.392 \text{ rad} \approx 22.49°. \tag{4.18}$$

Selecting the second frame at this angle ensures a sufficient difference in bounding box widths S_{real} for accurate size estimation in most vehicles.

4.4 Speed Estimation

To estimate vehicle speed, the position of a reference point — typically the lower-center of the bounding box — is tracked across consecutive frames. Let (X_t, Y_t) be the real-world coordinates of the vehicle at frame t, and let FPS denote the frame rate. The displacement Δd between consecutive frames is given by Equation (4.19):

$$\Delta d = \sqrt{\left(X_{t+1} - X_t\right)^2 + \left(Y_{t+1} - Y_t\right)^2}. \tag{4.19}$$

Since the time interval between frames is $\Delta t = 1/FPS$, the instantaneous speed in meters per second is computed using Equation (4.20):

$$v_{\mathrm{m/s}} = \frac{\Delta d}{\Delta t}. \tag{4.20}$$

To express the speed in kilometers per hour, the conversion factor 3.6 is applied, as shown in Equation (4.21):

$$v_{\mathrm{km/h}} = v_{\mathrm{m/s}} \times 3.6. \tag{4.21}$$

4.4.1 Smoothing and Multi-Frame Speed Estimation

While the above formulation provides an instantaneous speed estimate per frame, minor fluctuations in bounding-box detection can introduce noise. To mitigate this, speed calculations are smoothed using a buffer-based averaging method. Each tracked vehicle is assigned a queue storing its most recent positions and frame indices. Instead of relying solely on two consecutive frames, the velocity is estimated across multiple frames to reduce errors. Given a tracking history of N frames, the speed is computed by averaging pairwise displacement estimates, as described in Equation (4.22):

$$v_{\mathrm{km/h}} = \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{\sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2}}{(t_{i+1} - t_i)} \times 3.6. \tag{4.22}$$

This approach improves robustness, particularly when small variations in bounding-box detection introduce noise. A larger buffer size yields smoother speed estimates but may introduce a slight delay in responsiveness.

5 Software Architecture

A primary objective of this thesis is to develop a software pipeline that processes raw video input and outputs the estimated speed and dimensions of passing vehicles. This chapter details the software architecture, bridging the theoretical concepts from Chapter 4 with the hardware considerations in Chapter 3. The implementation consists of distinct modules, each handling a specific stage of the workflow.

5.1 Overview

The processing pipeline consists of the following key stages:

1. Preparation stage: This stage is conducted once before the system begins operation.
    a) Camera calibration: Determines the intrinsic parameters and distortion coefficients using a checkerboard pattern.
    b) Homography computation: Computes the homography matrix used for transforming pixel coordinates into real-world coordinates.
2. Preprocessing: Applies lens undistortion to each incoming frame to correct geometric distortions.
3. Object detection and tracking: Utilizes YOLOv8 to detect vehicles and assign persistent tracking IDs across frames.
4. Coordinate transformation: Maps bounding-box positions from pixel coordinates to real-world coordinates using the precomputed homography matrix.
5. Speed estimation: Computes vehicle speeds based on positional changes across consecutive frames.
6. Data export and post-processing: Logs vehicle tracking data to CSV files for further analysis.
7. Size estimation: Estimates the vehicle's length and width using the collected

tracking data.

The preparation stage is performed once before the system is operational. Object detection, tracking, and speed estimation may be executed in real-time, depending on hardware capabilities. Size estimation is conducted after the preceding stages have completed; however, it can be adapted to run concurrently for improved efficiency.

5.2 Preparation Stage

The preparation stage involves calibrating the camera and computing the homography matrix, both of which are essential for accurate coordinate transformation.

5.2.1 Camera Calibration

Camera calibration is an essential step to correct lens distortions and obtain accurate intrinsic parameters. The calibration procedure follows a structured workflow, as outlined in Algorithm 1, where checkerboard images are processed to detect corners, refine their positions, and compute the intrinsic parameters. The final calibration results, including the root mean square (RMS) error, are evaluated to ensure the accuracy of the estimated camera parameters.

Algorithm 1 Camera Calibration

```
Start Camera Calibration module
Load checkerboard images from the dataset
Initialize lists for object points and image points
while Calibration process is running do
    for each image in dataset do
        Convert image to grayscale
        Detect checkerboard corners using cv2.findChessboardCorners()
        if corners are found then
            Refine corner positions using cv2.cornerSubPix()
            Store object points and corresponding image points
        end if
        Compute camera matrix K and distortion coefficients D using OpenCV's fisheye calibration
        Calculate RMS error
        Save calibration results to file

        Shutdown Calibration module
```

5.2.2 Homography Computation

The homography computation follows a structured workflow, as outlined in Algorithm 2, where user-selected pixel coordinates are paired with their real-world counterparts to derive the transformation matrix.

Algorithm 2 Homography Computation

1: Load the reference image and apply the undistortion function
2: Display the undistorted image
3: Initialize lists for pixel coordinates and corresponding real-world coordinates
4: Prompt the user to select at least four reference points in the image
5: **for** each selected point **do**
6: Store the pixel coordinates (u, v)
7: Prompt the user to enter the corresponding real-world coordinates (X, Y)
8: Store the entered real-world coordinates
9: **end for**
10: Compute the homography matrix **H** using OpenCV's `cv2.findHomography()` function
11: Save the computed homography matrix to a file
12: Display visual confirmation of the transformation =0

5.3 Preprocessing

Once the camera is calibrated, each video frame undergoes preprocessing to correct geometric distortions using the previously computed intrinsic parameters. Additionally, the frame is resized to match the resolution required by the object recognition model. The preprocessing steps are outlined in Algorithm 3.

Algorithm 3 Frame Preprocessing

1: Load intrinsic camera parameters **K** and distortion coefficients **D** from the calibration file
2: Load the input frame
3: Apply lens undistortion using OpenCV's `cv2.remap()` with **K** and **D**
4: Resize the undistorted frame to match the input resolution required by the object detection model
5: Return the preprocessed image

5.4 Object Detection and Tracking

Accurate vehicle detection and tracking form the foundation for all subsequent processing steps, including coordinate transformation, speed estimation, and size estimation. The goal of this module is to identify vehicles in each frame, assign unique tracking IDs, and ensure that the same vehicle is consistently tracked across multiple frames. Each detected vehicle is represented by a bounding box, defined by the parameters x, y, w, h, where (x, y) represents the center of the bounding box in pixel coordinates, and w and h denote its width and height. The detection and tracking procedure is detailed in Algorithm 4.

Algorithm 4 Object Detection and Tracking

```
Load YOLOv8 model with pre-trained weights
Load camera calibration parameters K and D for undistortion
Initialize video stream and read the first frame
while video stream is active do
    Read the next frame from the video
    Apply frame preprocessing using the preprocessing module
    Pass the preprocessed frame to the YOLO model for object detection
    for each detected object do
        Extract bounding box coordinates and object class
        Assign a unique track ID using the tracking module
        Convert bounding box coordinates to real-world coordinates using the coordinate transformation module
        if previous frames exist for this track ID then
            Compute vehicle speed using the speed estimation module
        end if
        Store detection results including frame number, bounding box, real-world coordinates, speed, and track ID
    end for
    Annotate the frame with detection results, track IDs, and estimated speed
    Store annotated frame for visualization
end while
Save detection and tracking results to structured CSV files for further analysis
```

5.5 Coordinate Transformation

Object detection in YOLOv8 provides bounding-box coordinates in pixel space, which do not directly correspond to real-world locations. To accurately estimate a vehicle's position and dimensions, these coordinates must be transformed into a

real-world coordinate system using a homography transformation. In addition to mapping the bounding box's bottom-center, we also compute its real-world width S_{real} by transforming the bottom-left and bottom-right corners and calculating the distance between them. This process is described in Algorithm 5.

Algorithm 5 Coordinate Transformation

Start Coordinate Transformation module
Load the precomputed homography matrix **H** from the calibration file
while new detection data is available **do**
    for each detected object **do**
        Retrieve bounding-box parameters from detection results
        Compute the bottom-center pixel coordinates (u, v)
        Convert (u, v) to homogeneous coordinates and apply homography transformation
        Normalize the result to obtain real-world coordinates (X, Y)
        Compute bottom-left and bottom-right pixel coordinates
        Apply homography transformation to both points
        Compute the real-world bounding box width S_{real}
        Store transformed real-world coordinates (X, Y) and width W_{real} for further processing
    end for
end while
Shutdown Coordinate Transformation module

5.6 Speed Estimation

The speed computation follows a two-step process. First, the instantaneous speed is calculated by computing the displacement Δd between two consecutive frames using the Euclidean distance in the real-world coordinate system. Second, speed smoothing is applied by storing speed values over the last five frames in a buffer, with the final speed determined as the average of these values. The detailed speed estimation algorithm is outlined in Algorithm 6.

Algorithm 6 Speed Estimation

```
Start Speed Estimation module
Initialize a tracking dictionary for storing previous positions and frame indices
Set buffer size for averaging speed calculations
while new frame data is available do
    for each tracked object do
        Retrieve real-world coordinates (X_t, Y_t) and frame index t
        if previous position exists for this track ID then
            Retrieve previous coordinates (X_{t-1}, Y_{t-1}) and frame index t − 1
            Compute displacement between the positions on two frames
            Compute time difference between two frames based on FPS
            Compute instantaneous speed based on displacement and time difference
            Store instantaneous speed in a circular buffer for smoothing
        else
            Set speed to zero (first frame for this track ID)
        end if
        Append the new position and frame index to the tracking dictionary
    end for
end while
Compute final smoothed speed estimate as the average of stored values
Store computed speeds for further processing
Shutdown Speed Estimation module
```

5.7 Data Export and Post-Processing

Since the tracking data exported by the main processing pipeline contains information for all detected vehicles, a dedicated module is used to extract the data for a specific car. This module enables further analysis by filtering the relevant tracking records, computing real-world coordinates, and removing erroneous data. The processed results are exported as a CSV file, which includes the following columns: frame number, car ID, real-world X and Y coordinates of a bounding box, pixel width, and real width of a bounding box. The procedure for data export and post-processing is outlined in Algorithm 7.

Algorithm 7 Data Export and Processing for a Selected Car

1: Load raw tracking data from CSV file
2: Prompt the user to enter the target car ID
3: Filter tracking records to include only data for the selected car
4: Retrieve bounding-box coordinates, pixel width, and real width of a bounding box
5: Remove outlier data points based on statistical thresholds
6: **if** a reduced frame count is requested **then**
7: Select a subset of evenly distributed frames
8: **end if**
9: Save the final processed data to a CSV file

5.8 Size Estimation

To estimate the dimensions of a vehicle, two different calculation methods were developed. The least squares approach processes multiple observations of a vehicle's bounding-box width at different angles to minimize error and obtain a statistically robust estimate. The direct two-measurement approach uses only two selected frames where the vehicle is observed at different orientations and solves a system of equations to estimate the vehicle's width and length. While the least squares approach could offer more precision in cases where many observations are available, in this specific scenario, it does not provide a significant advantage. Instead, the two-measurement approach is used due to its simplicity, ease of implementation, and efficiency. Both approaches are based on the following system of equations, given in Equations (5.1) and (5.2):

$$S_{\text{real},1} = \cos(\alpha_1) + m\sin(\alpha_1), \tag{5.1}$$

$$S_{\text{real},2} = \cos(\alpha_2) + m\sin(\alpha_2), \tag{5.2}$$

where $S_{\text{real},1}$ and $S_{\text{real},2}$ are the real-world bounding-box widths observed in two different frames, and α_1 and α_2 are the angles between the vehicle and the camera in those frames. The unknowns and m correspond to the vehicle's length and width. To solve for and m in the direct two-measurement approach, we rewrite Equations (5.1) and (5.2) in matrix form as Equation (5.3):

$$\begin{bmatrix} \cos(\alpha_1) & \sin(\alpha_1) \\ \cos(\alpha_2) & \sin(\alpha_2) \end{bmatrix} \begin{bmatrix} \\ m \end{bmatrix} = \begin{bmatrix} S_{\text{real},1} \\ S_{\text{real},2} \end{bmatrix}. \tag{5.3}$$

This system is then solved using a determinant-based approach, where the denominator, ensuring solvability, is defined in Equation (5.4):

$$\Delta = \sin(\alpha_2)\cos(\alpha_1) - \sin(\alpha_1)\cos(\alpha_2). \tag{5.4}$$

Using Equation (5.4), the vehicle width m and length are computed as follows:

$$m = \frac{S_{\text{real},2}\cos(\alpha_1) - S_{\text{real},1}\cos(\alpha_2)}{\Delta}, \tag{5.5}$$

$$l = \frac{S_{\text{real},1}\sin(\alpha_2) - S_{\text{real},2}\sin(\alpha_1)}{\Delta}. \tag{5.6}$$

where Δ ensures that the equations remain solvable. If Δ is close to zero, the selected frames are too similar in orientation, and a different pair of frames should be chosen. The estimated values of l and m are then refined using empirical scaling factors to improve real-world accuracy. The step-by-step procedure for the size estimation process is outlined in Algorithm 8.

Algorithm 8 Vehicle Size Estimation (Two-Point Model)

1: Start Size Estimation module
2: Load transformed tracking data from CSV file
3: Select two distinct frames where the vehicle is observed at different angles
4: Extract bounding box widths $S_{\text{real},1}$ and $S_{\text{real},2}$
5: Compute angles α_1 and α_2 based on vehicle position relative to the camera
6: Compute the denominator (Δ)
7: Solve for vehicle length l and width m:

$$m = \frac{S_{\text{real},2}\cos(\alpha_1) - S_{\text{real},1}\cos(\alpha_2)}{\Delta}$$

8: Optionally apply empirical scaling factors for accuracy
9: Store estimated vehicle dimensions for further analysis
10: Shutdown Size Estimation module

5.9 Summary

This chapter presented the software architecture, detailing the key stages from preparation to speed and size estimation. The pipeline integrates preprocessing, object detection, coordinate transformation, and analysis modules, ensuring

a structured approach to vehicle tracking. The following chapter will present experimental results, validating the effectiveness of the developed system.

6 Results and Evaluation

This chapter presents the experimental validation of the proposed single-camera approach. After a brief overview of the testing procedure, the accuracy of both speed and dimension estimations is discussed, followed by an error analysis highlighting system limitations and potential improvements.

6.1 Experimental Procedure

To evaluate the accuracy and reliability of the proposed system, a series of controlled experiments were conducted. The setup aimed to replicate real-world conditions while maintaining precise ground-truth measurements for validation. Vehicles with known dimensions were recorded at different speeds, and the collected data was processed to assess the effectiveness of the detection, tracking, and estimation methods. The following subsections describe the test scenarios, data collection methodology, and evaluation criteria used in the experiments.

6.1.1 Test Scenarios

Validation was performed using multiple recording sessions on a straight road segment. Two cars, each with known dimensions, passed in front of the camera at preset speeds of 20km/h, 30km/h, 40km/h, and 50km/h. The GoPro Hero 5 Session (Chapter 3) was installed 3 meters above ground level, perpendicular to the vehicle path.

6.1.2 Ground-Truth Data

Vehicle speeds were obtained from the dashboard speedometer providing nominal values for approximate reference. Car dimensions (length l and width m) were based on manufacturer specifications.

6.2 Speed Estimation Accuracy

6.2.1 Impact of Smoothing Window

The accuracy of speed estimation is affected by the choice of the smoothing window size, which determines how many past frames contribute to the moving average. A small window results in rapid fluctuations due to detection noise, while a larger window smooths variations but introduces latency in tracking speed changes. To evaluate the optimal smoothing parameter, the same dataset was analyzed with different buffer sizes: 1, 5, 10, and 20 frames. The corresponding speed estimation results are shown in Figure 6.1.

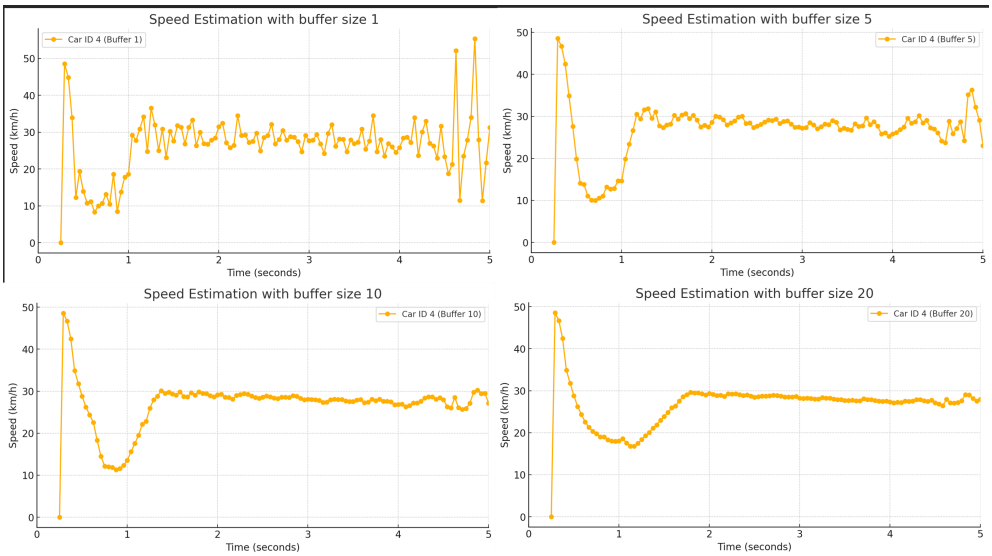

Figure 6.1: Comparison of speed estimation with different buffer sizes.

Analyzing the plots, it is evident that a buffer size of 1 results in highly erratic speed estimation due to frame-to-frame bounding box variations. Increasing the buffer to 5 frames improves stability but retains noticeable noise. A buffer of 10 frames strikes a balance between responsiveness and smoothness, effectively reducing fluctuations while maintaining real-time adaptability. However, with a buffer size of 20 frames, the speed estimation exhibits a noticeable lag before converging to the actual vehicle speed. This delay results from the longer averaging window, which smooths out short-term fluctuations but also causes a slower response to rapid speed changes. While this improves stability and reduces noise, it may hinder applications requiring real-time speed adjustments. Based on this analysis, a buffer size of 10 frames was selected as the optimal parameter, ensuring a reliable balance between noise reduction and tracking responsiveness.

6.2.2 Limitations of Speed Estimation on Image Edges

During testing, it was observed that speed estimation is significantly less reliable at the edges of the image. Several factors contribute to this issue. One factor is residual distortion; despite fisheye correction, some distortion remains, affecting object positioning near the edges. Another issue is the initialization artifact, where a car's initial speed is unknown and is set to zero when first detected, leading to incorrect values at the beginning of a track due to the smoothing window. Additionally, partial visibility affects accuracy, as a vehicle that is not yet fully in frame has a nearly fixed bounding box, impacting speed calculations. To analyze the extent of this effect, we examined the velocity as a function of position, $V(x)$. The results, shown in Figure 6.2, indicate that reliable speed estimation occurs primarily within the range $x = [-10, 10]$ meters.

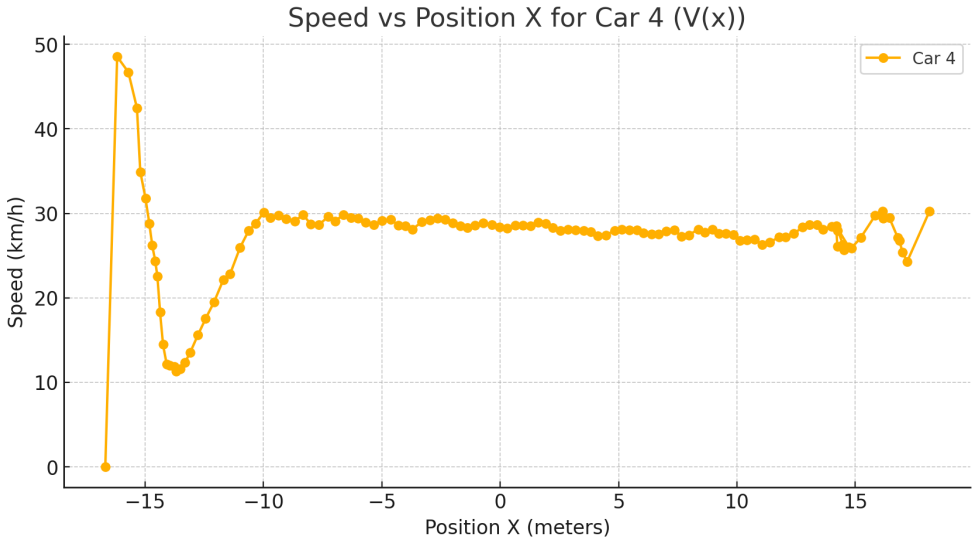

Figure 6.2: Speed vs. Position X.

Additionally, Figure 6.3 illustrates an example where a partially visible car results in an inaccurate speed estimate due to an incomplete bounding box.

Figure 6.3: Example of a partially visible car.

These observations highlight the need for additional filtering or correction techniques to improve speed estimation reliability near the image boundaries. For this reason for the result evaluation we will analyze only the results that stay within the range of $x = [-10, 10]$ meters.

6.3 Analysis of Speed Estimation Results

To evaluate the accuracy of speed estimation, we focused on the reliable range of $x = [-10, 10]$ meters, as identified in the previous section. The experiment was conducted on a regular street using only a speedometer for reference, meaning that achieving a perfectly accurate ground truth was not possible. For instance, at 20 km/h, cars had to change lanes halfway through, which visibly influenced the speed estimation graphs. Figures 6.4 and 6.5 illustrate the speed estimation at speeds of 20, 30, 40, and 50 km/h for Car 1 and Car 2 respectively.

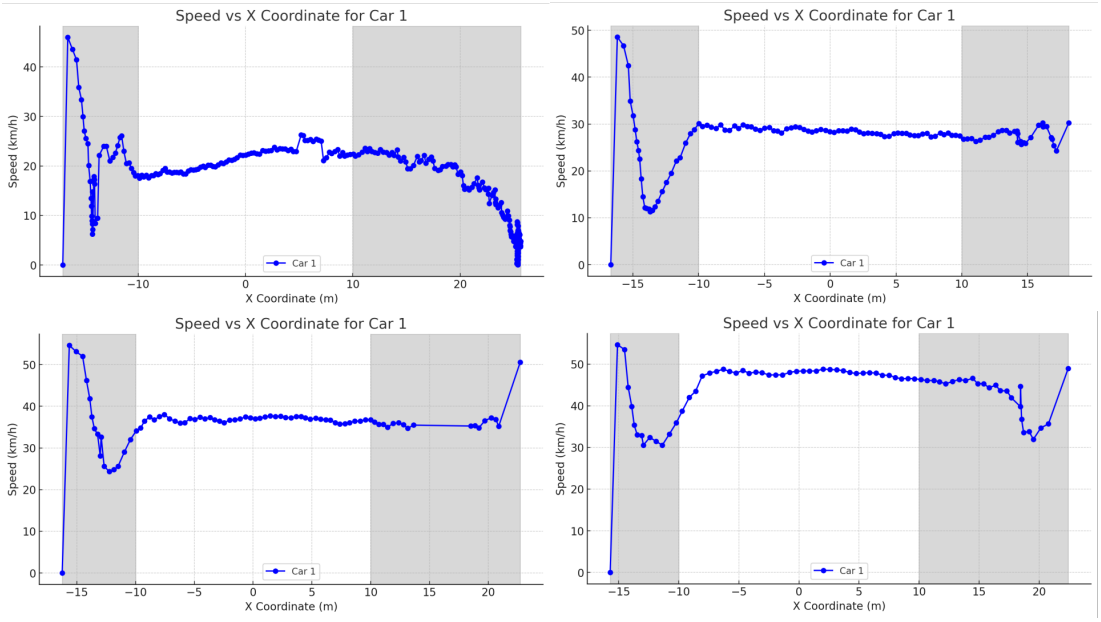

Figure 6.4: Speed estimations for Car 1.

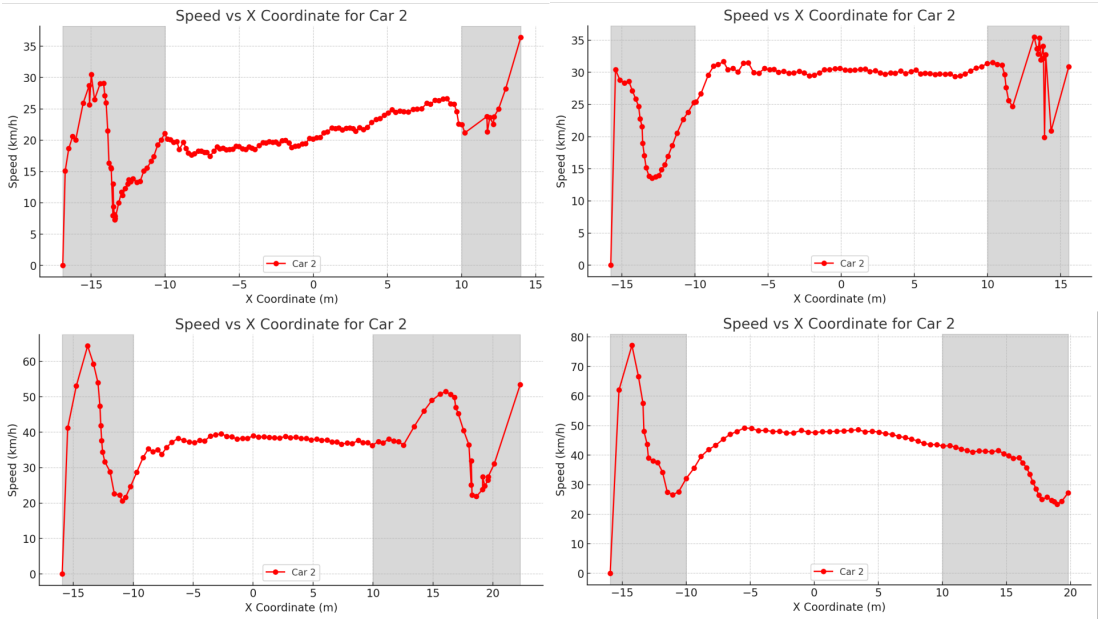

Figure 6.5: Speed estimations for Car 2.

The mean error and fluctuation for each speed scenario were calculated to quantify accuracy and consistency. The mean error represents the difference between the estimated speed and the reference speed, averaged over the reliable region. A positive value indicates overestimation, while a negative value suggests un-

derestimation. Fluctuation, defined as the standard deviation of speed within the reliable region, represents estimation stability. Lower fluctuation indicates smoother and more consistent results. Table 6.1 presents the computed mean error and fluctuation values for both test vehicles:

Table 6.1: Mean error and fluctuation in speed estimation for different test cases.

Car		**Speed (km/h)**			
		20	**30**	**40**	**50**
Car 1	Mean Error	1.2972	-1.4904	-3.1985	-2.6930
	Fluctuation	2.3423	0.7185	0.7346	1.9959
Car 2	Mean Error	1.0746	0.0765	-2.6639	-3.9654
	Fluctuation	2.6684	0.9393	1.9195	3.6780

The proposed method achieves an overall mean error of **-1.6437 km/h**, which is comparable to the 2 km/h error reported in prior single-camera methods [CD23]. However, unlike deep learning-based approaches that leverage temporal features or depth estimation, the geometric approach used here shows increased underestimation at higher speeds. While the method provides both speed and dimension estimation, further refinements in calibration and filtering could improve accuracy and robustness to match learned models. In summary, the speed estimation method demonstrates good accuracy within the defined range, with a minor tendency to underestimate higher speeds. Further improvements, such as better lane change handling and refined calibration, could enhance reliability in real-world applications.

6.4 Dimension Estimation Results

The results of dimension estimation revealed significant inconsistencies. As observed in Figure 6.6, the estimated width m varies drastically depending on the choice of the second measurement point, despite the first point being chosen where α is closest to zero. While the sign of m correctly represents the vehicle's position relative to the camera, its absolute value increases over time instead of converging to a constant, which contradicts the theoretical expectations outlined in Section 4.3.4.

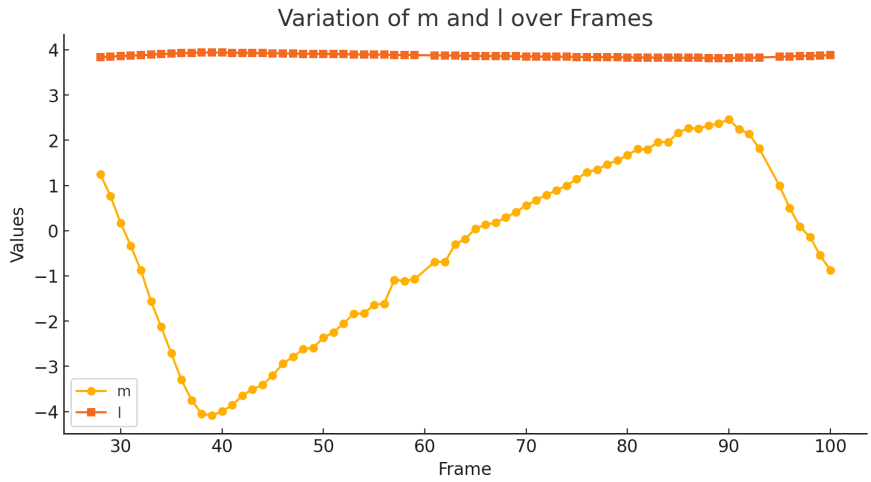

Figure 6.6: Variation of estimated width m (m) and length l (m) over frames.

6.4.1 Comparison with Ground Truth

Table 6.2 presents the numerical results, showing the estimated dimensions for two test vehicles across four different video samples with $\alpha = 45°$. The results indicate substantial deviations from the true vehicle dimensions.

Table 6.2: Comparison of True and Estimated Vehicle Dimensions Across Four Videos.

Video	**Car 1 (m)**	**Car 1 m (m)**	**Car 2 (m)**	**Car 2 m (m)**
Video 1	3.83	1.26	4.38	1.99
Video 2	3.92	1.47	4.35	2.54
Video 3	3.88	1.55	4.30	1.81
Video 4	3.83	1.55	4.33	1.84
True Values	4.25	1.76	4.71	1.82
Error (%)	-9.88	-28.41	-7.02	+9.34

6.5 Error Analysis and Discussion

Understanding the sources of error in the system is crucial for improving its accuracy and robustness. The following analysis examines the primary factors contributing to discrepancies in vehicle size and speed estimation. By identifying these issues, we can better assess the system's limitations and propose potential improvements. The discussion also highlights the challenges introduced by using a fisheye camera and its impact on coordinate transformations.

6.5.1 Sources of Estimation Errors

Several key problems contribute to the observed errors. One major issue is imperfect coordinate mapping. Ideally, the system should estimate the vehicle's length accurately by selecting the frame where the car is directly in front of the camera, using the bounding-box width as the reference car length. However, inconsistencies in homography transformation lead to erroneous measurements. Another source of error is bounding box noise, where fluctuations in bounding-box dimensions introduced by the detection algorithm affect the computed vehicle size. While averaging multiple frames can reduce this noise, it does not resolve the fundamental discrepancies in our case. These problems should, in principle, be mitigated by analyzing multiple frames. However, our results indicate that even with extensive frame analysis, the errors persist, pointing to a more fundamental issue that is discussed in Subsection 6.5.2.

6.5.2 Mismatch Between Fisheye and Pinhole Camera Models

A likely root cause of these inaccuracies is the inherent distortion of the GoPro Hero 5 Session's fisheye lens, which does not conform to the ideal pinhole camera model. The pinhole model assumes that all light rays pass through a single focal point before reaching the image plane, resulting in a linear perspective projection. This allows accurate homography transformations and coordinate mapping. However, fisheye lenses follow an entirely different projection model, where straight lines appear curved unless perfectly corrected. Despite applying undistortion techniques, residual distortions remain. The assumption of a linear transformation between image and ground-plane coordinates becomes invalid, leading to systematic errors in size estimation [Unk24].

7 Conclusion

The findings of this thesis highlight both the strengths and challenges of employing a monocular camera for vehicle speed and dimension estimation. Future advancements, including enhanced correction models and adaptive filtering techniques, could further improve the system's robustness and generalizability. The next section summarizes the key contributions of this thesis and outlines the main findings derived from experimental validation.

7.1 Summary of Contributions

This thesis explored a methodology for vehicle speed and dimension estimation using a single-camera system, integrating deep learning and geometric transformations. The main contributions of this work include the development of a monocular vehicle tracking system, where a GoPro Hero 5 Session was used as the primary vision sensor to track vehicles and estimate their physical dimensions and speed from a single viewpoint. Deep learning-based detection was implemented using YOLOv8, providing robust object detection and enabling accurate tracking of vehicles under varying conditions. A homography transformation was employed to map detected bounding box positions from image space into real-world coordinates, facilitating precise speed and size calculations. A dimension estimation algorithm was introduced, leveraging the bounding box size to infer vehicle dimensions. To validate the proposed approach, controlled experiments were conducted using vehicles of known dimensions and speeds, providing quantitative assessments of speed estimation accuracy and measurement consistency. Additionally, an error analysis was performed to examine the limitations of fish-eye lens calibration, bounding box noise, and homography mapping inaccuracies, identifying areas for further improvement.

7.2 Limitations and Future Work

Despite achieving promising results, several limitations were identified. Speed estimation was found to be less reliable near the edges of the camera frame due

to residual distortions and detection inconsistencies. The width estimation algorithm exhibited variability across different frames, indicating that the proposed method is not reliable when used with a fisheye lens. Additionally, while YOLOv8 provided high detection accuracy, further optimization is required to achieve real-time performance suitable for deployment in embedded systems. Future research directions include incorporating improved homography refinement techniques to reduce geometric mapping errors and exploring the use of a standard pinhole camera instead of a fisheye lens. Expanding testing to diverse environmental conditions, including nighttime and adverse weather scenarios, would further validate the system's robustness. Additionally, optimizing the system for real-time execution on embedded hardware would enable practical deployment in traffic monitoring applications.

7.3 Final Remarks

This study demonstrated that a single-camera setup can effectively estimate vehicle speeds and dimensions using a combination of deep learning and geometric transformations. While some challenges remain, particularly in refining size estimation, the findings provide a foundation for future advancements in cost-effective, scalable traffic monitoring solutions. With further optimizations and real-world testing, single-camera vehicle tracking systems can offer a viable alternative to multi-sensor setups, enhancing accessibility and affordability in intelligent transportation systems.

Bibliography

[CD23] Andrej Cvijetić and Slobodan Dukanović. "Deep learning-based vehicle speed estimation using the YOLO detector and 1D-CNN". In: (2023), pp. 1–4. DOI: 10.1109/IT57431.2023.10078518.

[GoP25] GoPro Support. *HERO5 Session Field of View (FOV) Information*. Accessed: February 2025. Feb. 2025. URL: https://community.gopro.com/s/article/HERO5-Session-Field-of-View-FOV-Information?language=de.

[IBM23] IBM Data and AI. *Faster R-CNN vs. YOLO vs. SSD: Object Detection Algorithms*. Accessed: February 2025. 2023. URL: https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc.

[KDA22] H. Khosravi, R. Asgarian Dehkordi, and A. Ahmadyfard. "Vehicle speed and dimensions estimation using on-road cameras by identifying popular vehicles". In: *Scientia Iranica* 29.5 (2022), pp. 2515–2525. DOI: 10.24200/sci.2020.55331.4174.

[Lab24] LabelVisor. *YOLOv8 vs. Mask R-CNN: In-Depth Analysis and Comparison*. Accessed: February 2025. 2024. URL: https://www.labelvisor.com/yolov8-vs-mask-r-cnn-in-depth-analysis-and-comparison.

[Naa+24] Arshi Naaz et al. "Detecting Car Speed using Object Detection and Depth Estimation: A Deep Learning Framework". In: *arXiv preprint* 2408.04360 (2024). DOI: 10.48550/arXiv.2408.04360.

[Nim24] NimbleFins. *Average Car Dimensions: Length, Width, and Height*. Accessed: February 2025. Apr. 2024. URL: https://www.nimblefins.co.uk/cheap-car-insurance/average-car-dimensions.

[NK07] Valsamis Ntouskos and George Karras. "Automatic calibration of digital cameras using planar chess-board patterns". In: *Optical 3-D Measurement Techniques VIII* 1 (Jan. 2007), p. 9.

[Oh+18] Jiyong Oh et al. "A Comparative Study on Camera-Radar Calibration Methods". In: *15th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (2018), pp. 1057–1062. DOI: 10.1109/ICARCV.2018.8581329.

[Ope24] OpenCV. *Homography Transformation Tutorial*. Accessed: February 2025. 2024. URL: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html.

[PJC21] Aarav Pandya, Ajit Jha, and Linga Reddy Cenkeramaddi. "A Velocity Estimation Technique for a Monocular Camera Using mmWave FMCW Radars". In: *Electronics* 10.19 (2021), p. 2397. DOI: 10.3390/electronics10192397.

[Rob24] Roboflow. *Roboflow: Streamline Computer Vision Datasets*. Accessed: February 2025. 2024. URL: https://roboflow.com.

[Sax+24] Gaurav Saxena et al. "Deep Learning–based Vehicle Counting and Speed Estimation in Surveillance Videos using Object Detection". In: *SSRN Electronic Journal* (2024). DOI: 10.2139/ssrn.4573959.

[Unk24] Authors Unknown. "A Comprehensive Overview of Fish-Eye Camera Distortion Correction Methods". In: *arXiv* (2024). Accessed: February 2025. URL: https://arxiv.org/html/2401.00442v2.

[Wu+15] Wencheng Wu et al. "Vehicle speed estimation using a monocular camera". In: 9407 (2015), p. 940704. DOI: 10.1117/12.2083394.