POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Image Analysis and Computer Vision Project

Car Space Occupancy

Author:
Salvi Niccolò Villa Alessio Zani Beatrice

Student ID:
10773726 10791818 10758494

Advisor:
Prof. Caglioti Vincenzo

Academic Year:
2024-25

Contents

1 | Introduction

1.1. Background and Motivation

Vehicle space occupancy estimation represents a fundamental challenge in modern intelligent transportation systems, autonomous vehicle navigation, and traffic monitoring applications. The ability to accurately determine the spatial footprint of moving vehicles from visual data is crucial for collision avoidance, path planning, parking assistance, and traffic flow analysis. Traditional approaches often rely on complex sensor fusion systems, expensive LiDAR technology, or require extensive computational resources that limit real-time application feasibility. The challenge becomes particularly pronounced in low-light conditions where conventional computer vision algorithms struggle to maintain accuracy. Nighttime driving scenarios, tunnel environments, and poorly illuminated areas represent critical operational contexts where robust vehicle detection and space estimation are essential for safety-critical applications. These conditions necessitate specialized approaches that can leverage available visual cues, such as vehicle lighting systems, to maintain reliable performance.

1.2. Related Work

Vehicle detection and tracking under nighttime conditions using a single calibrated camera has emerged as a challenging yet increasingly active area of research, driven by its relevance to intelligent transportation systems and autonomous navigation. Existing literature can be broadly categorized into four main areas: deep-learning-based detectors adapted to low-light environments, traditional light-based methods, monocular 3D localization techniques, and multi-frame tracking algorithms.

In the realm of deep learning, recent works have introduced architectural enhancements to object detectors specifically designed to operate in low-light scenarios. Dark-YOLO, proposed by Liu et al.[3], incorporates a trainable denoising and brightening module within the YOLO detection framework. By combining cross-scale feature pyramids and attention mechanisms, it achieves 71.3% mAP@50 on the ExDark dataset while maintaining real-time inference capability. Similarly, YOLA by Hong et al.[2] introduces a Lambertian reflectance-based feature extractor to enforce illumination invariance directly within the backbone network. This method avoids explicit image preprocessing while significantly improving recall in nighttime detection tasks. Xiao et al.[7] present LIDA-YOLO, which applies unsupervised domain adaptation from daytime to nighttime imagery. By aligning multi-scale feature distributions across lighting conditions, their model general-

izes effectively to dark environments without requiring manually labeled night-time data. Another relevant contribution is from Yang et al.[8], who propose a GAN-based style transfer pipeline that transforms daytime images into synthetic nighttime variants. By training on a mixture of real and stylized images—including synthetic scenes rendered in CARLA—the augmented dataset enhances the robustness of conventional YOLO detectors to night scenes.

Prior to the dominance of deep learning, classical vision pipelines exploited geometric and photometric properties of vehicle lighting. Xu et al.[9] propose a multi-stage pipeline that first segments salient bright regions and then verifies them using superpixel-weighted HOG descriptors classified by a support vector machine (SVM). A Kalman filter is then used for temporal association. This method proves effective in scenarios where only rear light pairs are visible. Satzoda and Trivedi[5] develop a light-pair detector based on Haar-like features and AdaBoost. Their approach uses camera calibration to reject false detections based on physical spacing constraints between detected lights and further infers approximate vehicle distance.

Monocular methods for 3D localization address the challenge of mapping image-based detections onto the real-world road plane. A common approach is inverse perspective mapping (IPM), which, given a known camera calibration, warps 2D bounding boxes into bird's-eye view projections, enabling consistent estimation of object dimensions and positions[1]. Rezaei et al.[4] further propose a zero-calibration method that leverages publicly available map data to automatically infer extrinsic parameters. This enables monocular localization and perspective correction without requiring manual setup or ground-plane constraints.

Finally, multi-frame tracking approaches have been widely adopted to refine noisy detections and ensure temporal consistency in vehicle localization. Zhang et al.[10] introduce ByteTrack, a tracker that retains low-confidence detections during data association using motion-based intersection over union (IoU), significantly reducing the rate of track fragmentation in challenging nighttime sequences. DeepSORT, proposed by Wojke et al.[6], extends classical Kalman filtering by incorporating deep appearance features, thus enabling more reliable tracking under occlusions and varying illumination. More recently, He et al.[11] propose a probabilistic clustering method that models pairs of headlights using Gaussian mixture models over time, achieving multiple object tracking accuracy (MOTA) scores exceeding 90% on nighttime highway videos.

In summary, the current state-of-the-art in nighttime vehicle detection and tracking involves an integrated pipeline consisting of low-light-adapted detectors, robust multi-object trackers, and geometric projection techniques to estimate real-world occupancy. Despite recent advances, challenges persist in generalizing across diverse lighting conditions, handling ambiguous feature visibility, and achieving reliable 3D localization in the absence of strong perspective cues.

1.3. Problem Statement

This project addresses the specific challenge of developing a vision-based system capable of analyzing videos of moving vehicles captured by a fixed camera to draw a rectangular parallelepiped bounding box around the vehicle for each frame under low-light conditions. The primary objective is not to determine the actual space occupied by the car, but rather to accurately estimate and visualize the 3D bounding volume that encompasses the entire vehicle using geometric reconstruction techniques. The system must operate with minimal prior information, requiring only the intrinsic calibration parameters of the camera (**K** matrix) and a simplified model of the observed vehicle including basic dimensional parameters such as length, width, and the spatial configuration of rear lights.

1.4. Assumptions

The system operates under the following assumptions:

- The camera is static and intrinsically calibrated (calibration matrix K is known).
- The car has a vertical symmetry plane and two symmetric rear lights that are visible in the scene.
- The road is locally planar.
- Between consecutive frames, the vehicle either translates forward or follows a motion with constant curvature.

Implementation and Resources

The implementation developed for this project closely follows the methodology illustrated in specific slides provided by the course instructor. Both the full working code and the reference slides are available in the following public GitHub repository.[1]

[1] `https://github.com/NiccoloSalvi/IACV-SpaceOccupancy`

2 | Experimental Setup

2.1. Data Collection

The experimental data for this project was collected using an **iPhone 13**, equipped with a dual-camera system (wide and ultra-wide lenses). Specifically, the main wide-angle lens was selected due to its superior optical properties and stability in low-light conditions. The video was recorded in an urban environment, capturing moving vehicles at nighttime to reflect the conditions targeted by this study.

The camera was mounted on a stable tripod at a fixed position and inclination, approximately 2.5 meters above the ground, ensuring a clear and consistent perspective of the roadway. The distance from the camera to the road was approximately 10 meters, providing sufficient perspective to capture clear geometric features on the vehicle.

The recording settings chosen were:

- Resolution: 3840×2160 pixels (4K)
- Frame rate: 30 frames per second
- Exposure: Automatic (AE lock enabled), optimized for nighttime recording
- Focus: Automatic (AF lock enabled)
- Format: HEVC (High Efficiency Video Coding)

It is important to note that if AE (auto-exposure) and AF (auto-focus) are not locked during video capture, the camera may continually adjust exposure and lens focus. Exposure variations (such as ISO or shutter speed) do not alter the camera's intrinsic matrix, which is determined by the focal length and optical center. However, autofocus can induce physical movement of the lens elements, slightly changing the effective focal length through a phenomenon known as lens breathing. This may introduce per-frame variations in the intrinsic matrix. Therefore, in applications requiring a stable calibration—such as precise vehicle localization using geometric constraints—it is strongly recommended to lock both AE and AF during video acquisition to maintain consistency.

2.2. Camera Calibration

Accurate camera calibration is crucial for the geometric computations required by this project. In order to collect as many calibration frames as possible, we recorded a video

of the checkerboard using the exact same camera settings applied during vehicle capture. From this video, 99 frames were extracted showing the checkerboard from various angles and distances, ensuring robust input for calibration. This approach helped to improve the numerical stability and accuracy of the intrinsic parameter estimation. To obtain the camera's intrinsic parameters (the K matrix), a standard calibration procedure was implemented. A calibration pattern (a planar chessboard with known dimensions: square size of 24.5mm) was used, capturing multiple images at varying angles and distances from the camera.

Calibration was initially performed using custom Python code, leveraging OpenCV functions (cv2.findChessboardCorners, cv2.calibrateCamera). To ensure accuracy and reliability, the resulting intrinsic parameters were cross-validated using the iOS Camera API, specifically leveraging metadata provided by the AVCaptureDevice API. This API reports intrinsic parameters computed internally by iOS during video recording.

The computed intrinsic matrix obtained from our custom calibration procedure was:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3.20149890e+03 & 0 & 1.93982925e+03 \\ 0 & 3.20637527e+03 & 1.06315413e+03 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Additionally, the calibration process also produced the following distortion coefficients:

$$\text{Distortion coefficients: } \begin{bmatrix} 2.43773846e-01 \\ -1.59544680e+00 \\ -1.15284213e-03 \\ 4.19886247e-04 \\ 3.56681588e+00 \end{bmatrix} \tag{2.2}$$

Moreover, the calibration process resulted in a root mean square (RMS) reprojection error of:

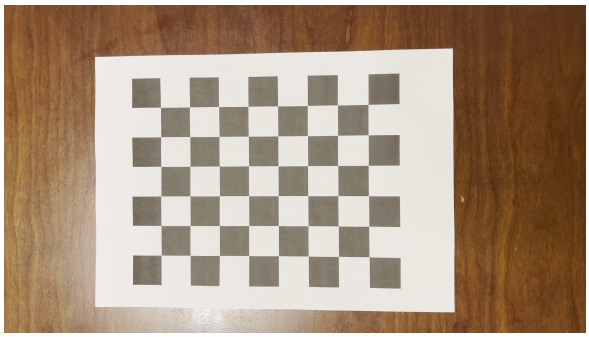
RMS Reprojection Error: 0.7705

The root mean square reprojection error of 0.7705 pixels confirms the accuracy and consistency of the intrinsic parameters obtained, particularly considering the smartphone camera context and the use of diverse viewpoints.

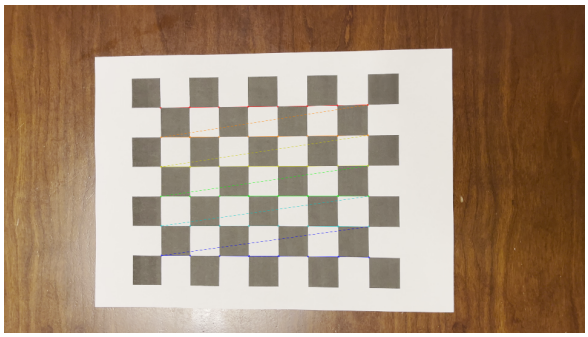
Cross-checking these parameters with those provided by the AVCaptureDevice API yielded some noticeable differences. The matrix obtained from the iOS API was:

$$K_{\text{API}} = \begin{bmatrix} 2805.4324 & 0 & 1919.5735 \\ 0 & 2805.4324 & 1077.1753 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

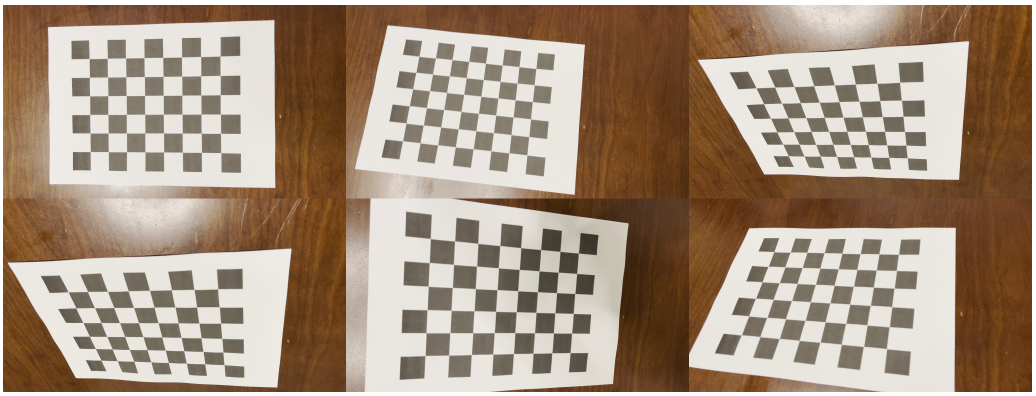
While this matrix appears significantly different in focal length values, we believe it may reflect internal lens parameterizations not directly influenced by real-world geometry or may be affected by system-level post-processing. Given that our OpenCV-based calibration was performed using real scene images and yielded consistent reprojection errors, we consider our computed matrix more reliable for geometric reasoning tasks.



(a) Checkerboard corners detected using `cv2.findChessboardCorners`.



(b) Detected corners annotated on frame.



(c) Multiple calibration views for geometric diversity.

Figure 2.1: Detected checkerboard corners (top) and selection of calibration frames (bottom).

2.3. Vehicle Model and Reference Dimensions

The vehicle used as the reference model in this project is the **Škoda Fabia**. Its rear geometry was selected due to its clear symmetry, availability of technical dimensions, and suitability for feature-based pose estimation. All geometric measurements, including the positions of rear lights, license plate corners, and mirrors, were manually extracted from the official reference image and scaled to millimeters.

The following are the main physical dimensions used:

- **Height of internal rear lights (A–B)**: 1000 mm
- **Height of external rear lights (E–F)**: 1200 mm
- **Height of license plate lower corners (C–D)**: 900 mm
- **Width of the license plate (distance C–D)**: 520 mm
- **Distance between internal rear lights (A–B)**: 860 mm
- **Distance between external rear lights (E–F)**: 1400 mm
- **Total vehicle width (O–F)**: 1732 mm
- **Wheel-to-wheel width (O–O)**: 1457 mm
- **Angle φ between the vertical and the rear surface of the vehicle**: 9.46°

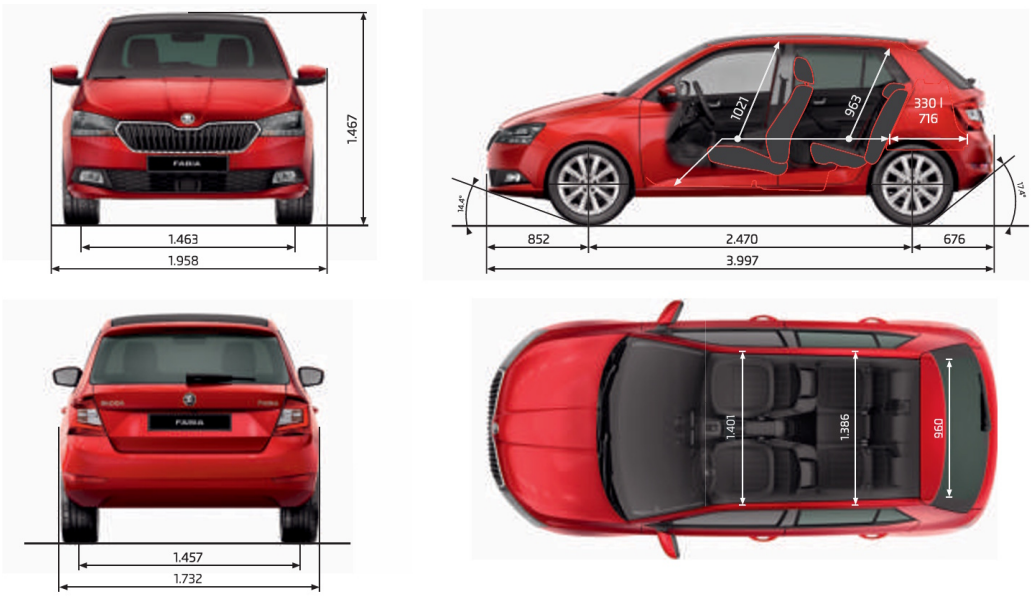


Figure 2.2: 2D CAD-style schematic of the reference vehicle (Škoda Fabia), annotated with labeled feature points and known dimensions in millimeters.

3 | Feature Extraction

This section presents a methodology to extract geometric and photometric features of a vehicle's rear side, focusing specifically on the detection of taillights, the localization of the license plate and the localization of the side mirror from image frames. These features serve as critical landmarks for estimating the rear plane and pose of the vehicle in 3D space.

3.1. Frames extraction from video

After filming the nighttime video and calibrating the camera, we moved to extract a set of evenly spaced frames from our video file for further analysis or processing. Specifically, we opened the video, computed the total number of frames, and then selected a fixed number of frames uniformly distributed across the video timeline. We then saved the frames extracted in a folder.

Figure 3.1 shows four frames extracted from the original video. The symmetric taillights and the license plate are clearly visible across all frames, and serve as key features for 3D localization and orientation estimation.

(a) Frame 0 **(b) Frame 4** **(c) Frame 8** **(d) Frame 12**

Figure 3.1: Sequence of extracted frames used for symmetric feature tracking

3.2. Taillights Extraction

The detection process begins by isolating the vehicle from the background using a convolutional neural network trained on object detection (such as YOLO). This network identifies and localizes the vehicle in each image through a bounding box. The image is then cropped to this bounding box, reducing irrelevant background content and focusing analysis on the vehicle's rear.

Once the rear of the vehicle has been isolated, we aim at locating the red taillights. Rather than operating in RGB color space, which is sensitive to illumination changes, the image is converted to HSV (Hue, Saturation, Value). This shift is non-trivial: in HSV space, hue represents color in a more stable manner, making red easier to isolate under variable lighting conditions such as shadows, glares, or overexposure.

Two hue ranges are defined—one near 0° and another near 180°—to encompass the dual ends of the red spectrum, since taillights have a stronger red in the heart of lights and a lighter red around them. These ranges are used to create binary masks that isolate red regions. Morphological operations follow, cleaning up the masks by removing small noise and enhancing continuous regions. From here, contours are detected and filtered. The obtained mask looks as follows:

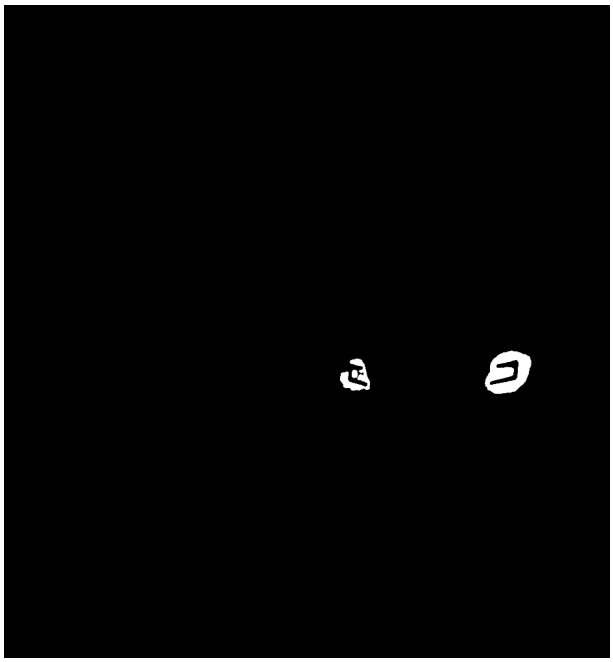


Figure 3.2: Mask found for red taillights

The assumption is that the two largest red regions within this refined mask correspond to the taillights. This is a reasonable consideration, especially in urban night driving scenarios where taillights are often the most prominent red features. The methodology then computes two points for each taillight: the center and the bottom outer edge. These points are not are going to serve later as structural anchors for orientation estimation and projection.

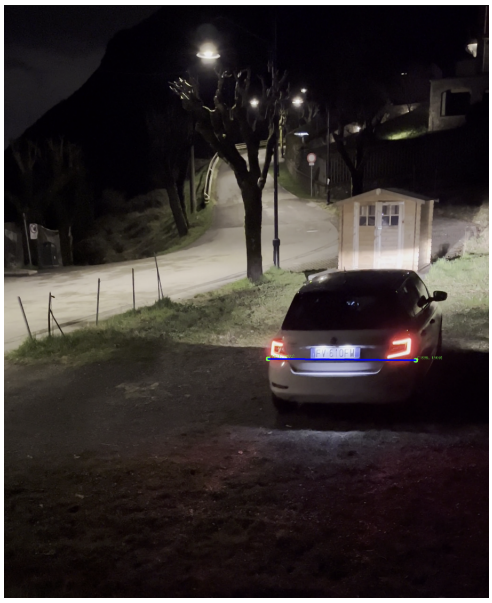


Figure 3.3: Example of taillights detection from a frame

Using these taillight detections, the orientation of the rear plane is inferred by calculating the angle of the line joining the taillights. This angle is used to rotate the image such that the taillights lie along a horizontal axis, making subsequent feature extraction more stable and geometrically consistent.

3.3. License Plate Extraction

Following rotation, the license plate is expected to lie directly below the line connecting the taillights. A region of interest (ROI) is defined between and below the taillights to constrain the search space. Within this region, segmentation is performed to isolate the license plate, using its characteristic colour properties—primarily white for the body of the plate and blue for the side strip, as found in many European plates.

Figure 3.4: Mask found for plate

Contours are then extracted and ranked based on geometric properties. Among the candidates, the one with the largest area and an appropriate aspect ratio (typically between 1.5 and 5.5) is selected as the license plate. The bounding box surrounding this contour defines the spatial extent of the plate in the image.

The final output of this process is the localization of the license plate via its bounding box and the precise positions of the taillights. These elements jointly define a consistent geometric configuration on the vehicle's rear surface and are essential for downstream tasks such as vanishing point estimation, camera resectioning, and 3D back-projection of the rear plane.

Figure 3.5: Example of licence plate detection from a frame

3.4. Side Mirror Extraction

This section outlines a procedure for detecting the side mirror of a vehicle from a single image frame. The mirror serves as a salient lateral landmark that contributes to understanding the vehicle's spatial extent and orientation in 3D space.

The process begins with isolating the vehicle from its surroundings using an object detection model based on the YOLO architecture, as in the previous section. To account for side elements like mirrors that may lie slightly outside the bounding box, the crop is expanded slightly along the horizontal axis.

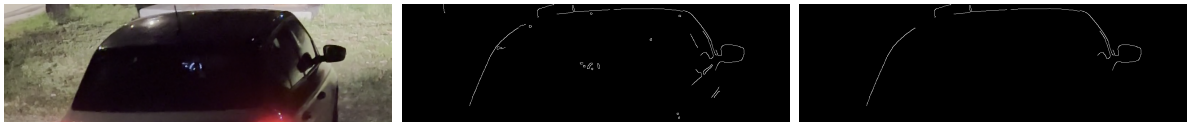
Once the vehicle region is extracted, the image is preprocessed to highlight structural edges. The cropped frame is first converted to gray-scale, then smoothed using a Gaussian blur to reduce noise. This is followed by Canny edge detection, which enhances high-contrast boundaries and delineates object contours. This step ensures that only well-defined shapes—such as the mirror's outline—are retained for further analysis.

Contours are extracted from the resulting edge map and filtered to retain only those above a certain arc length threshold. This helps eliminate small and irrelevant features while preserving prominent shapes likely to correspond to vehicle components. The filtered contours are then drawn onto a blank canvas for visual inspection and geometric reasoning.

To locate the side mirror, the algorithm searches for the contour point with the most extreme horizontal (x-axis) position. This point is selected based on the assumption that the mirror protrudes outward from the main body of the vehicle and will be positioned near the outermost edge of the cropped frame.

Once identified, the position of the mirror is adjusted to match the coordinates of the original image by reapplying the offset introduced during the initial cropping. This ensures spatial consistency with other detected features.

The output is a single point representing the mirror's location in the global image frame. This point can be integrated with other detected features—such as taillights or license plates—to improve vehicle orientation estimation, determine lateral spread.



(a) Cropped part **(b) Edges Detected** **(c) Filtered Edges**

Figure 3.6: Intermediate steps of side mirror detection

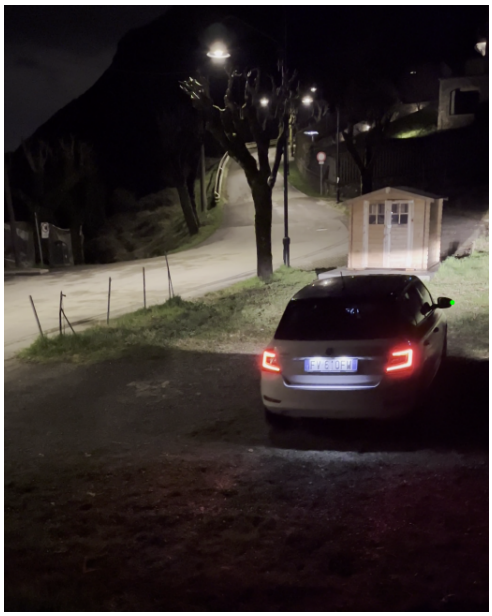


Figure 3.7: Example of side mirror detection from a frame

4 | Methodology

4.1. Methodology Overview

This chapter introduces the overall approach followed to estimate the space occupied by a moving vehicle using visual data acquired in low-light conditions. Based on the input video, the intrinsic calibration matrix and the extracted features obtained in the previous chapters, we developed and implemented four distinct localization methods.

Each method relies on geometric reasoning and leverages known or assumed properties of the scene, such as symmetry, planarity, or the car's motion profile. While each method aims at reconstructing the 3D position and orientation of the vehicle, they differ in the number and type of visual features used, and in their robustness to perspective limitations.

The methods presented in the following chapters are:

1. **Standard localization via homography**: uses four coplanar points on the rear facade of the car (e.g., rear lights and license plate corners) to estimate pose from a single image using homography decomposition.
2. **Nighttime localization from pair of images**: designed for low-light settings, this method uses symmetric rear light points tracked across two frames. It exploits temporal motion cues to estimate the pose of the vehicle on the road.
3. **Localization under poor perspective using out-of-plane symmetric features**: when perspective cues are weak, this method uses symmetric features located on different planes of the car's 3D structure (e.g., lights and mirrors) to recover pose by exploiting inter-frame symmetry and vanishing geometry.
4. **PnP-based vehicle pose estimation from key points**: this method estimates the vehicle's 3D pose from a single image by solving the Perspective-n-Point (PnP) problem using a set of known 3D key points (e.g., rear lights, license plate corners, and side mirror) and their 2D projections. By including non-coplanar points such as the side mirror, the method improves robustness and accuracy, especially in low-perspective conditions.

Each method is presented with its theoretical foundations, practical implementation, and a critical discussion of its applicability. Comparative results will be provided at the end to assess their relative performance and reliability in real-world conditions.

4.2. Method 1: Standard localization via homography

This method enables the estimation of a vehicle's 3D position and orientation from a single image, under the assumption of known intrinsic camera parameters and a simplified vehicle model. The approach relies on the identification of symmetric and well-localized features—typically the rear lights and the corners of the license plate—which are assumed to lie on the vehicle's rear plane. By exploiting known real-world distances between these features, it is possible to triangulate their positions in 3D space and reconstruct a rear-facing coordinate frame aligned with the vehicle geometry. The final output is a 3D bounding box that approximates the vehicle's occupied space. While this method requires only a single frame, its performance is highly sensitive to feature localization accuracy and the presence of sufficient perspective distortion in the image.

4.2.1. Theoretical Foundations

The mathematical framework underlying the method is founded on three pillars of projective geometry and photogrammetry.

Pinhole Camera Model and Back-Projection

A standard pinhole camera model is adopted, where the transformation from a 3D point in the camera space, $\mathbf{P} = [X, Y, Z]^T$, to an image point in homogeneous coordinates, $\tilde{\mathbf{x}} = [u, v, w]^T$, is described by the intrinsic matrix $K \in \mathbb{R}^{3\times3}$:

$$\lambda\tilde{\mathbf{x}} = K\mathbf{P}$$

where λ represents the scene depth. The inverse operation, or **back-projection**, is essential for 3D reconstruction. Given an image point $\mathbf{x}_{pix}$, its homogeneous coordinates $\tilde{\mathbf{x}}$ are mapped to a 3D directional ray $\mathbf{d}$ in the camera space, according to the relation:

$$\mathbf{d} = K^{-1}\tilde{\mathbf{x}}$$

This vector $\mathbf{d}$ defines the line on which the 3D point $\mathbf{P}$ lies, but not its exact position along it.

Triangulation with Geometric Constraints

To resolve the depth ambiguity, a metric constraint is introduced, namely a known real-world distance such as the license plate width, w_{plate}. Given two image points p_0 and p_1 and their corresponding normalized 3D rays $\mathbf{d}_0$ and $\mathbf{d}_1$, they form a triangle with the camera center. The angle θ between the two rays can be calculated via the dot product:

$$\theta = \arccos(\mathbf{d}_0 \cdot \mathbf{d}_1)$$

Assuming that the two 3D points $\mathbf{P}_0$ and $\mathbf{P}_1$ are at approximately the same depth from the camera, the triangle formed by $\mathbf{P}_0$, $\mathbf{P}_1$, and the camera center can be approximated

as isosceles. The average depth d can then be estimated geometrically:

$$d_{approx} = \frac{w_{plate}}{2\sin(\theta/2)}$$

This estimate is subsequently refined to ensure that the Euclidean distance between the reconstructed 3D points, $\mathbf{P}_0 = d_0\mathbf{d}_0$ and $\mathbf{P}_1 = d_1\mathbf{d}_1$, exactly satisfies the constraint $\|\mathbf{P}_1 - \mathbf{P}_0\| = w_{plate}$.

Vanishing Point Geometry

The vehicle's orientation is recovered by exploiting the properties of vanishing points. In projective geometry, parallel lines in 3D space appear to converge to a single point on the image plane, known as the vanishing point. By identifying two or more lines on the image known to be parallel in reality (e.g., the top and bottom edges of the license plate), their intersection point $\mathbf{v_p}$ can be computed. The back-projection of $\mathbf{v_p}$ yields a 3D vector $\mathbf{v}_{dir}$ that indicates the direction of these parallel lines in the camera space, corresponding to the vehicle's longitudinal axis.

$$\mathbf{v_p} = l_1 \times l_2 \quad \Longrightarrow \quad \mathbf{v}_{dir} = K^{-1}\mathbf{v_p}$$

Algorithmic Pipeline

The implementation of the method follows a defined sequence of operations.

1. **Preprocessing and Feature Extraction**: The image is corrected for optical distortions using known calibration parameters (K, `dist`). Subsequently, four coplanar keypoints are extracted: two corners of the license plate (`TL`, `TR`) and the two taillights (`L2`, `R2`). The coordinates of these points are also undistorted.

2. **3D Point Reconstruction**: The `triangulate_plate_points` function implements the constrained triangulation described above to compute the 3D coordinates, $\mathbf{P}_0$ and $\mathbf{P}_1$, of the license plate corners.

3. **Vanishing Point Estimation**: The function `compute_vanishing_direction` calculates the intersection of two lines in the image: the first connects the license plate corners (points `TL` and `TR`), and the second connects the taillights (points `L2` and `R2`). These lines, assumed to be parallel in the real world, intersect in the image due to perspective projection. Their intersection defines a vanishing point $\mathbf{v_p}$ in the image plane, which is then back-projected into a 3D direction vector $\mathbf{v}_{dir} = K^{-1}\mathbf{v_p}$ that approximates the vehicle's forward direction in the camera coordinate frame.

4. **Vehicle Coordinate Frame Construction**: The `build_vehicle_frame` function establishes a local coordinate system attached to the vehicle.

   - **X-axis**: Defined as the unit vector connecting the two 3D license plate points, $\mathbf{x}_{axis} = \text{normalize}(\mathbf{P}_1 - \mathbf{P}_0)$.

   - **Y-axis**: Calculated from the vanishing point direction $\mathbf{v}_{dir}$, made orthogonal to $\mathbf{x}_{axis}$ through a Gram-Schmidt orthogonalization process.

- **Z-axis**: Obtained via the cross product $\mathbf{z}_{axis} = \text{cross}(\mathbf{x}_{axis}, \mathbf{y}_{axis})$ to form a right-handed coordinate system.
- **Origin**: Placed at the center of the license plate segment, then translated along the local Z-axis.
- **Rotation Matrix**: The vehicle's rotation matrix R relative to the camera is constructed by stacking the three axes as columns: $R = [\mathbf{x}_{axis}|\mathbf{y}_{axis}|\mathbf{z}_{axis}]$.

5. **3D Bounding Box Generation**: The 8 vertices of the cuboid are defined in the vehicle's local coordinates, based on its known dimensions. Each local vertex $\mathbf{p}_{local}$ is then transformed into the camera's coordinate system using the estimated pose (rotation R and translation origin): $\mathbf{p}_{camera} = R\mathbf{p}_{local} + \text{origin}$.

4.2.2. Results and Discussion

The outcome of this method is shown in Figure 4.1. While the reconstructed bounding box follows the expected geometric structure, the projected result shows visible misalignment with the actual vehicle in the image. This discrepancy is primarily attributed to insufficient perspective in the observed view, which limits the accuracy of depth estimation during triangulation. In particular, nearly frontal or rear views result in collinear or weakly separated viewing rays, making depth inference unstable. Furthermore, small localization errors in selecting the symmetric features—especially the taillights points can lead to amplified inaccuracies in the 3D reconstruction. These limitations motivated the development of more robust techniques described in the subsequent sections.

Figure 4.1: Projection of the estimated 3D bounding box using the single-frame localization method. Weak perspective and minor feature localization errors affect the accuracy.

4.3. Method 2: Nighttime localization from pair of images

4.3.1. Introduction

This method aims to localize a vehicle at night by using a pair of images taken at different time instances (non consecutive). Unlike standard object detection methods, this approach leverages geometric reasoning and camera calibration to estimate the 3D position and orientation of the vehicle using only the taillights visible in nighttime conditions. The technique relies on identifying two pairs of corresponding symmetric points (in our case the taillights) on the rear of a moving vehicle in two non consecutive frames and applies principles from projective geometry and perspective analysis to infer spatial information.

4.3.2. Methodology

Knowing that the camera calibration matrix was previously computed, and having previously found the rear lights reference points, we define the image points as follows:

$$\text{Left taillight: } L_1, \quad \text{Right taillight: } R_1 \quad \text{(frame 1)}$$

$$\text{Left taillight: } L_2, \quad \text{Right taillight: } R_2 \quad \text{(frame 2)}$$

We then consider these image points in homogeneous coordinates:

$$\tilde{L}_1 = \begin{bmatrix} x_{L_1} \\ y_{L_1} \\ 1 \end{bmatrix}, \quad \tilde{R}_1 = \begin{bmatrix} x_{R_1} \\ y_{R_1} \\ 1 \end{bmatrix}, \quad \tilde{L}_2 = \begin{bmatrix} x_{L_2} \\ y_{L_2} \\ 1 \end{bmatrix}, \quad \tilde{R}_2 = \begin{bmatrix} x_{R_2} \\ y_{R_2} \\ 1 \end{bmatrix}$$

Vanishing points and vehicle motion

Two vanishing points needed for this method, are computed as:

- V_x: from the intersection of the taillight segments in each of the two frames:

$$\ell_1 = \tilde{L}_1 \times \tilde{R}_1, \quad \ell_2 = \tilde{L}_2 \times \tilde{R}_2$$

$$V_x = \ell_1 \times \ell_2$$

- V_y: from the intersection of two left rear lights, and two right rear lights of the two frames:

$$\ell_L = \tilde{L}_1 \times \tilde{L}_2, \quad \ell_R = \tilde{R}_1 \times \tilde{R}_2$$

$$V_y = \ell_L \times \ell_R$$

These vanishing points are back-projected into the camera coordinate system using the inverse of the calibration matrix K.

$$\vec{d_x} = K^{-1}V_x, \quad \vec{d_y} = K^{-1}V_y$$

This operation yields direction vectors, called *back-projected rays*, which represent the 3D directions in space corresponding to the vanishing points observed in the image. Each ray originates from the camera center and points along the direction in which a set of 3D parallel lines project to the same vanishing point in the image. These rays are then normalized to unit vectors for the next geometric computation.

$$\vec{d_x} = \frac{\vec{d_x}}{\|\vec{d_x}\|}, \quad \vec{d_y} = \frac{\vec{d_y}}{\|\vec{d_y}\|}$$

The vehicle motion is assumed to be straight (no steering) if:

$$\vec{d_x} \cdot \vec{d_y} \approx 0$$

which we checked in our code confirming our assumption.

3D Reconstruction of rear plane and depth estimation

To estimate the 3D position and orientation of the vehicle's rear plane, we exploit the geometric structure observed in a pair of images and the known camera calibration matrix $K \in \mathbb{R}^{3\times3}$. We consider 3D world points $\mathbf{X} \in \mathbb{R}^3$ project onto 2D image points $\mathbf{x} \in \mathbb{P}^2$ according to the mapping:

$$\tilde{\mathbf{x}} \sim K[R \,|\, t]\mathbf{X}$$

In our case, since we are working with fixed world geometry (i.e., the rear of the car) and trying to estimate depth and orientation, we simplify the analysis to the single-camera frame and image geometry.

Vanishing line of the rear plane The vehicle's rear is a planar surface, and it is observed in perspective, which introduces converging lines. In projective geometry, the intersection points of pairs of corresponding lines on this plane define **vanishing points** directions of parallel lines in 3D space. We identify two vanishing points:

- V_x: from horizontal taillight edges (width-wise direction),
- V_y: from vertical motion of taillights between frames (depth-wise motion).

The **vanishing line** ℓ of the rear plane is obtained as the cross product of the two vanishing points in homogeneous image coordinates:

$$\ell = V_x \times V_y$$

This line represents the intersection of the rear plane with the image plane, meaning it contains all the vanishing points of lines lying on the rear surface.

Figure 4.2: Lights segments and resulting vanishing line.

Computing the normal plane In projective geometry and camera imaging, every line ℓ in the 2D image corresponds to a 3D plane in space that passes through the camera's optical center and contains all the projecting rays onto that line. This is because the camera center is a fixed point in 3D space, and all points along the image line are projections of 3D points lying on some plane intersecting the camera center.

The vanishing line ℓ, defined as the cross product of two vanishing points V_x and V_y:

$$\ell = V_x \times V_y$$

represents the image of the rear plane of the vehicle. Intuitively, this line corresponds to the horizon line of that plane in the image, containing all vanishing points of directions lying on the rear surface.

To determine the orientation of this plane in 3D, we back-project the image line ℓ into the camera coordinate system using the transpose of the camera intrinsic matrix K:

$$\vec{n} = K^\top \ell$$

This operation produces a vector $\vec{n}$ that is **normal** (perpendicular) to the corresponding 3D plane in the camera coordinate frame.

This result can be understood by considering the duality between points and lines in projective geometry, combined with the role of the intrinsic matrix K:

- The intrinsic matrix K maps a 3D direction vector $\mathbf{d}$ expressed in the camera coordinate system to its corresponding point $\tilde{\mathbf{p}}$ in the normalized image plane through the relation:

$$\tilde{\mathbf{p}} = K\mathbf{d}$$

Here, $\tilde{\mathbf{p}}$ is a homogeneous 2D image point (or vanishing point) representing the projection of the 3D direction.

- Lines in the image, such as ℓ, are represented as vectors orthogonal to points (in the dual projective space). When we want to find the 3D plane corresponding to an image line, we need to "lift" this line back into 3D space.
- The transpose $K^\top$ operates on image lines ℓ to map them back into 3D vectors normal to the planes whose images they represent. More formally, since ℓ satisfies $\ell^\top \tilde{\mathbf{p}} = 0$ for all points $\tilde{\mathbf{p}}$ on that line, applying $K^\top$ to ℓ produces a vector $\vec{n}$ such that:

  $$\vec{n}^\top \mathbf{d} = 0$$

  for all directions $\mathbf{d}$ whose image points lie on ℓ.

  Thus, $\vec{n} = K^\top \ell$ defines the normal vector to the 3D plane associated with the image line ℓ.

Since $\vec{n}$ is defined up to scale, it is normalized to obtain a unit normal vector:

$$\vec{n} = \frac{\vec{n}}{\|\vec{n}\|}$$

The normalized vector $\vec{n}$ thus precisely encodes the orientation of the vehicle's rear plane relative to the camera frame. It points orthogonally outward from the rear surface, enabling further computations such as projecting points onto the plane or estimating distances from the camera.

Back-projected rays to the taillights Next, we compute the direction vectors from the camera center to the left and right taillights in the image. These are known as **back-projected rays**, obtained by applying the inverse of the calibration matrix to the image points:

$$\vec{r}_{L_1} = \frac{K^{-1}\tilde{L}_1}{\|K^{-1}\tilde{L}_1\|}, \quad \vec{r}_{R_1} = \frac{K^{-1}\tilde{R}_1}{\|K^{-1}\tilde{R}_1\|}$$

Each ray represents the direction in 3D along which a taillight lies, assuming the scene is rigid and the taillight is a fixed point in the world.

Angle between rays and depth estimation Given these rays, the angular separation between them in 3D space encodes information about the apparent size of the taillight segment. The angle θ between the rays is computed via the dot product:

$$\cos(\theta) = \vec{r}_{L_1} \cdot \vec{r}_{R_1}, \quad \theta = \arccos(\cos(\theta))$$

This angle tells us how "wide" the taillight separation appears from the camera's point of view. Given the **real-world width** w between the taillights (e.g., from CAD data),

we can apply a simple geometric principle to estimate the depth d of the car's rear from the camera:

$$d = \frac{w}{2\sin(\theta/2)}$$

This formula arises from the **law of sines** in triangle geometry. The rays $\vec{r}_{L_1}$ and $\vec{r}_{R_1}$ define a triangle with a known base w and angle θ at the vertex (camera center). The depth d corresponds to the distance from the camera to the base of the triangle, projected along the median ray.

3D reconstruction and bounding box

3D coordinates of the taillights:

$$\vec{P}_{L_1} = d \cdot \vec{r}_{L_1}, \quad \vec{P}_{R_1} = d \cdot \vec{r}_{R_1}$$

The midpoint between taillights is:

$$\vec{C}_{\text{rear}} = \frac{\vec{P}_{L_1} + \vec{P}_{R_1}}{2}$$

The rear ground contact point is computed by shifting vertically:

$$\vec{C}_{\text{ground}} = \vec{C}_{\text{rear}} + h \cdot \vec{n}$$

where h is the taillight height from the ground.

The local coordinate frame of the vehicle is defined as:

$$\text{Forward: } \vec{f} = \vec{d}_y$$

$$\text{right: } \vec{r} = \frac{\vec{r}_{R_1} - \vec{r}_{L_1}}{\|\vec{r}_{R_1} - \vec{r}_{L_1}\|}$$

$$\text{up: } \vec{u} = \vec{n}$$

Using the vehicle's CAD dimensions (length l, width w, height h), we are able to construct the 8 corners of the 3D bounding box.

Projection and visualization

Each 3D corner $\vec{P}_i$ is projected into 2D image space as:

$$\tilde{p}_i = K\vec{P}_i, \quad p_i = \left(\frac{\tilde{p}_i[0]}{\tilde{p}_i[2]}, \frac{\tilde{p}_i[1]}{\tilde{p}_i[2]}\right)$$

Figure 4.3 shows the final result of the pose estimation using only rear taillight cues under nighttime conditions.

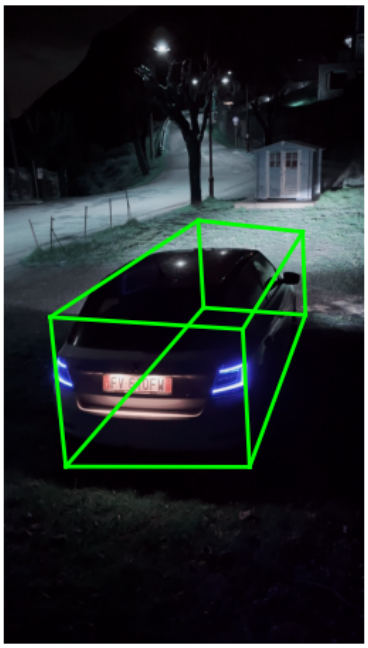


Figure 4.3: Bounding box resulting from method 2

4.4. Method 3: Poor perspective localization using out-of-plane features

4.4.1. Introduction

The theoretical foundation of Method 3 addresses the fundamental limitation of standard homography-based localization when perspective effects are insufficient. As demonstrated in the professor's slides, the standard homography-based method can fail with poor perspective, particularly when vanishing points are close to infinity, necessitating an iterative approach that can handle such degenerate cases.

4.4.2. Theoretical framework

Geometric foundation and the poor perspective problem

The core geometric insight behind this method is the reduction of the full 3D-to-2D projection problem to a simplified 2D-to-1D projection problem, involving a single unknown parameter: the rotation angle θ between the camera's x-axis and the direction of the vehicle's lateral axis (i.e., the direction of segment AB or CD). Following the professor's methodology, rather than using the actual 3D points E, A, B, and F, the method considers vertically translated versions of these points whose z-coordinates match the z-coordinate of the camera center O, thus projecting them onto a horizontal plane through O.

This transformation simplifies the projection model: since all model points now lie on a

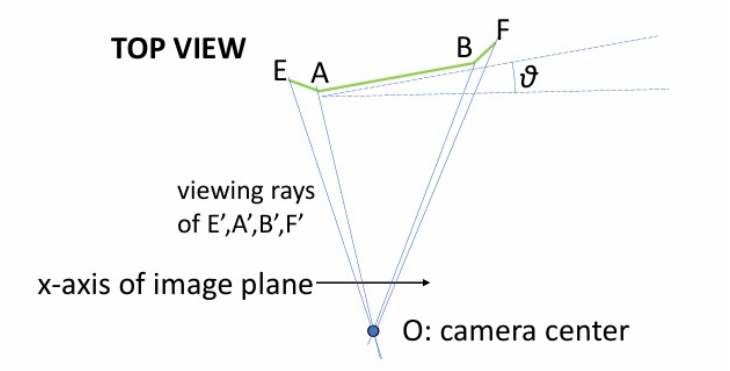


Figure 4.4: Top-view schematic showing camera center O, vehicle direction, and rotation angle θ

horizontal plane through the camera center, the problem reduces to estimating a 2D-to-1D projection governed by the single angle θ, which can be recovered through geometric constraints.

Iterative pose estimation framework

The method employs an iterative refinement loop that alternates between enforcing geometric constraints and updating pose parameters. While individual geometric cues may be unreliable under poor perspective, their integration into a global iterative system yields a robust and solvable estimation framework.

Triangulation with known distances Given two image points with a known 3D separation, triangulation is used to recover their 3D positions. For points A and B with known distance `AB_distance` and estimated direction $\mathbf{u}_{AB}$, we solve:

$$t \cdot \mathbf{r}_A - s \cdot \mathbf{r}_B = \texttt{AB\_distance} \cdot \mathbf{u}_{AB} \tag{4.1}$$

where $\mathbf{r}_A$ and $\mathbf{r}_B$ are the normalized camera rays from back-projection, and t, s are depth scalars.

Back-plane normal computation The orientation of the vehicle's back plane is obtained from triangulated 3D points. For example, using points A, B, C, and D:

$$\mathbf{n}_{\text{back}} = \texttt{normalize}\left((\mathbf{B}_{3d} - \mathbf{A}_{3d}) \times (\mathbf{D}_{3d} - \mathbf{C}_{3d})\right) \tag{4.2}$$

Camera vertical direction The vertical direction of the camera is computed by rotating the back-plane normal $\mathbf{n}_{\text{back}}$ around the vehicle's lateral axis $\mathbf{u}_{\text{dir}}$ by the known inclination angle ϕ:

$$\mathbf{v}_{\text{cam}} = \cos(\phi) \cdot \mathbf{n}_{\text{back}} + \sin(\phi) \cdot (\mathbf{u}_{\text{dir}} \times \mathbf{n}_{\text{back}}) + (1 - \cos(\phi)) \cdot \langle \mathbf{u}_{\text{dir}}, \mathbf{n}_{\text{back}} \rangle \cdot \mathbf{u}_{\text{dir}} \tag{4.3}$$

Horizon Line Projection and θ Computation

A key step in the estimation pipeline involves projecting selected image points onto the horizon line to extract the yaw angle θ.

Vertical Vanishing Point The vertical vanishing point $\mathbf{V}_z$ is computed from the camera's vertical direction:

$$\mathbf{V}_z = \mathbf{K} \cdot \mathbf{v}_{\text{cam}} \tag{4.4}$$

Horizontal Vanishing Line The horizon line l_∞ is derived from the same vertical direction:

$$l_\infty = \mathbf{K}^{-T} \cdot \mathbf{v}_{\text{cam}} \tag{4.5}$$

Horizon Projection Each image point is projected onto the horizon by computing the intersection of the vertical line through $\mathbf{V}_z$ with l_∞:

$$\texttt{point\_on\_horizon} = (\mathbf{V}_z \times \mathbf{p}) \times l_\infty \tag{4.6}$$

Yaw Angle Computation Once points A'' and B'' are projected on the horizon, the angle θ is extracted as:

$$\theta = \arctan 2(B''_y - A''_y, B''_x - A''_x) \tag{4.7}$$

Convergence and Stability

The iterative algorithm converges by repeatedly satisfying the above constraints while adjusting the estimated angle and geometric parameters.

4.4.3. Methodology

Initial Pose Estimation from Rear Plane Features

The process begins with a coarse pose estimation using four coplanar points on the rear facade of the car—rear lights (A, B) and license plate corners (C, D). This estimate is based on vanishing point analysis: the lateral vanishing point V_x is obtained from the intersection of the lines AB and CD and then back-projected using the camera matrix K to obtain the lateral direction in 3D. Although this estimate is often inaccurate under poor perspective, it establishes a geometrically consistent initialization, including the back-plane normal and a first approximation of the camera's vertical direction.

Triangulation with Known Physical Constraints

Once symmetric 2D points and their real-world distances are known from the vehicle CAD model, triangulation is used to reconstruct their 3D coordinates. The method back-projects the points to normalized rays and solves for depths that satisfy both the projection geometry and the known distance between points:

$$t \cdot \mathbf{r}_A - s \cdot \mathbf{r}_B = \texttt{AB\_distance} \cdot \mathbf{u}_{AB} \tag{4.8}$$

This constrained triangulation ensures physical plausibility and geometric consistency with the vehicle model even in the presence of poor perspective or noisy detections.

Horizon Line Projection and Yaw Angle Computation

The yaw angle θ is refined by projecting selected image points onto the horizon line. The vertical vanishing point V_z is computed from the rotated back-plane normal using Rodrigues' formula. The horizontal vanishing line l_∞ is then computed from V_z and the calibration matrix K.

Each image point is projected onto l_∞ by intersecting the vertical line through V_z with the horizon line:

$$\texttt{point}_{\text{horizon}} = (\mathbf{V}_z \times \mathbf{p}) \times l_\infty \tag{4.9}$$

The yaw angle θ is then computed between projected points (e.g., A'', B''):

$$\theta = \arctan 2(B''_y - A''_y, B''_x - A''_x) \tag{4.10}$$

When multiple symmetric pairs are available, a circular mean is used to robustly estimate the average yaw angle.

Iterative Refinement Framework

The core refinement loop performs the following updates at each iteration:

- Update vehicle lateral direction (direction along X axis) using the current θ
- Perform triangulation using updated rays and known distances
- Refine the back-plane normal via cross product or SVD
- Rotate the back-plane normal to compute the new vertical direction
- Recompute vanishing geometry (V_z, l_∞)
- Project symmetric points onto l_∞ and update θ

The updated yaw estimates are averaged using circular statistics. The loop continues until convergence criteria are met—either a small enough change in θ, consistent geometry, or a max iteration count.

4.4.4. Results and Evaluation

The proposed method was evaluated on a test frame under poor perspective conditions, using the rear lights, license plate corners, and external rear lights (E–F) as symmetric features for pose estimation.

Iterative Yaw Angle Convergence

The method started from an initial yaw angle estimate of $\theta_{\text{init}} = 9.46°$. Through the iterative refinement loop, the angle was progressively updated based on vanishing geometry and horizon line projections. The convergence log is as follows:

- Iteration 1: $\theta = 6.27°$, $\Delta\theta = -2.34°$
- Iteration 2: $\theta = 5.89°$, $\Delta\theta = -1.88°$
- Iteration 3: $\theta = 5.58°$, $\Delta\theta = -1.55°$
- Iteration 4: $\theta = 4.37°$, $\Delta\theta = -1.28°$
- Iteration 5: $\theta = 4.17°$, $\Delta\theta = -0.21°$
- Iteration 6: $\theta = 4.14°$, $\Delta\theta = -0.04°$

The process converged after 7 iterations, yielding a final yaw estimate of $\theta_{\text{final}} = \mathbf{4.14}°$. This demonstrates the effectiveness of the proposed method in refining vehicle orientation even under weak perspective cues.

Geometric Consistency of Triangulated Features

The method achieves high consistency between the triangulated 3D distances and the real-world measurements from the CAD model:

Feature Pair	**Computed Distance (m)**	**Expected Distance (m)**
A–B (rear lights, internal)	0.860	0.86
C–D (license plate corners)	0.520	0.52
E–F (rear lights, external)	1.400	1.40

This confirms that the triangulation module, guided by the estimated yaw and direction vector, remains geometrically reliable.

Final Projection Result

Figure 4.5 shows the final result of the pose estimation: a 3D bounding box is projected onto the image using the computed pose and the simplified car model. The orientation aligns visually with the vehicle's position, confirming the correctness of the yaw and translation estimation.

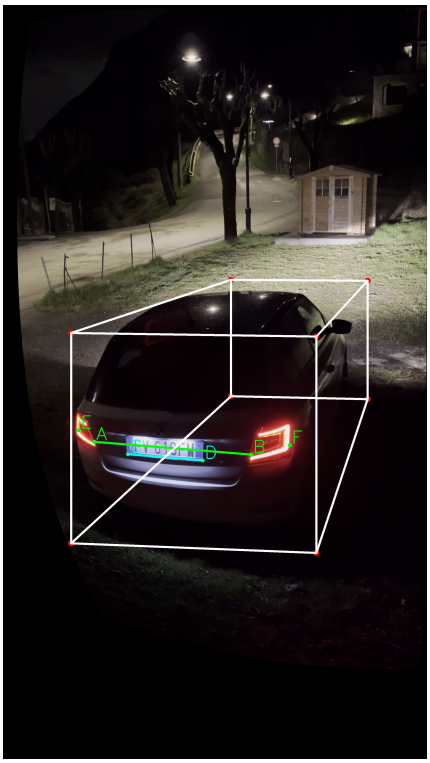


Figure 4.5: Final pose estimation result using Method 3. The bounding box aligns with the car's geometry, demonstrating successful localization under poor perspective.

4.5. Method 4: PnP-based vehicle pose estimation from key points

4.5.1. Introduction

This method leverages the Perspective-n-Point (PnP) problem to estimate the 3D pose of a vehicle from a single image frame. Unlike homography-based approaches, PnP allows for accurate reconstruction even when using out-of-plane points, such as the side mirror. The method is implemented in two main variants: one using 4 coplanar points (rear lights and license plate), and another using 5 points, including the mirror, to assess the impact of additional 3D depth on localization accuracy.

4.5.2. Theoretical framework

The perspective-n-point problem

The PnP problem involves estimating the position and orientation (pose) of a camera relative to a known 3D object, given:

- A set of $n \geq 4$ known 3D points in the object coordinate system,
- Their corresponding 2D projections in the image plane,

- The camera's intrinsic matrix K.

Mathematically, the PnP problem seeks to find the rotation matrix $\mathbf{R} \in SO(3)$ and translation vector $\mathbf{t} \in \mathbb{R}^3$ that satisfy the camera projection equation:

$$s \cdot \mathbf{p}_i = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}] \cdot \mathbf{P}_i \tag{4.11}$$

where $\mathbf{p}_i$ represents the 2D image coordinates, $\mathbf{P}_i$ the corresponding 3D world coordinates, $\mathbf{K}$ the camera intrinsic matrix, and s a scale factor. The rotation matrix $\mathbf{R}$ belongs to the *Special Orthogonal Group* $SO(3)$, which denotes the set of all 3×3 orthogonal matrices with determinant equal to 1. This constraint ensures that $\mathbf{R}$ represents a proper rotation in 3D space, preserving lengths and angles without reflection.

PnP vs homography

The main distinction between PnP and homography methods (`findHomography`) lies in their assumptions:

- Homography assumes that all 3D points lie on the same plane (planar geometry).
- PnP makes no such assumption and performs well even with non-coplanar configurations. It is thus more robust in low-perspective conditions or when using points with different depths.

PnP algorithm variants

Modern PnP solvers employ different strategies for pose estimation. **Iterative PnP** methods require an initial pose estimate and refine it through non-linear optimization, typically providing higher accuracy when a reasonable initialization is available. **EPnP (Efficient PnP)** solvers, implemented through the `SOLVEPNP_EPNP` method, compute the pose using a non-iterative approach based on expressing 3D points as weighted sums of four virtual control points. This method offers computational efficiency and numerical stability without requiring initial pose estimates, making it particularly suitable for real-time applications. **PnPRANSAC** combines pose estimation with robust outlier detection using random sample consensus, making it suitable for noisy feature correspondences where some point matches may be incorrect.

4.5.3. Methodology

Four-point PnP approach: license plate and taillights

The first methodology employs a four-point correspondence system utilizing the vehicle's license plate and taillights positions. This approach leverages the geometric regularity and predictable spatial relationships of these vehicle features to establish reliable 3D-to-2D point correspondences.

Feature point selection and 3D model definition The 3D vehicle model is defined using four key reference points:

- **License plate corners**: two points representing the lower corners of the rear license plate.
- **Taillights points**: two points representing the rear taillights.

PnP solver implementation and comparison Two distinct PnP solving strategies were implemented and evaluated:

- **Iterative PnP with initial pose estimation**: This approach first computes a rough pose estimate using analytical methods, then refines this estimate through iterative optimization. The initial pose is derived from the license plate plane geometry, providing a stable starting point for convergence.
- **Direct PnP solution**: This method computes the pose directly from the four-point correspondences without requiring initialization. The solver employs robust estimation techniques to handle potential noise and outliers in the feature detection process.

Experimental evaluation revealed that both approaches converge to nearly identical solutions when applied to the same feature correspondences.

Five-point PnP approach: enhanced model with mirror integration

To improve pose estimation accuracy and provide additional geometric constraints, we extended the methodology to incorporate a fifth reference point: the vehicle's side mirror position.

Extended 3D model definition The enhanced model incorporates:

- **License plate corners** (2 points)
- **Headlight centers** (2 points)
- **Side mirror center** (1 point)

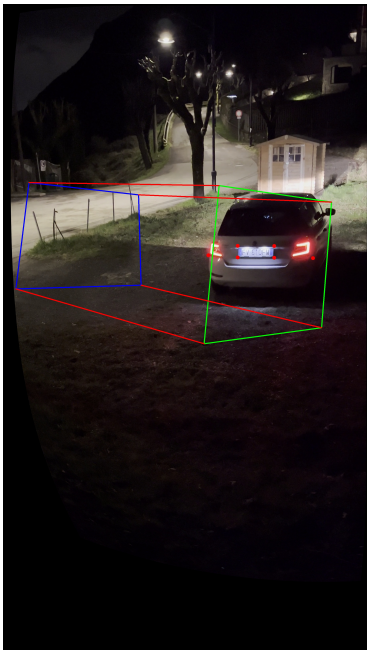
The addition of the side mirror point provides several advantages:

1. **Geometric diversity**: the mirror point lies off the rear plane, providing depth variation
2. **Constraint redundancy**: Additional correspondences improve pose estimation robustness
3. **Bounding box refinement**: the mirror position enables more accurate vehicle extent estimation

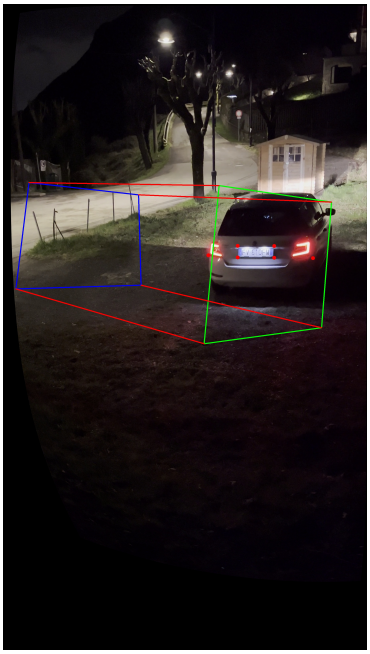
4.5.4. Results and visual comparison

Figure 4.6 compares the localization results obtained using the four-point model with two different PnP solvers: the Iterative algorithm and EPnP. Both methods yield nearly

identical poses and projected bounding boxes, confirming the reliability and consistency of our initial estimation of the Spose.



(a) 4-point PnP using Iterative solver



(b) 4-point PnP using EPnP solver

Figure 4.6: Pose estimation comparison using four points and different PnP solvers

Figure 4.7 shows the result obtained by extending the model with a fifth point corresponding to the vehicle's side mirror. The projected bounding box exhibits greater accuracy and spatial consistency, especially in the horizontal extent, validating the benefit of including out-of-plane points in the PnP setup.

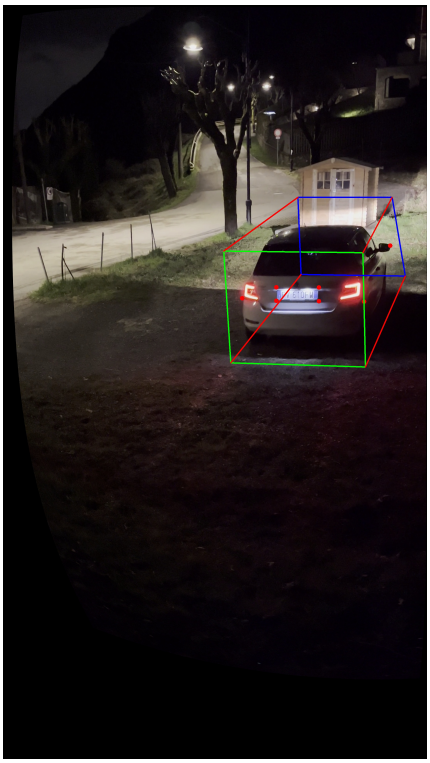


Figure 4.7: Pose estimation using five-point model (license plate, headlights, and mirror)

5 | Conclusion

This chapter summarizes the key findings of the proposed methods for vehicle space occupancy estimation under low-light conditions. It provides a comparative evaluation of the approaches introduced, highlighting their strengths, limitations, and suitability for different scenarios.

- **Method 1** relies on a homography computed from four coplanar vehicle points. While it is simple and computationally efficient, it heavily depends on strong perspective cues. When these cues are weak or absent—such as in frontal or rear views—the method often fails to recover a meaningful pose, limiting its applicability in real-world scenarios.
- **Method 2** builds upon geometric reasoning using vanishing points to estimate the angle between back-projected rays of the two taillights. This method demonstrates high accuracy and reliability when vanishing points can be reliably detected. It avoids strong planar assumptions and performs well under a wide range of perspectives. However, its dependence on precise line detection and vanishing point localization can become a weakness in low-resolution or cluttered images.
- **Method 3** introduces a more sophisticated and iterative approach, reducing the 3D-to-2D problem to a 2D-to-1D projection and refining pose estimates using known vehicle dimensions. It proves to be one of the most robust methods, especially under weak perspective conditions where other methods struggle. Its ability to recover accurate poses from minimal cues makes it highly suitable for real-world deployment. The trade-off lies in the increased computational complexity and reliance on iterative refinement.
- **Method 4** uses the Perspective-n-Point (PnP) framework to estimate vehicle pose from known 3D-to-2D correspondences. While it is highly flexible and accurate when accurate correspondences are available, its performance is sensitive to point detection quality and the assumption of correct vehicle geometry. Additionally, it may not perform well when all visible points lie on the same plane, causing pose ambiguity.

In summary, Methods 2 and 3 offer the best balance between accuracy and robustness. Method 2 excels when vanishing points are well-defined, while Method 3 provides resilience in more ambiguous scenes. Methods 1 and 4, although useful in specific contexts, face limitations in generalizability and reliability. Future work may explore hybrid approaches that combine vanishing point geometry with iterative refinement for even greater performance in diverse environments.

5.1. Comparison of Results

The figure 5.1 shows the comparison between the implemented methods across different frames.

5.2. Future Work

Future work will aim to automate the retrieval of vehicle data by implementing a neural network to recognize license plate characters. The recognized plate will be used to query public databases to obtain the car's make and model. This information will then allow access to 3D CAD model repositories, providing accurate geometrical data without manual input. Additionally, improvements will be made to the robustness of license plate and light detection, using more advanced object detection algorithms to handle challenging conditions such as occlusion or low visibility.

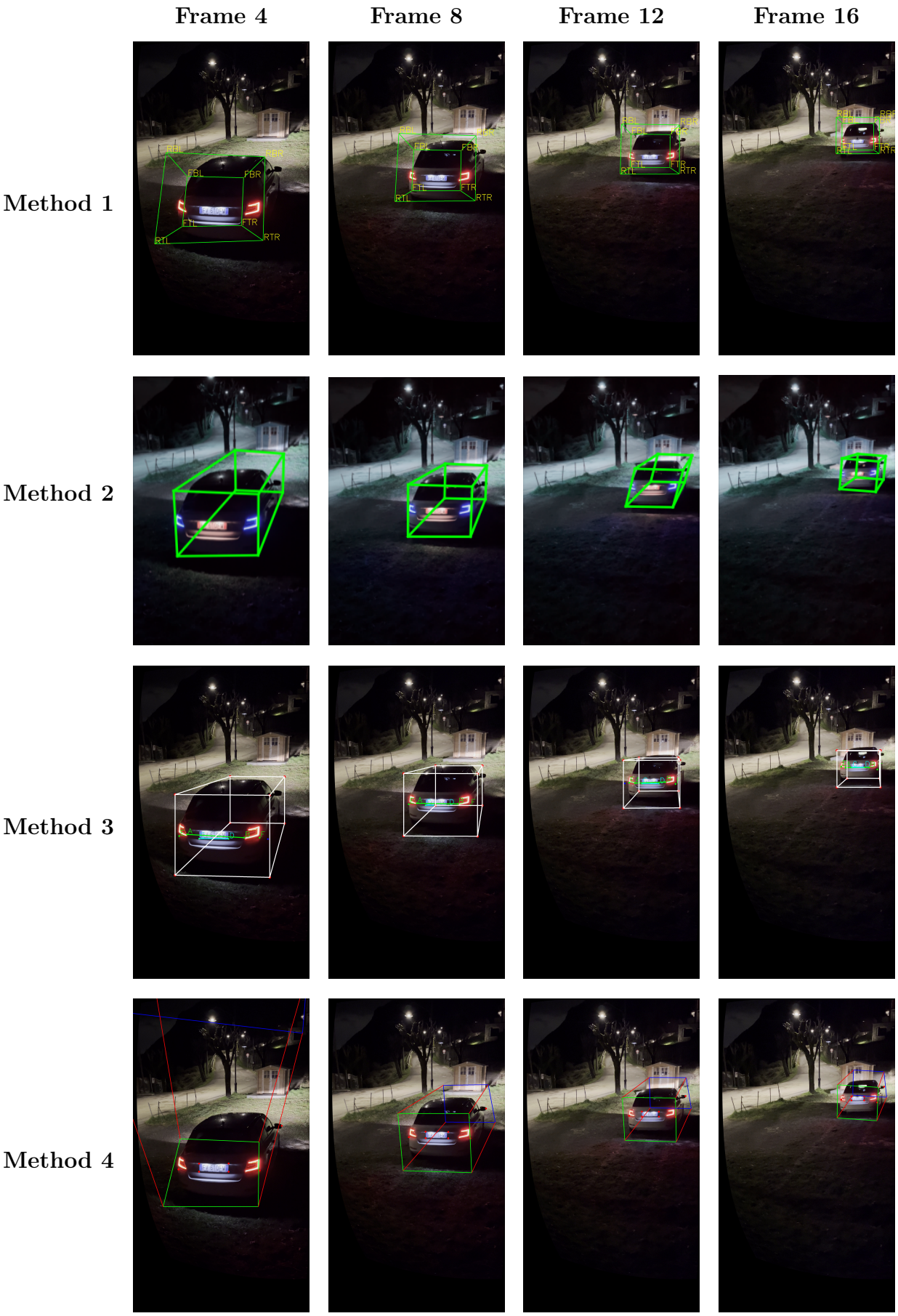


Figure 5.1: Comparison of results from different methods across four selected frames

List of Figures

Bibliography

[1] M. Ali et al. Real-time vehicle distance estimation using single view geometry. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, page xxx-xxx, 2020. Section 3.2.1 describes IPM for perspective distortion removal.

[2] M. Hong, S. Cheng, H. Huang, H. Fan, and S. Liu. You only look around: Learning illumination invariant feature for low-light object detection, 2024. URL `https://arxiv.org/abs/2410.18398`.

[3] Y. Liu, S. Li, L. Zhou, H. Liu, and Z. Li. Dark-yolo: A low-light object detection algorithm integrating multiple attention mechanisms. *Applied Sciences*, 15(9), 2025. ISSN 2076-3417. doi: 10.3390/app15095170. URL `https://www.mdpi.com/2076-3417/15/9/5170`.

[4] A. Rezaei, X. Yin, and F. Bai. Zero-calibration monocular 3d vehicle localization via satellite-ground alignment. *IEEE Transactions on Intelligent Vehicles*, 8(3):456–468, 2023.

[5] R. K. Satzoda and M. M. Trivedi. Symmetry-based vehicle detection and distance estimation using rear lights. In *International Conference on Intelligent Transportation Systems*, pages 2347–2354, 2019.

[6] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *International Conference on Image Processing*, pages 3645–3649, 2017.

[7] Y. Xiao and H. Liao. Lida-yolo: An unsupervised low-illumination object detection based on domain adaptation. *IET Image Processing*, 18:1178–1188, 2024. doi: 10.1049/ipr2.13017. URL `https://digital-library.theiet.org/doi/abs/10.1049/ipr2.13017`.

[8] Y. Yang, H. Zhen, Y. Huang, and J. J. Yang. Enhancing nighttime vehicle detection with day-to-night style transfer and labeling-free augmentation, 2024. URL `https://arxiv.org/abs/2412.16478`.

[9] L. Zhang, W. Xu, C. Shen, and Y. Huang. Vision-Based On-Road Nighttime Vehicle Detection and Tracking Using Improved HOG Features. *Sensors*, 24(5):1590, 2024. doi: 10.3390/s24051590.

[10] Y. Zhang, X. Wang, W.-L. Wang, and X. Liu. Bytetrack: Multi-object tracking

by associating every detection. In *European Conference on Computer Vision*, pages 1–17, 2022.

[11] Q. Zou, H. Ling, Y. Pang, Y. Huang, and M. Tian. Joint head-light pairing and vehicle tracking by weighted set packing in nighttime traffic videos. *IEEE Transactions on Intelligent Transportation Systems*, 19:1950–1961, 2017. doi: 10.1109/TITS.2017.2745684.