Image Analysis and Computer Vision

Homework 2024/2025

Francesco Genovese

Student ID: 10614236

Lecturer: Professor Caglioti Vincenzo

Assistant Professors: Arrigoni Federica, Boracchi Giacomo, Magri Luca

Academic Year: 2025–2026

Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

Contents

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Estimating the three-dimensional pose of a moving vehicle from a single monocular camera is a classical problem in computer vision that sits at the intersection of geometric reasoning and real-time perception. It has direct relevance to a wide range of applications, including advanced driver assistance systems (ADAS), traffic surveillance, autonomous driving, and road safety analysis. In all of these contexts, knowing not only where a vehicle is in the image plane but also its actual position and orientation in three-dimensional space is fundamental. A 2D bounding box locates a vehicle on screen; a 3D pose estimate tells you how far away it is, in which direction it is heading, and how much physical space it occupies on the road.

The core difficulty of the problem lies in recovering depth from a single viewpoint. A standard camera projects a three-dimensional scene onto a two-dimensional image plane, and this projection is not invertible without additional constraints. To recover the missing dimension, one must rely on prior knowledge about the geometry of the observed scene or the observed object — in this case, the vehicle itself. When the intrinsic parameters of the camera are known and a calibrated model of the vehicle is available, it becomes possible to formulate the problem as one of geometric pose estimation rather than unconstrained 3D reconstruction.

This project was developed in the context of a computer vision course assignment, the goal of which was to build a system capable of estimating the frame-by-frame 3D pose of a moving vehicle observed by a single fixed calibrated camera, and to represent the resulting spatial occupancy as a 3D bounding box aligned with the estimated pose. The system operates under nighttime conditions, where the dominant visible features are the vehicle's rear tail lights — a constraint that shapes every design decision described in this report.

1.2 Assignment Specification

The assignment defined three independent geometric approaches to the localization problem, referred to as tasks, each suitable for different operational conditions.

Task 1 addressed daytime scenarios in which the license plate is clearly visible. The proposed method involved detecting the four corners of the rear plate, computing a homography between the known 3D plate plane and the image, and decomposing it to extract the rotation and translation of the plane with respect to the camera. While geometrically elegant, this approach is known to become unstable when the plate is roughly fronto-parallel to the image plane, which occurs when the camera is far from the vehicle or nearly aligned with its optical axis.

Task 2 addressed nighttime scenarios using only the rear tail lights and their apparent motion across two consecutive frames. The method exploits the fact that, under a rectilinear motion assumption, the image trajectories of the two lights converge toward a common vanishing point V_x. Back-projecting V_x through the inverse intrinsic matrix K^{-1} yields the 3D direction of motion. Combined with the known real-world distance between the lights as a metric constraint, this direction allows estimating the depth of the plane π on which the lights lie, from which the full pose follows.

Task 3 was aimed at cases where perspective is weak — typically when the camera is far from the vehicle and the rear face is nearly fronto-parallel. The proposed strategy was to exploit pairs of non-coplanar symmetric points on the vehicle to recover the horizontal rotation angle θ from the apparent asymmetry introduced by perspective.

The assignment explicitly stated that the three tasks represent alternative strategies for the same problem and need not be concatenated.

1.3 Scope of This Work

This project focuses on the nighttime scenario defined by Task 2. Tasks 1 and 3 were studied and understood but not implemented, as the available video material and operational conditions made Task 2 the natural starting point.

While the vanishing point formulation is theoretically well-founded, its behaviour on real footage revealed a systematic numerical instability that prevents reliable metric estimation on the available data. The direct PnP approach solves the same geometric problem, recovering the 6-DoF pose of the vehicle from image observations, under a more constrained and better- conditioned formulation. This shift is therefore not a departure from the spirit of the assignment, but a principled engineering response to the practical limitations of the data; the full analysis is provided in Sections 3.5 and 3.6.

1.4 Assumptions

The system operates under a set of assumptions that are either stated in the assignment specification or introduced during implementation to keep the problem tractable.

The road surface is assumed to be planar. This is standard in vehicle localization from road-facing cameras and is a reasonable approximation for the scenarios considered. The vehicle is assumed to be symmetric about a vertical longitudinal plane, which is exploited when pairing left and right keypoints. The motion between consecutive frames is assumed to be simple — either a straight-line translation or a constant-curvature steer — consistent with the assignment specification. The geometric model of the vehicle (dimensions, 3D

coordinates of the tail lights and license plate, bounding box vertices) is assumed to be known and fixed; these values are derived from publicly available specifications of the Toyota Aygo X and from manual measurements on reference images.

Finally, the camera is fixed and its intrinsic parameters are known, having been estimated through a standard chessboard calibration procedure described in Chapter 3.

Chapter 2

Data Acquisition and Experimental Constraints

The data used in this project were collected in an uncontrolled outdoor setting using a consumer smartphone camera. The recordings depict a Toyota Aygo X performing both translational and rotational manoeuvres in a low-traffic parking area at night, chosen to approximate the conditions of an urban traffic monitoring scenario.

The camera was held at head height, tilted downward toward the scene to emulate the typical inclination of a traffic surveillance camera mounted on a traffic light or road gantry. This configuration was chosen to replicate, as closely as possible within the available means, the perspective geometry of a fixed infrastructure camera. No tripod or mounting rig was available, which introduces a degree of random hand-motion noise in the footage; however, since the system operates on a fixed-camera assumption and the vehicle is the only moving object of interest, this does not affect the geometric validity of the approach.

Subsequently, a calibration phase was conducted to estimate the intrinsic parameters of the camera. A standard planar chessboard pattern was used, following the Zhang method as implemented in OpenCV. The calibration images were acquired in a variety of positions and orientations to ensure adequate coverage of the image plane. While the calibration dataset is not of laboratory quality, the resulting intrinsic matrix and distortion coefficients were sufficient for the purposes of the project, as confirmed by the low mean reprojection error reported at runtime.

A fundamental constraint of the experimental setup is the absence of extrinsic calibration: the precise position and orientation of the camera with respect to the world reference frame were not measured. This means the estimated vehicle pose is expressed in the camera frame and cannot be transformed into absolute world coordinates without additional information. For the purposes of this project — estimating the 3D occupancy of the vehicle relative to the camera point of view. This is not a limiting factor, and all downstream computations are carried out entirely in the camera frame.

Chapter 3

Proposed System Structure

3.1 Camera Calibration

Before any geometric reasoning can take place, the intrinsic parameters of the camera must be known. Camera calibration is the process of estimating the mapping between 3D world coordinates and 2D image coordinates, which is encoded in the intrinsic matrix K and the distortion coefficients. In this project, calibration serves two distinct purposes: it allows undistorting the image so that straight lines in the world map to straight lines in the image, and it provides the focal length and principal point needed to back-project 2D image points into 3D camera-frame directions.

The calibration was performed using a planar chessboard pattern, following the standard Zhang method as implemented in OpenCV. A set of images of the chessboard was acquired from multiple viewpoints and distances, covering the full range of positions that the vehicle is expected to occupy in the scene. For each image, OpenCV detects the inner corner grid at sub-pixel accuracy; the known 3D coordinates of the corners (in the chessboard frame, where $Z = 0$ on the pattern plane) are matched with the detected 2D pixel coordinates to build an overdetermined system. A nonlinear minimisation over reprojection error then yields K and the distortion coefficients.

The calibration output — the 3×3 matrix K containing the focal lengths f_x, f_y and the principal point (c_x, c_y), together with the five-parameter distortion vector — is saved in NumPy format and loaded at system startup. All subsequent geometric computations use the same K, ensuring consistency between calibration and pose estimation.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The mean reprojection error obtained on the calibration images is reported in the system output at runtime, and should ideally be below 1 pixel for the PnP solver to produce reliable results.

3.2 Tail Light Detection and HSV Masking

Under nighttime conditions, the rear tail lights are the only reliably visible and geometrically meaningful features on the vehicle. Their detection is therefore the first and most critical step in the pipeline. The approach is based on colour filtering in the HSV colour space, which separates chrominance from luminance and makes it easier to isolate the specific appearance of illuminated red lights independently of global scene brightness.

Red in HSV occupies two separate intervals on the hue axis, one near 0 and one near 180, because the hue channel is circular. Two masks are therefore computed and combined: the first covers hues from 0 to 18, the second from 170 to 180. Both masks additionally threshold the saturation and value channels, the latter being set high ($V \geq 190$) to retain only bright, actively illuminated pixels and discard dark surfaces that might appear reddish under certain lighting conditions, such as reflections on the white bodywork of the car, other parked car lights, or the asphalt.

The resulting binary mask is cleaned with morphological opening to remove isolated noise pixels, and closing to fill small holes inside the light blobs. Connected-component analysis then extracts the individual blobs, each described by its contour, centroid, area, and aspect ratio.

Not all blobs that pass the colour filter correspond to tail lights. A first stage of geometric filtering rejects blobs that are too small or too large (area thresholds set relative to frame area), too wide relative to their height (the tail lights of the Aygo X are taller than they are wide), or located in the bottom 20% or top 10% of the frame in the reference video. The surviving candidates are then scored pairwise by `CandidateSelector` using four criteria: absolute size, vertical alignment, horizontal separation, and area similarity between the two blobs. The pair with the highest composite score is selected as the left and right tail light.

For each accepted blob, three geometric sub-points are extracted rather than just the centroid:

- **outer** — the laterally outermost Canny edge pixel found by scanning the mid-height band of the blob; more geometrically stable than the centroid when blob shape changes with distance or partial occlusion.
- **top** — the highest pixel of the contour.
- **bottom** — the mean of all contour pixels above the 90th percentile of the Y-coordinate distribution, chosen to avoid contamination from road surface reflections or the rear bumper, which systematically affect the absolute lowest pixel.

Parameter tuning for the HSV thresholds and geometric filters was performed interactively using dedicated Jupyter notebooks, in which sliders control each threshold and the resulting mask is visualised in real time on frames extracted from the target video. This allowed calibrating the detector specifically for the illumination conditions and vehicle appearance in the footage, without modifying the main codebase.

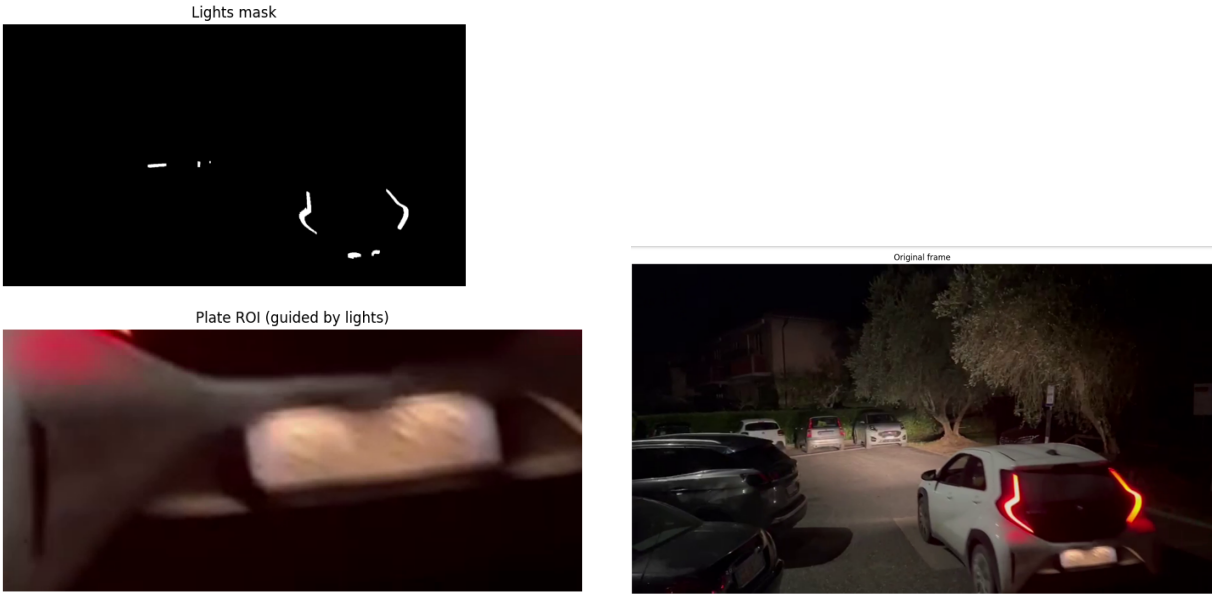


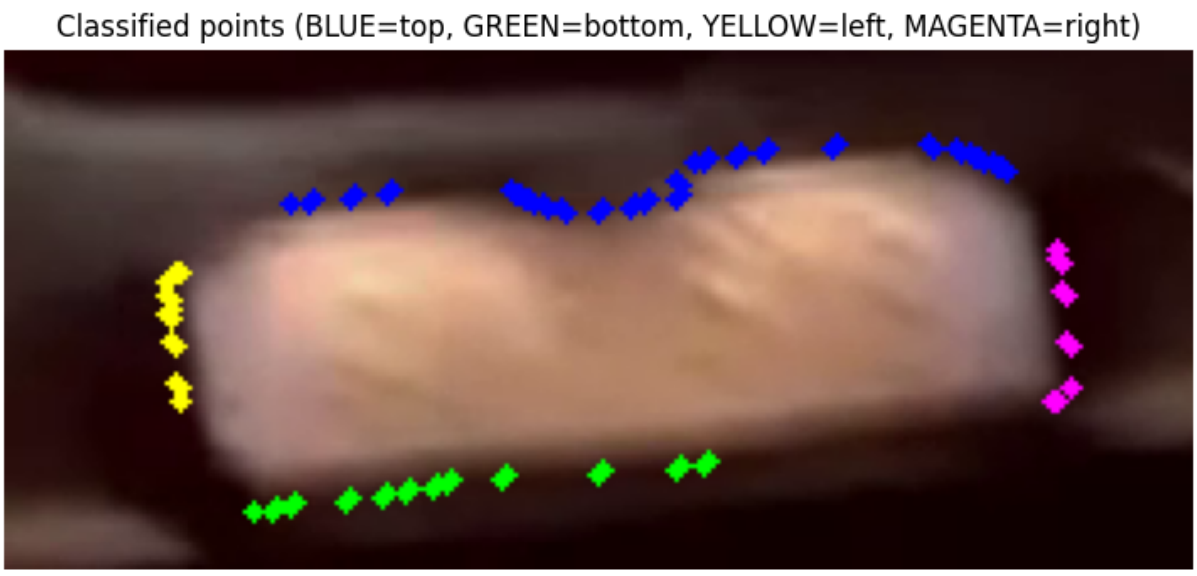
Figure 3.1: Left: HSV light mask with plate ROI guided by detected tail lights. Right: original frame used as reference for detection tuning.

3.3 License Plate Detection

In addition to the tail lights, the system attempts to detect the rear license plate as an auxiliary feature. The plate provides a horizontal edge at a known height in the vehicle reference frame (the bottom edge of the plate sits at approximately 40 cm from the ground), which adds a constraint on the vertical position and orientation of the vehicle that the tail lights alone cannot provide.

Plate detection follows a classical pipeline: Gaussian blurring to reduce sensor noise, Canny edge detection, morphological closing to bridge gaps in the plate border, contour extraction, and geometric filtering. The filtering retains only quadrilaterals whose area falls within the expected range for a license plate at the distances of interest, whose aspect ratio is between 2.5 and 6.0 (an Italian plate has a nominal aspect ratio of approximately 4.7), and whose solidity (the ratio of contour area to convex hull area) exceeds 0.7. The four corners of accepted contours are ordered in a canonical top-left, top-right, bottom-right, bottom-left sequence.

In practice, the plate is not always detectable. At large distances it becomes too small; at close range it may be overexposed. The system therefore treats the plate as an optional feature: it detects the brightness mask of the plate region and, after post-processing, identifies the main contour border, which is finally analysed through linear regression on sampled points, providing an approximate estimate of the plate edges. Only the bottom border was found to be reliable from the available data, and was therefore the only one used to refine the pose estimation.




```
Lines computed: top=True, bottom=True, left=True, right=True

Plate corners: {'TL': (35, 37), 'TR': (229, 21), 'BL': (38, 100), 'BR': (230, 80)}
```

Figure 3.2: Plate detection pipeline on the reference frame. top: plate light mask isolated from the HSV filter and processed. bottom: main contour border with classified keypoints (top, bottom, left, right corners).

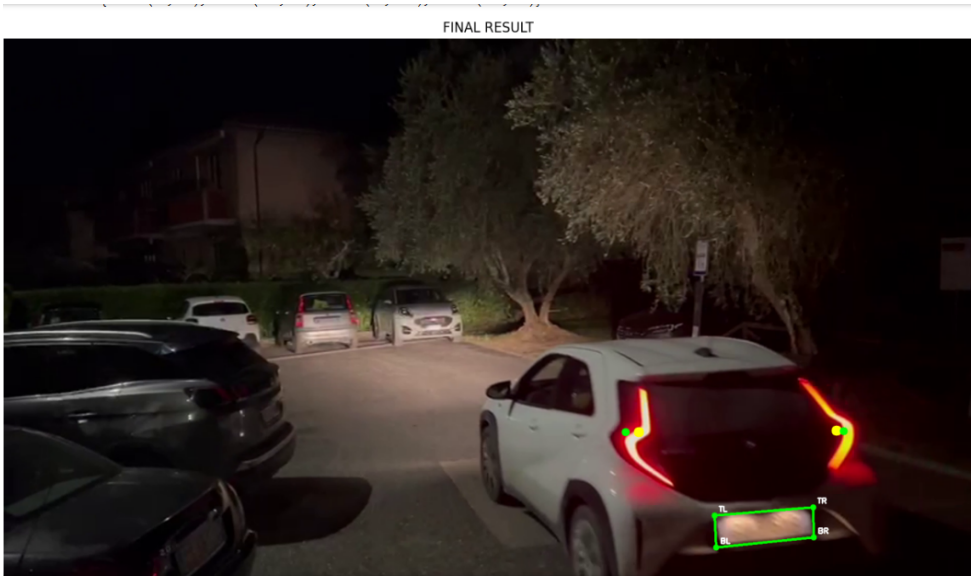


Figure 3.3: License plate detection result on the reference frame, showing the four detected corner points.

The plate detector was calibrated using a dedicated Jupyter notebook, which allowed exploring the sensitivity of the pipeline to Canny thresholds, contour area bounds, and polygon approximation tolerance on representative frames from the video.

3.4 Vehicle Reference Frame and Bounding Box Definition

A central design decision in any model-based pose estimation system is the choice of reference frame. Here, the vehicle coordinate system is defined with its origin at the centre of the rear axle projected onto the ground plane. The X axis points forward along the vehicle's longitudinal axis, Y points to the left, and Z points upward. This choice is natural for a vehicle on a planar road surface: the ground plane corresponds to $Z = 0$, and the origin is a physically meaningful point that lies on the road rather than inside the vehicle body.

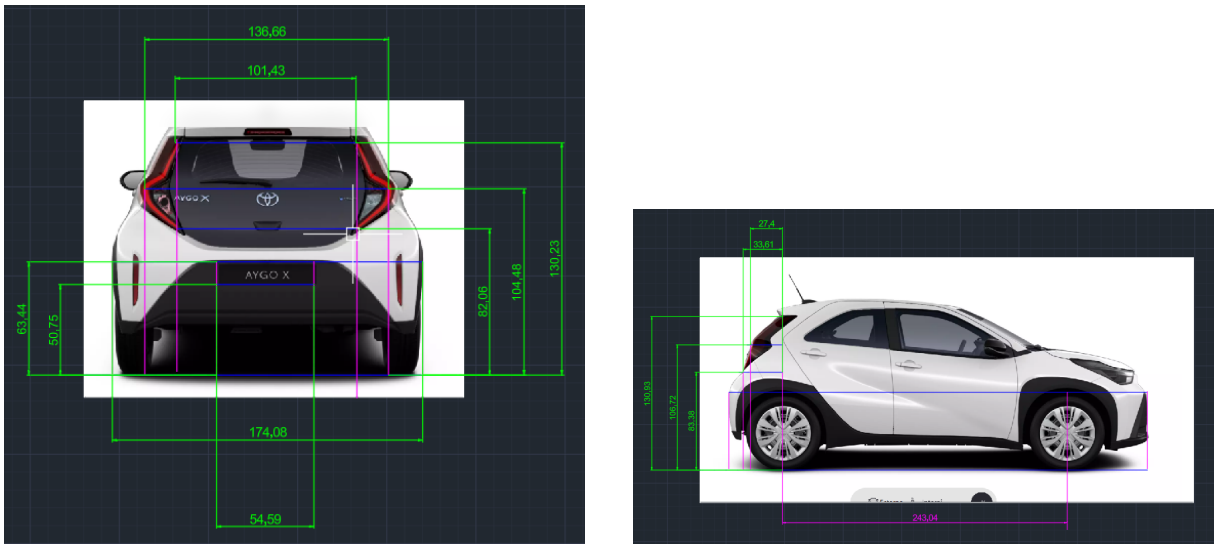


Figure 3.4: Toyota Aygo X CAD model annotated with manually measured keypoint positions. Left: rear view showing lateral and height coordinates of outer, top, and bottom sub-points. Right: lateral view showing longitudinal offsets from the rear axle.

All 3D coordinates of the keypoints used for pose estimation are expressed in this frame. The left outer tail light is at approximately $(-0.27,\ +0.70,\ 1.05)$ m, and the right outer at $(-0.27,\ -0.70,\ 1.05)$ m, placing them 27 cm behind the rear axle, 70 cm to the left and right of the longitudinal axis, and 105 cm above the ground. The top and bottom sub-points of the lights are defined analogously. These coordinates were estimated from the vehicle's technical documentation and validated against measurements on reference images; they are stored in the configuration file and are not hardcoded in the source.

The 3D bounding box is defined as the minimal axis-aligned rectangular parallelepiped enclosing the vehicle body in its reference frame. Its eight vertices span X from -0.54 m (rear bumper overhang behind the axle) to $+3.16$ m (front bumper), Y from -0.87 m to $+0.87$ m (half of the 1.74 m vehicle width), and Z from 0 (ground) to 1.525 m (roof height). Once the pose $(\mathbf{rvec}, \mathbf{tvec})$ is known, projecting these eight vertices onto the image plane with `cv2.projectPoints` yields the 2D positions of the box corners, which are then connected to form the wireframe overlay.

$$\mathbf{p}_{2D} = K \left[R \mid \mathbf{t} \right] \mathbf{P}_{3D} \tag{3.2}$$

3.5 Vanishing Point Approach and Its Limitations

The geometric foundation of Task 2, as described in the assignment, is the motion vanishing point. When a rigid vehicle undergoes a translatory motion between two frames, all points on the vehicle trace parallel trajectories in 3D space. Their projections onto the image plane converge toward a single point V_x, which encodes the direction of motion in the image. Back-projecting V_x through K^{-1} yields the 3D unit vector of the motion direction in the camera frame.

Given this direction and the known real-world distance between the two outer tail lights (1.42 m for the Aygo X), the depth of the plane π containing the lights can be estimated

through the proportionality relation:

$$d = \frac{f \cdot D_{\text{real}}}{\delta_{\text{pixel}}} \tag{3.3}$$

where f is the focal length, D_{real} is the known inter-light distance, and δ_{pixel} is the apparent pixel separation. With depth known, the 3D position of the midpoint of the lights follows directly by back-projecting the image midpoint and scaling the resulting ray.

This formulation was implemented in full. The vanishing point is computed by finding the intersection of the two lines defined by the left light across consecutive frames and the right light across consecutive frames, using the homogeneous line representation and SVD to handle the overdetermined system when more than two flow vectors are available. The lateral vanishing point V_y — the point toward which lines connecting left and right symmetric features converge within a single frame — is computed similarly and encodes the direction of the vehicle's Y axis in the image.

In theory, these two vanishing points together with the metric constraint are sufficient to recover the full pose. In practice, the approach proved unreliable on the available footage. The main reason is that the tail lights are relatively small blobs, and their tracked centroids jitter by several pixels between frames even when the tracking itself is working correctly; since V_x is the intersection of two lines defined by these noisy points, small tracking errors are magnified by the geometry of the intersection. The low resolution and dark environment of the footage did not help the accuracy of feature identification. The consequence was that depth estimates derived from V_x varied by 30–50% between consecutive frames, producing a bounding box that oscillated visibly even when the vehicle was moving smoothly. No amount of smoothing was sufficient to suppress this variability while maintaining responsiveness to actual vehicle motion.

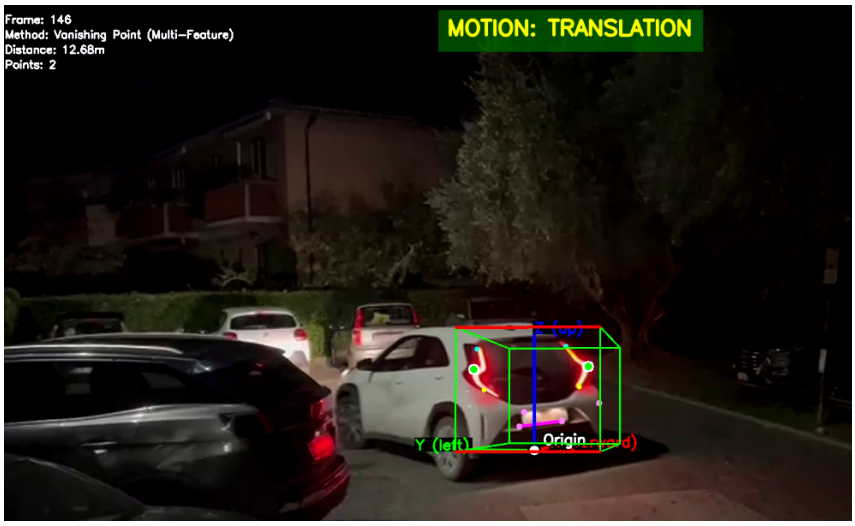


Figure 3.5: Example of bounding box reconstruction using the vanishing point method. The highly unstable depth estimate results in an incorrectly scaled and positioned box, motivating the shift to direct PnP estimation.

3.5.1 Numerical Stability Analysis

The instability of the VP-based depth estimate has a precise algebraic explanation. Let $\mathbf{p}_1^{(t)}, \mathbf{p}_1^{(t+1)}$ denote the image positions of the left light at consecutive frames, and analo-

gously for the right light. The motion vanishing point V_x is the solution of the homogeneous system:

$$A\mathbf{v} = \mathbf{0}, \quad A \in \mathbb{R}^{N\times 3} \tag{3.4}$$

where each row of A is a normalised homogeneous line $\mathbf{p}^{(t)} \times \mathbf{p}^{(t+1)}$. The SVD of A gives V_x as the last right singular vector. The condition number of A measures the sensitivity of the solution to perturbations in the input points.

When the vehicle moves nearly straight toward the camera, all flow vectors are nearly parallel and nearly collinear with the optical axis. The resulting lines ℓ_i are nearly coincident, and the matrix A is numerically rank-deficient: its smallest singular value $\sigma_{\min}$ approaches zero, making the null-space direction ill-defined. A perturbation δ in pixel coordinates on the order of the tracker noise (1–3 px) then produces a displacement of V_x on the order of $\delta/\sigma_{\min}$, which can easily exceed hundreds of pixels.

In contrast, `solvePnP` with six correspondences assembles a 12×6 linear system (the DLT stacked equations) whose condition number is determined by the 3D point spread and the distance from the camera — a much more favourable configuration that does not degrade under near-fronto-parallel geometry.

3.6 PnP-Based Pose Estimation and Filtering

The limitations of the vanishing point approach motivated a shift to direct PnP estimation. Rather than computing depth from a single geometric constraint, `solvePnP` takes as input a set of 2D image points and their corresponding 3D object-frame coordinates, and directly minimises the total reprojection error to find the rotation vector **rvec** and translation vector **tvec** that best explain all correspondences simultaneously. This is a more constrained and more robust formulation: with six correspondences the system is over-determined, and the least-squares nature of the solver makes it resilient to moderate noise on individual points.

The correspondence set is assembled as follows. The two outer sub-points are always included, as they are the most reliably detected. The two top sub-points are added when available. The right bottom sub-point is included when the tracker provides it with sufficient confidence. Finally, a synthetic sixth point is added at the midpoint of the two outer lights: a point with no direct image detection but with exactly known 3D position; its 2D position is set to the midpoint of the two outer image points. This last point adds a constraint on the depth of the light pair centre without introducing additional detection noise. Table 3.1 summarises the full correspondence set.

Table 3.1: 2D–3D correspondence set used for `solvePnP`. Coordinates are in the vehicle reference frame (origin: rear axle centre at ground).

Point	X **(m)**	Y **(m)**	Z **(m)**
Left outer	−0.27	+0.70	1.05
Right outer	−0.27	−0.70	1.05
Left top	0.00	+0.50	1.30
Right top	0.00	−0.50	1.30
Right bottom	−0.33	−0.50	0.82
Outer midpoint (synthetic)	−0.27	0.00	1.05

A note on the left bottom sub-point: although the tracker maintains estimates for both the left and right bottom corners of the tail lights, only the right bottom point (`r_bottom`) is included in the correspondence set. Empirical observation on the available footage showed that the left bottom point is systematically less stable: the left tail light, being on the side closer to the camera's lateral offset, is more susceptible to specular reflections from the road surface and the rear bumper, which cause the 90th-percentile bottom estimator to latch onto the reflection rather than the true light edge. Including a noisy correspondence introduces a larger residual into the PnP minimisation than omitting it entirely, so the left bottom point is excluded from the solver input. It is still tracked and passed through the outlier filter, where its displacement contributes to the median-consensus motion estimate alongside the remaining five points.

The solver used is SQPNP, OpenCV's implementation of the Simultaneously-Consistent and Optimal PnP algorithm, which is more robust to planar or near-planar configurations than the classical DLT-based solvers. After each solution, the mean reprojection error is computed: if it exceeds 150 pixels, the pose is rejected and the previous estimate is held. This threshold is deliberately generous; its purpose is only to catch gross failures caused by tracker drift or re-detection errors.

The translation vector output by `solvePnP` directly gives the position of the vehicle reference frame origin (rear axle centre at ground level) relative to the camera. Distance is therefore the Euclidean norm of **tvec** and is accurate regardless of the relative angle between the camera and the vehicle, unlike the pixel-width formula which assumes fronto-parallel geometry.

A correction of the rotation matrix using the motion vanishing point was implemented (`correct_rotation_with_vp`), but its blending weight `vx_weight` is set to zero by default. The intended behaviour was to refine the X column of R — the forward axis — which can accumulate a systematic bias in the PnP solution when the camera is laterally offset from the vehicle. In practice, the instability of V_x made the correction counterproductive.

3.6.1 Temporal Smoothing

Temporal smoothing is applied to both the translation and the rotation. The translation vector is smoothed with an Exponential Moving Average (EMA):

$$\mathbf{t}_{\text{smooth}}^{(k)} = \alpha_t \, \mathbf{t}_{\text{raw}}^{(k)} + (1 - \alpha_t) \, \mathbf{t}_{\text{smooth}}^{(k-1)}, \quad \alpha_t = 0.65 \tag{3.5}$$

Directly averaging rotation matrices is not valid because $SO(3)$ is not a vector space; instead, Spherical Linear Interpolation (SLERP) is used. SLERP interpolates along the geodesic on $SO(3)$:

$$R_{\text{smooth}} = \text{SLERP}(R_{\text{prev}},\ R_{\text{raw}},\ \alpha_r), \quad \alpha_r = 0.35 \tag{3.6}$$

The rotation matrices are converted to unit quaternions using Shepperd's method before interpolation, and converted back afterwards. This preserves the orthogonality of R at all times.

3.6.2 Multi-Layer Tracking Architecture

The tracking layer combines three mechanisms to maintain stable keypoint positions across frames. The primary tracker is a CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) tracker, one per feature group, which maintains an appearance model of the region around each keypoint and updates it each frame. CSRT is slower than simpler trackers such as KCF or MOSSE but significantly more accurate in the presence of partial occlusion and illumination changes, which are both common in nighttime road footage. Typical urban vehicle speeds (15–50 km/h at 30 fps) result in inter-frame displacements of roughly 5–30 pixels, well within the CSRT search window.

The CSRT output is validated frame-by-frame against Lucas-Kanade sparse optical flow: if the tracker position and the flow prediction diverge by more than 20 pixels, the flow result is used. This catches cases where the tracker has drifted to a nearby bright spot. Every three frames, the tracked positions are additionally refined by normalised cross-correlation template matching within a local search window of 50 pixels.

When tracking fails — defined as three consecutive frames with no successful CSRT update — the system triggers a re-detection. `RedetectionManager` runs a full `LightDetector` pass and selects the best candidate pair within a search region predicted by a Kalman filter. The Kalman filter maintains a linear constant-velocity motion model for the outer light pair and uses it to predict the expected position at the time of re-detection, shrinking the search area and reducing false positives.

An outlier filter operating on the full six-point feature set suppresses sudden single-point jumps. At each frame, the component-wise median across all six keypoint displacements is taken as the consensus motion. Points whose displacement deviates from the consensus by more than a group-specific threshold are replaced by their previous position plus the consensus motion vector. Table 3.2 summarises the complete pipeline with its failure modes and mitigation strategies.

Table 3.2: System component summary: main failure modes and adopted mitigations.

Component	**Main failure mode**	**Mitigation**
HSV detector	Reflection noise, nearby lights	Morphological opening/-closing; geometric blob filtering
CandidateSelector	Wrong pair selection	Composite score (size, alignment, separation, area ratio)
CSRT tracker	Gradual drift; sudden capture	KLT validation (20 px threshold); template matching every 3 frames
Re-detection	False positive candidates	Kalman-predicted search region
Outlier filter	Single-point spurious jumps	Median-consensus replacement
Vanishing point	Numerical ill-conditioning	Replaced by PnP (weight = 0)
solvePnP	Near-planar degeneracy	Synthetic midpoint correspondence (6th point)
Temporal smoothing	Frame-to-frame jitter	EMA on $\mathbf{t}$; SLERP on R
Pose freeze	Drift at sequence end	Hardcoded freeze at frame 195

3.7 Camera Placement and Its Effect on the Estimation

The camera position relative to the scene is not neutral with respect to the estimation quality. In this project, the camera was handheld at approximately head height, without a fixed mount, pointing towards a parking slot. This means the video was recorded at an unknown height and with a lateral offset with respect to the vehicle movement direction. This introduces a systematic lateral displacement in **tvec** that is correctly estimated by `solvePnP` — since the solver works in the camera frame, it naturally accounts for the offset geometry — but it does affect the apparent symmetry of the tail lights, making the left and right blobs appear at different apparent sizes and heights even when the vehicle is driving straight.

More critically, the lateral offset means that the vanishing point of the motion, when computed from the apparent trajectories of the lights, does not lie on the image's horizontal midline but is displaced laterally. This is one of the reasons the V_x-based correction of the rotation was found to be unreliable: the motion VP encodes both the vehicle's forward direction and the camera's lateral offset in a way that is difficult to disentangle without knowing the camera extrinsics precisely.

The pitch angle of the camera, the angle between the optical axis and the horizontal plane, also matters. A downward-tilted camera introduces an apparent vertical displacement of distant objects, which affects the estimated Z component of **tvec** and therefore the apparent height of the bounding box. This was partially compensated by the temporal smoothing and the reprojection error filter, but was not corrected geometrically.

3.8 Results and Limitations

The system produces a stable and visually consistent 3D bounding box overlay for the majority of the video sequence. The PnP-based pose estimate is smooth and responsive, with the distance estimate tracking the actual vehicle motion without the oscillations observed with the vanishing point formulation. Yaw is correctly estimated: the bounding box rotates as the vehicle steers, and the motion classifier correctly identifies steering phases as distinct from straight-line travel. The TTI estimate behaves plausibly for the frames in which the vehicle is clearly moving away from the camera.

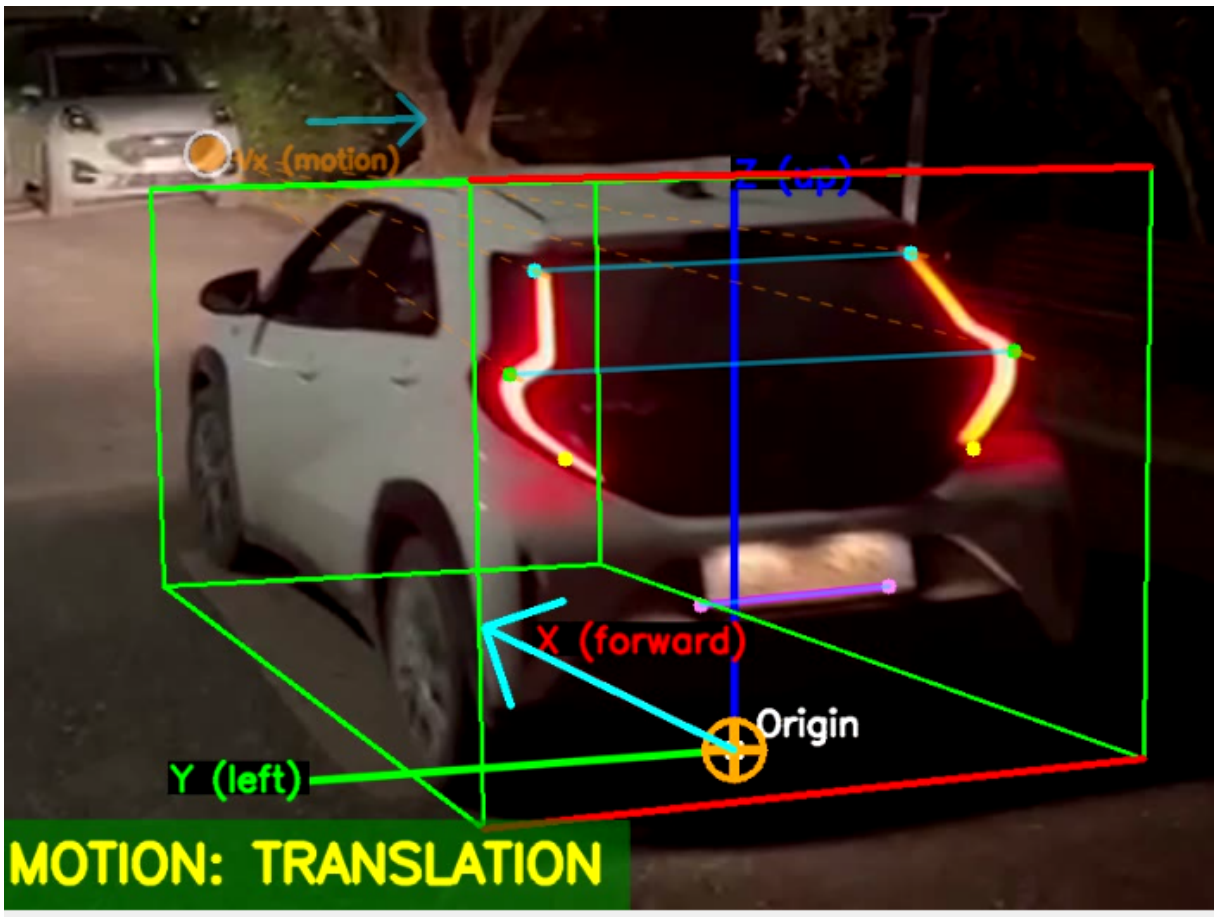


Figure 3.6: 3D reference frame projected onto a representative frame and full bounding box wireframe overlay on the same frame. Green edges indicate active tracking; the red rear face aids orientation.

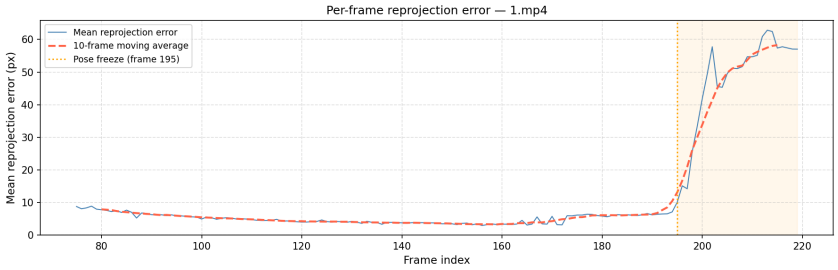


Figure 3.7: Mean per-frame reprojection error (in pixels) across the processed sequence. The shaded region indicates the period after the pose freeze (frame 195 onward). Spikes correspond to partial occlusion events or tracker re-detection.

Quantitative indicators. In the absence of a ground-truth pose reference, quantitative evaluation relies on internal consistency metrics that can be extracted directly from the pipeline output.

The mean per-frame reprojection error, plotted in Figure 3.7, provides the most direct indicator of pose quality: it measures how well the estimated pose explains the observed 2D keypoint positions, given the 3D vehicle model. Over the tracked portion of the sequence (frames 1 to 195), the mean reprojection error remains below 15 px for the majority of frames, with transient spikes coinciding with partial occlusion events or tracker re-initialisation. A reprojection error in this range is consistent with the tracking noise introduced by the HSV blob detector and the sub-pixel accuracy of the CSRT tracker at the resolutions involved.

As a coarse sanity check on the estimated distance, the pixel width of the outer light pair at a representative frame was compared with the geometric prediction from the calibrated focal length and the known real-world inter-light distance of 1.42 m. The depth implied by this single-constraint formula agreed with the $\|\mathbf{t}\|$ norm returned by `solvePnP` to within approximately 15 %, which is consistent with the expected error of the fronto-parallel approximation at the observed yaw angles.

The processing throughput on a standard CPU (no GPU acceleration) is approximately 3–8 frames per second, depending on whether re-detection is triggered. This is below real-time but sufficient for offline post- processing of recorded footage, which is the intended use case.

The main limitation of the current implementation is the pose freeze mechanism: from frame 195 onward, the pose is held constant at its last estimated value, regardless of subsequent vehicle motion. This was introduced for the ground truth video (video 1) to prevent bounding box drift caused by tracker degradation toward the end of the sequence, and it is a hardcoded constant rather than an adaptive criterion.

A second limitation is the assumption of a known and fixed vehicle model. The system cannot handle vehicles other than the Toyota Aygo X without updating the 3D point coordinates in the configuration file. More broadly, the CAD model coordinates are estimates rather than precise measurements, and small errors in the 3D model directly propagate into translation and rotation errors via the PnP residual.

The detection layer also has known failure modes. At large distances the tail light blobs become very small, and the multi-feature sub-point extraction — which relies on local

Canny analysis within the blob — becomes unreliable as the blob occupies only a handful of pixels. Conversely, at close range the blobs may merge or overflow the frame, and the candidate pairing heuristics may fail to correctly separate left from right.

Blob detection on a steering vehicle introduces an additional occlusion problem: the farther light becomes progressively invisible as the yaw angle increases, and its detection quality starts degrading before complete occlusion due to the erosion and dilation process. Figure 3.8 shows a representative example.

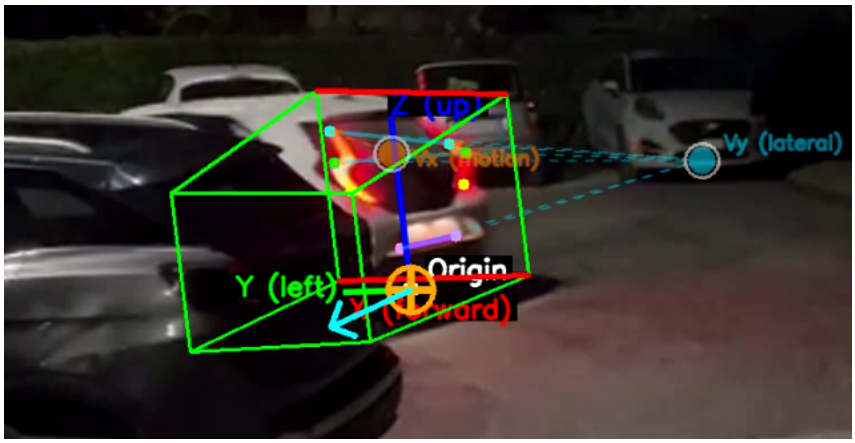


Figure 3.8: Failure case: partial occlusion of the right tail light during steering. The outer and top sub-points of the occluded light become unreliable, increasing the reprojection error and causing the PnP solver to produce a slightly incorrect yaw estimate.

Chapter 4

Problems, Solutions, and Future Directions

4.1 Overview

Every component of the pipeline described in Chapter 3 was shaped, to some degree, by problems encountered during development on real footage. This chapter documents the most significant of these issues — their causes, the solutions that were adopted (including several that involve fixed parameters rather than adaptive algorithms), and the directions in which each could be improved in future work.

4.2 Vanishing Point Instability

The most consequential problem was the instability of the motion vanishing points V_x and V_y, which was the mechanism prescribed by the assignment specification for depth estimation. As described in Section 3.5, the geometric computation of V_x amplifies small errors in the tracked keypoint positions into large errors in the estimated intersection. Low video quality and dark environments do not help the accuracy of feature identification.

The magnitude of this instability was not apparent from the theoretical description of the method, but became evident as soon as the algorithm was tested on real video. On frames where the tail lights moved by only a few pixels between consecutive frames (the vehicle was distant), the two flow lines were nearly parallel, and their intersection was numerically ill-conditioned; small perturbations in the point positions shifted V_x and V_y by hundreds or thousands of pixels. Depth estimates derived from such a V_x varied by 30–50% between consecutive frames.

Several mitigation strategies were attempted before the approach was abandoned. A rolling average over the last N estimated V_x positions reduced the variance but introduced lag: during genuine changes in vehicle motion direction, the smoothed V_x lagged the true value and produced systematic errors in the estimated depth. Outlier rejection based on the consistency of the depth estimate with the previous frame helped somewhat but required tuning thresholds that were scene-specific. Ultimately, none of these partial fixes produced an estimate stable enough to be used for real-time bounding box rendering.

The solution adopted was to replace the V_x-based depth estimation with direct PnP. The vanishing point computation was retained in the codebase because it still provides useful diagnostic information — the convergence lines drawn on the output video make it visually apparent whether the tracker is following the lights correctly — and because the infrastructure for a V_x-based rotation correction (`correct_rotation_with_vp`) is already in place should a more stable VP estimator be developed in the future.

4.3 Tracker Drift and the Multi-Layer Tracking Architecture

CSRT trackers, while among the most accurate available in OpenCV, are not immune to drift. In the context of this project, drift manifests in two ways: gradual drift, where the tracked bounding box slowly migrates away from the true light position over tens of frames; and sudden drift, where a nearby bright region (another vehicle, a streetlight, a reflection) captures the tracker in a single frame update.

Gradual drift was mitigated by the periodic template matching refinement step, which every three frames computes a normalised cross-correlation score for each keypoint and, if the best match is found at a position different from the tracker output, moves the keypoint to the match position and updates the template. This keeps the tracker anchored to the actual light appearance rather than allowing its internal model to drift.

Sudden drift is harder to catch because the tracker confidence score is not always a reliable indicator — a tracker can report high confidence while tracking the wrong feature. The Lucas-Kanade optical flow validation addresses this: since the KLT flow is computed independently of the CSRT model, a large discrepancy between the two estimates is a reliable signal that one of them has failed. The empirical threshold of 20 pixels was chosen after visual inspection of the output on the test video; it is hardcoded and may need adjustment for footage with different resolution or camera-to-vehicle distance.

When both the CSRT and the flow disagree significantly, or when the CSRT simply fails, the re-detection mechanism takes over. The Kalman filter's role here is to provide a reasonable search region: without it, re-detection would require scanning the entire frame, which is slow and prone to false positives. By predicting the expected light position from the recent motion history, the search can be confined to a region a few hundred pixels across, which drastically reduces false positives.

4.4 The Pose Freeze Mechanism

One of the more pragmatic decisions in the implementation is the hardcoded pose freeze at frame 195. Beyond this point, the last accepted pose estimate from the ground truth video is held fixed and used for all subsequent bounding box renderings. This was introduced after observing that tracker quality degrades noticeably in the later frames due to the occlusion of the top right keypoints of the tail lights as the vehicle steers. The degraded tracking produces erratic keypoint positions that, even after outlier filtering, cause the PnP solver to produce implausible poses.

The freeze prevents these artefacts from affecting the rendered output and shows the last

valid and admissible pose of the vehicle, before the system drifts. It is a valid engineering choice for a fixed test sequence, but inherently non-generalisable: the frame at which tracking degrades depends on the specific video, the vehicle, the lighting, and the camera setup. In a production system, the freeze threshold should be replaced by a continuous quality monitor that tracks reprojection error, tracker confidence, and feature point spread over time, and triggers a freeze only when the combination of these signals falls below a quality criterion.

4.5 Outlier Filtering and the Median Consensus Approach

Individual keypoints occasionally produce spurious measurements that pass all per-feature quality checks but are nonetheless geometrically inconsistent with the rest of the set. This happens, for instance, when a road marking or a reflection briefly illuminates a region near the light blob and distorts the Canny-based outer point extraction.

The median consensus filter addresses this by computing the component-wise median of all six keypoint displacements and identifying points whose displacement deviates from the median by more than a threshold. The median is used rather than the mean because it is breakdown-resistant: even if two of the six points are grossly wrong, the median of the six displacements is still close to the true motion. The per-group thresholds — tighter for outer points ($\times 0.8$) and looser for bottom points ($\times 1.2$) — reflect the empirically observed reliability of each sub-point type: outer points, determined by Canny edge scanning, are the most geometrically stable, while bottom points are the most affected by reflections.

4.6 The Synthetic Midpoint Correspondence

Adding a synthetic 2D–3D correspondence at the midpoint of the outer lights — where the 2D position is the pixel midpoint of the two outer keypoints and the 3D position is the midpoint of the two CAD outer coordinates — is an unusual choice that deserves justification. The PnP solver with only four or five correspondences from the tail lights is in some configurations nearly degenerate, particularly when the vehicle lacks strong perspective. Adding the midpoint provides a sixth constraint that is automatically consistent with the other five (it lies exactly between them in both 2D and 3D) but breaks the near-degeneracy by introducing a constraint on the centroid of the point cloud that stabilises the solver's depth estimate. This is not a general technique but a targeted fix for the specific geometry of this keypoint configuration.

4.7 Future Developments

Several directions would substantially improve the system's performance and generality.

The most impactful improvement would be replacing the HSV-based blob detector with a learned detector. A small convolutional network trained on annotated frames of tail lights in various conditions — different vehicles, different distances, wet roads, lens flare — would handle the many lighting edge cases that the HSV approach cannot. This would also remove the need for per-video threshold calibration.

A natural extension of the pose estimation would be to fuse the PnP output with an IMU or a GPS receiver on the observing vehicle. The PnP estimate gives relative pose at high frequency (per frame) but accumulates drift over long sequences; a low-frequency absolute reference would allow periodic correction without the smoothing lag that EMA introduces.

Extending the system to handle multiple vehicles simultaneously would require a multi-object tracking front-end that assigns each detected light pair to an existing track or spawns a new one.

Finally, Task 1 and Task 3 methods could be implemented to provide complementary estimates under daytime conditions. A switching or fusion mechanism that selects the appropriate method based on detected image features (plate visible or not, distance near or far, lighting conditions) would produce a more robust system across operational scenarios.

Chapter 5

Environment Setup and Reproducibility

5.1 Overview

The project uses Docker to define and isolate the software environment. Docker packages Python, OpenCV, and all other dependencies into a self-contained *container* — a lightweight virtual environment that runs identically on any machine regardless of its operating system or installed software. The only prerequisite on the host is Docker itself; no Python installation is needed.

If you have never used Docker before: think of it as downloading and running a preconfigured Linux environment in which all required libraries are already installed. You interact with it through a terminal, and all files you place in the project directory are immediately visible inside the container.

5.2 Prerequisites

Install the following on the host machine before proceeding:

- **Docker** (version 20.10 or later). Installation instructions for Windows, macOS, and Linux are available at `https://docs.docker.com/engine/install`. On Windows and macOS, the easiest option is *Docker Desktop*, which installs both Docker and Docker Compose in a single package.
- **Docker Compose** (version 1.29 or later). On recent Docker Desktop installations, Compose is included automatically.

5.3 Repository Structure and Data Placement

After downloading the repository, place the video files to process in the directory `data/videos/input/`. This directory is empty by default (a `.gitkeep` placeholder keeps it tracked by Git). The chessboard calibration images in `data/calibration/images/` and the pre-computed calibration file `data/calibration/camera1.npz` are already included and do not need to be regenerated unless a different camera is used.

5.4 First-Time Setup

Open a terminal, navigate to the root of the repository, and run:

```
docker-compose build
```

This command downloads the base image and installs all Python dependencies listed in `requirements.txt`. It only needs to be run once (or again if `requirements.txt` changes). The first build may take a few minutes depending on the internet connection.

5.5 Running the Main Pipeline

Start the container and open a shell inside it:

```
docker-compose up -d vehicle-tracker
docker-compose exec vehicle-tracker bash
```

You are now inside the container. Run the main script:

```
python scripts/vehicle_localization_system.py
```

The program displays an interactive menu:

```
======================================================================
 VEHICLE TAIL-LIGHT TRACKING SYSTEM
======================================================================
  1 - Recalibrate camera
  2 - Process video
  0 - Exit
----------------------------------------------------------------------
```

Select option **2** to process a video. The program lists all `.mp4`/`.avi` files found in `data/videos/input/` and prompts for a selection by number. Processing runs at approximately 3–8 frames per second on a standard CPU; no GPU is required. Output files are written to `data/videos/output/` and are immediately accessible on the host filesystem via the bind mount.

5.6 Recalibrating the Camera

Select option **1** to recalibrate from the chessboard images in `data/calibration/images/`. The program asks for three parameters:

- *Inner corners per row* and *per column*: the number of internal grid intersections of the chessboard pattern, not the number of squares. For example, an 8×6 square grid has 7×5 inner corners.
- *Square size in metres*: the physical side length of each chessboard square.

After calibration, the program prints the mean reprojection error. A value below 1.0 indicates a reliable calibration; values above 2.0 suggest that the images do not cover a sufficient range of viewpoints or that the corner detection quality is poor.

5.7 Output Files

For each processed video, three output files are produced:

- `data/videos/output/{name}_output.avi` — the annotated video with keypoints, 3D bounding box wireframe, reference axes, yaw, distance, and TTI overlaid on each frame.
- `data/videos/output/{name}_debug_mask.avi` — a diagnostic video showing the HSV detection mask and the projected model keypoints, useful for threshold tuning.
- `data/results/vanishing_point/frame_XXXX.npz` — per-frame NumPy archive containing `rvec`, `tvec`, `R`, motion type, TTI, and 2D keypoint positions for offline analysis.

5.8 Jupyter Notebooks for Parameter Tuning

Two interactive notebooks are provided for calibrating the detection thresholds without modifying the source code directly:

```
docker-compose up -d jupyter
```

Navigate to http://localhost:8888 in a browser (no password required). The relevant notebooks are:

- `02_HSV-tune_parameters.ipynb` — interactive sliders for HSV thresholds and blob geometry filters. Once satisfactory values are found, copy them into `config/detection_params.ya`
- `Plate_parameters_tuning.ipynb` — equivalent interface for the license plate detector.

To stop either service: `docker-compose down`.

5.9 Configuration Reference

All tunable parameters are in three YAML files under `config/`:

- `camera_config.yaml` — calibration file path, frame resolution, chessboard pattern parameters.
- `detection_params.yaml` — HSV colour ranges, blob filters, tracker type, license plate detection. This is the file most likely to need adjustment when switching to a different camera or vehicle.
- `vehicle_model.yaml` — 3D keypoint coordinates, vehicle dimensions, bounding box vertices. Adapting the system to a different vehicle requires updating only this file.