CHALMERS
UNIVERSITY OF TECHNOLOGY

Lane Detection / LDW project

Prakash Raju Sridharraju | prakashr@student.chalmers.se

February 21, 2026

Introduction

This project develops a vision-based lane detection and Lane Departure Warning (LDW) pipeline using a monocular camera and classical image-processing methods. The system performs camera calibration and image undistortion, applies a perspective transform to obtain a bird's-eye view of the road, and uses HLS colour thresholds together with Sobel gradients to isolate lane markings. Histogram-based initialisation and sliding-window tracking are then used to follow left and right lane boundaries, while second-order polynomials provide a compact mathematical representation of each lane line. Solid versus dashed markings are classified based on the vertical distribution of lane pixels with temporal smoothing, and lane-edge distances are converted from pixels to metres to trigger LDW warnings when the robot approaches solid lane boundaries. The complete pipeline is integrated into a ROS 2 node and evaluated both on recorded road video and in real time on a lab robot platform.

Active safety Context

Unintended lane departures are a major contributor to single-vehicle crashes and head-on collisions, making lane-keeping assistance and LDW key functions within modern Advanced Driver Assistance Systems. Production LDW systems typically rely on forward-looking cameras to track lane markings and compute the vehicle's lateral position relative to the current lane, issuing warnings when the vehicle drifts towards a boundary without an intentional lane change. This project mirrors that architecture on a smaller scale by combining calibrated camera sensing, robust lane-mark segmentation, and geometric lane modelling to estimate boundary distances in real time and generate legally aware warnings only for solid lines. By implementing the full chain from raw images to lane-edge distance and warning logic in ROS 2, the project illustrates how vision-only perception can be engineered into a practical active safety function, while also highlighting limitations related to lighting, worn markings, and road geometry that are critical for real-world deployment.

Keywords: Time-to-Collision, rear-end collision, Forward Collision Warning, Autonomous Emergency Braking, driver behaviour analysis

Camera Calibration, Image Pre-Processing and Feature Extraction

Camera intrinsic calibration and undistortion

The first step in the project was to find the corners of the chessboard in each image. Using these points, we computed the camera's intrinsic matrix and distortion coefficients. For this purpose, we used the provided chessboard images and defined an `undistort` function to return an undistorted image.

Undistorted means correcting an image so that lens distortions, including radial and tangential distortions, are removed. As a result, straight lines in the real world appear straight in the image. We used OpenCV's `cv2.undistort` function to achieve this correction.

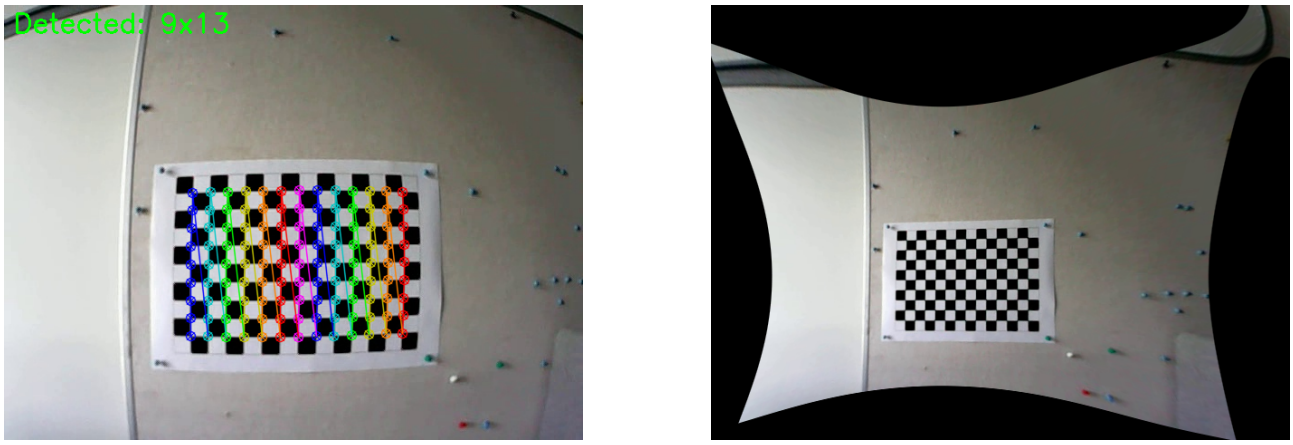
The resulting camera matrix and distortion coefficients obtained from calibration are as follows:

$$\mathbf{K} = \begin{bmatrix} 486.23 & 0 & 270.10 \\ 0 & 486.13 & 276.52 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{D} = [-0.4650, 0.1950, 0.0008, 0.0054, -0.0324]$$

These parameters allow accurate correction of lens distortion, ensuring that images reflect the true geometry of the scene.

Figure 4a shows an example chessboard image with the detected corners highlighted. These corners are used to compute the camera calibration parameters.



(a) Chess board corners

(b) Undistorted

Figure 1: Chessboard image with detected corners used for camera calibration and undistorted image

Perspective transform design and bird's-eye view generation

The next step was to perform a **perspective transform**. This involves mapping a trapezoidal region of interest in the original road image to a rectangular "bird's-eye view," where lane lines appear vertical and parallel. This transformation simplifies lane detection and allows accurate measurement of distances on the road.

Source and Destination Points

We manually selected the source points (`src`) to define the trapezoid containing the lane lines:

$$\begin{bmatrix} \text{Top-left} \\ \text{Top-right} \\ \text{Bottom-right} \\ \text{Bottom-left} \end{bmatrix} \quad \text{src} = \begin{bmatrix} 208 & 210 \\ 300 & 210 \\ 520 & 285 \\ 0 & 285 \end{bmatrix} \quad \text{dst} = \begin{bmatrix} 100 & 0 \\ w-100 & 0 \\ w-100 & h \\ 100 & h \end{bmatrix}$$

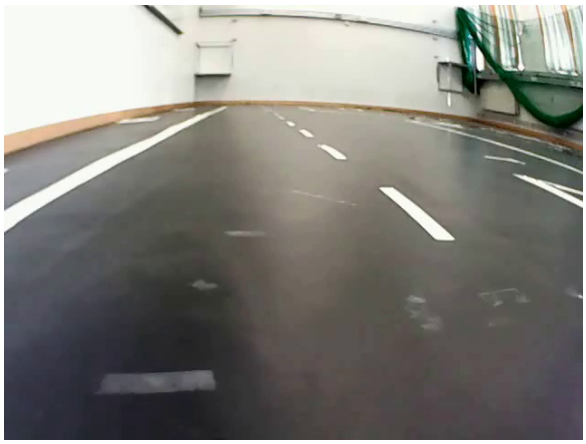
The top points were chosen to match the apparent convergence of lane lines in the distance, while the bottom points were wider to include the lanes near the vehicle. These points were mapped to rectangular destination points (`dst`) to form the bird's-eye view:

Justification

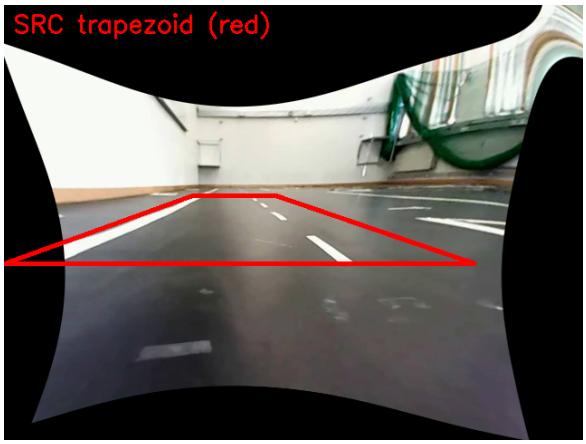
The trapezoid was selected to focus on the lane region, ensuring that lane lines are correctly captured while minimizing distortion from surrounding areas. By mapping this region to a rectangular bird's-eye view, the lane lines become vertical and parallel, facilitating easier detection and measurement.

Visualization

Figure 2b shows the chosen trapezoid drawn on the original video frame (2a), the resulting warped image (2c) and the binary mask (2d). This confirms that the lane lines have been transformed into a vertical, parallel orientation suitable for further lane detection.

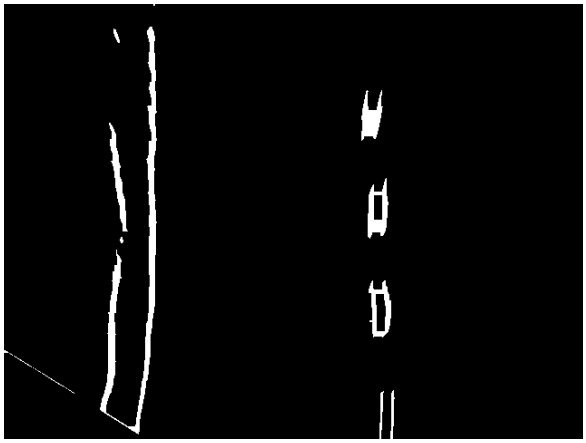


(a) Sample frame



(b) original with trapezoid

(c) Bird's eye view warp



(d) Binary conversion

Figure 2: video/frame transition step by step.

Binary filtering pipeline (HLS and Sobel) definition

- The binary conversion pipeline converts a color road image into a black-and-white mask that highlights only the lane markings.
- The image is analyzed through three parallel filters
  - L-channel (Lightness) in HLS color space: Detects white lanes by selecting bright pixels with values between 150–255.
  - S-channel (Saturation) in HLS color space: Detects yellow lanes by identifying highly saturated pixels with values between 90–255, even under shadows.
  - Sobel gradient filter: Applied to the grayscale image to detect vertical edges, measuring horizontal intensity changes. Strong gradients above 25 indicate lane boundaries.
- A pixel is marked as part of a lane only if it passes both the color test (bright white or saturated yellow) and the edge test (strong vertical gradient).
- This combined logic helps eliminate false positives such as sky, traffic signs, or road texture.

Why this works

- The method combines color and geometric validation, ensuring detected lanes are both visually consistent (via HLS color filtering) and structurally accurate (via Sobel edge detection).
- Using the HLS color space makes detection stable under varying lighting, while morphological closing (5×5 kernel) connects dashed lines and removes small noise for smooth, continuous lane markings.
- Fixed thresholds (L=150, S=90, Sobel=25) are optimized for typical brightness contrasts between lane markings and asphalt, making the system robust across weather and lighting conditions without needing adaptive algorithms.

Lane Boundary Detection, Tracking and Lane Departure Estimation

After creating the binary lane mask (see Task 1), the next step was to develop an algorithm that can identify the lane boundaries, classify them as solid or dashed, and trigger lane departure warnings when the vehicle approaches solid lane markings. The system processes video frames in real-time to provide continuous monitoring of the vehicle's position within its lane.

First, we used a sliding window search algorithm combined with histogram analysis. The histogram was computed by summing pixel values along each column of the bottom half of the binary image. This revealed peaks corresponding to the base of the left and right lanes. Starting from these base points, rectangular sliding windows were moved upward along the image to detect lane pixels. If enough pixels were found within a window, the next window was recentered to follow the lane. This process allowed the algorithm to track the lanes even if they were curved, partially faded, or broken. The sliding windows effectively act like two vertical rulers that follow the lanes, capturing all relevant pixels for further modeling.

Once the lane pixels were identified, a second-degree polynomial was fitted to each lane line, producing smooth continuous curves. The polynomials are defined as:

$$x_{\text{left}}(y) = a_l y^2 + b_l y + c_l, \quad x_{\text{right}}(y) = a_r y^2 + b_r y + c_r$$

These equations provide a mathematical representation of the lanes, allowing accurate calculation of the vehicle's lateral position relative to each lane boundary. For video sequences, temporal smoothing was applied by maintaining a history of the last 10 frames' polynomial fits, which were averaged to reduce noise and prevent sudden jumps in lane detection.

Lane classification into solid or dashed types was performed by analyzing the vertical distribution of detected lane pixels. The algorithm divides the image height into 20 equal bins and counts how many bins contain lane pixels. If more than 70% of bins are active (contain pixels), the lane is classified as solid; otherwise, it is classified as dashed. This classification undergoes temporal consensus filtering, requiring 60% of the last 10 frames to agree before the final classification is output. This prevents flickering between solid and dashed states due to momentary detection errors.

The Lane Departure Warning (LDW) system calculates the lateral distance from the vehicle center to each lane boundary by evaluating the polynomial at the bottom of the image (closest to the vehicle) and converting pixel distance to meters using the calibrated scale factor (3.7m lane width / 640 pixels = 0.0058 m/pixel). A warning is triggered only when:

- The lane is successfully detected (polynomial fit exists)
- The lane is classified as Solid (not dashed, which is legal to cross)
- The lateral distance is less than 0.20 meters (20 cm threshold)

The system issues directional warnings ("LDW: DRIFTING LEFT" or "LDW: DRIFTING RIGHT") displayed as red text overlay on the video output. This design ensures warnings are only issued for illegal solid lane boundary violations, not for intentional lane changes across dashed lines.

Strengths

- Robust to partial lane visibility due to independent left/right lane processing; one lane can fail without disabling the entire system..

- Temporal smoothing eliminates detection flickering and produces stable, reliable lane classifications across consecutive frames.
- Efficient sliding window reuses previous frame's polynomial fits as starting points, significantly reducing computational cost for video sequences.
- Legally-aware warnings that distinguish between solid (illegal to cross) and dashed (legal to cross) lane markings, preventing false alarms during lane changes.
- Metric-based thresholds using real-world distances (meters) rather than pixels, ensuring consistent warning behaviour regardless of image resolution or perspective.

Weaknesses

- Requires both lane markings visible for full functionality; heavily worn, occluded, or absent lanes degrade performance.
- Can be affected by road artifacts such as tar patches, shadows, construction zone markings, or reflective road studs that create false positives in the binary mask.
- Quadratic polynomials may not capture sharp curves or S-curves accurately; higher-degree polynomials or spline fitting would be needed for complex road geometry.
- Fixed perspective transform assumes flat road; steep hills or elevation changes can distort the bird's eye view and introduce distance measurement errors.
- No explicit vehicle dynamics modelling; the system does not account for vehicle speed, steering angle, or yaw rate, which could improve warning timing and reduce false alerts during normal cornering.

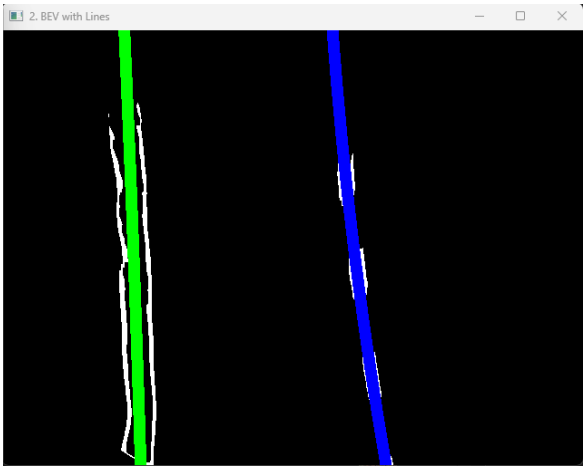


Figure 3: Example binary image of lane markings used for lane detection and polynomial fitting.

Lane Departure Estimation

After detecting and modeling the lane lines, the next step was to estimate the vehicle's lateral position relative to each lane boundary independently. This allows the system to monitor proximity to solid lane markings and issue warnings when the vehicle approaches boundaries that are illegal to cross.

The vehicle position estimation works as follows. Using the polynomial fits of the left and right lane lines, the x-coordinates of both lanes at the bottom of the image (closest to the vehicle, y = 479 pixels) are calculated by evaluating the polynomial equations:

$$x_{\text{left}} = a_l(479)^2 + b_l(479) + c_l$$
$$x_{\text{right}} = a_r(479)^2 + b_r(479) + c_r$$

Assuming the vehicle is at the horizontal center of the image, the lateral offset in pixels to each boundary is:

$$\text{offset}_{\text{left,pix}} = x_{\text{left}} - x_{\text{vehicle,center}}$$
$$\text{offset}_{\text{right,pix}} = x_{\text{right}} - x_{\text{vehicle,center}}$$

These pixel offsets are converted to real-world meters using the calibrated scaling factor:

$$\text{offset}_{\text{left,m}} = \text{offset}_{\text{left,pix}} \times x_{\text{m-per-pix}}$$
$$\text{offset}_{\text{right,m}} = \text{offset}_{\text{right,pix}} \times x_{\text{m-per-pix}}$$

where

$$x_{\text{m-per-pix}} = \frac{3.7}{640} = 0.0058 \text{ m/pixel.}$$

A threshold of 0.20 meters was chosen for safety. The lane departure warning logic operates independently for each boundary:

- Left Lane Warning: Triggered when the vehicle approaches within 0.20m of a solid left lane boundary (using absolute value since left offset is negative)
- Right Lane Warning: Triggered when the vehicle approaches within 0.20m of a solid/dashed right lane boundary

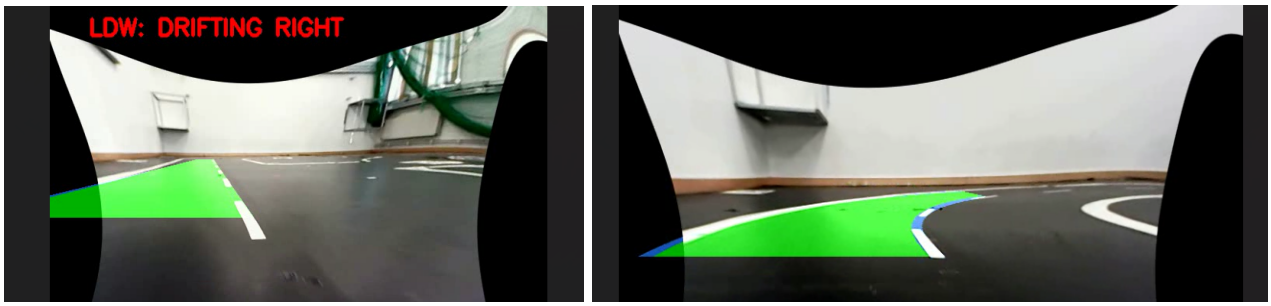
Warnings are displayed in red text overlaid on the video output, indicating which direction the vehicle is drifting. Crucially, warnings are issued based on lane markings and the position of the robot, ensuring the system respects legal lane-change zones and prevents false alarms during intentional lane changes.

Lane classification temporal smoothing is applied to the solid/dashed determination (maintaining a 10-frame history with 60% consensus requirement), which prevents classification flickering. However, the lateral distance measurements themselves are updated each frame without temporal filtering, ensuring the system responds immediately to actual changes in vehicle position.

This system is robust and practical because:

- Each lane is evaluated independently, so the detection failure of one lane does not disable warnings for the other
- Works effectively with partial lane visibility (e.g., one lane occluded by vehicles or worn markings)
- Direct boundary distance measurement is more intuitive and reliable than lane-centre-based approaches
- No exponential smoothing on distance prevents lag in warning response during rapid lateral drift

This system is robust and practical. It handles both straight and curved roads, works with typical variations in lane width, and adapts to partial lane visibility. By independently monitoring each lane boundary and providing immediate warnings, it helps prevent unintentional lane departures, which is crucial for driver assistance or autonomous vehicle applications. The combination of accurate lane detection, polynomial modelling, independent boundary distance calculation, and solid-lane-only warnings creates a fully integrated lane departure warning module that is reliable and legally aware.



(a) LDW visual (b) Polyfit function

Figure 4: Demonstration of our lane departure algorithm: straight lane (left) and curved lane (right).

Real-Time ROS 2 Integration and Embedded Implementation

We integrated the full lane pipeline into a ROS2 node ($image-process.py : LaneDepartureNode$). Each camera frame is undistorted with $camera-params.npz$, warped to bird's eye using a fixed homography, thresholded in HLS + Sobel to build a binary mask, and processed with a sliding window search to fit a line for left and right lanes. A 10-frame history classifies each line as solid or dashed. At the bottom of the image, we compare

the car centre to lane edges; if the car is within 0.20 m of a solid line, we publish a BuzzerState (short burst, repeated), we tested on a taped "lane" indoors: the node detected both lines, tracked moderate curves, and the buzzer sounded when the robot approached a solid/dashed edge. The biggest challenges were lighting/reflective floors (fixed thresholds), camera angle drift (warping errors). To keep the system running in real time on the Raspberry Pi, we drove the robot slower than its maximum speed; this appears to be a computational limitation of the software or the structure of the algorithm.

Images(5) of the demo day are added in the appendix, and videos of demo day and self-try in my apartment are shared in the folder

System Evaluation, Limitations and Future Improvements

Operating conditions with LDW performance

Performs Well

- **Highway driving with clear lane markings:** The algorithm performs best on highways with continuous, well-maintained white or yellow lines. The binary thresholding effectively highlights bright markings against dark asphalt.
- **Moderate curves:** Second-degree polynomial fitting accurately models gentle to moderate road curvature, ensuring reliable lane tracking on typical highway bends.
- **Good lighting conditions:** During daytime with uniform lighting and minimal shadows, the L-channel and S-channel thresholds detect lane markings consistently without introducing false positives.
- **Solid lane markings:** Temporal smoothing using a 10-frame consensus effectively differentiates solid from dashed lines, ensuring stable classification for warning decisions.
- **Stable camera mounting:** With a fixed camera position relative to the vehicle, the perspective transform remains accurate throughout operation.

Fails or Performs Poorly

- **Sharp curves and S-curves:** The second-degree polynomial cannot model very tight or complex lane geometry accurately, leading to significant approximation errors and incorrect lateral distance estimates.
- **Faded or worn lane markings:** Old or poorly maintained roads with low-contrast markings fall below intensity thresholds ($L : 150{-}255$, $S : 90{-}255$), causing incomplete or failed detection.
- **Heavy shadows and lighting transitions:** Sudden changes in brightness from bridges, trees, or overpasses confuse the thresholding process, producing false lanes or missing real ones.
- **Nighttime and adverse weather:** Conditions such as rain, fog, snow, or low light degrade image quality. Reflections on wet roads create false lane detections, while poor visibility prevents proper identification.
- **Construction zones:** Temporary or overlapping markings (old and new) cause confusion, as the algorithm cannot differentiate valid from invalid lane boundaries.
- **Multi-lane scenarios:** The system assumes exactly two lane boundaries (left and right of the vehicle) and fails in cases with more than two visible lanes.
- **Hills and banked curves:** The perspective transform assumes a flat surface; steep or banked roads distort the bird's-eye view and lead to incorrect distance calculations.

Failure modes and environmental sensitivities

Limitation 1: No Predictive Capability

The system is purely reactive, estimating only the current lateral position without accounting for vehicle dynamics such as speed, steering angle, or lateral velocity. At highway speeds (100 km/h $\approx$ 28 m/s), even a small reaction delay of 0.2 s means the vehicle travels approximately 5.6 m before the driver responds—potentially too late to prevent lane departure on curves or during inattentive driving.

Limitation 2: Environmental Sensitivity Without Self-Awareness

The algorithm lacks self-assessment capability for its own detection confidence. In adverse conditions (rain, fog, worn markings), it may issue false warnings or fail silently without notifying the driver that the LDW is inactive. This creates a dangerous false sense of security, as the driver might unknowingly rely on a non-functional system.

Limitation 3: Fixed Calibration and Scaling

All distance calculations depend on a fixed scaling factor:

$$X_{\text{m-per-pix}} = \frac{3.7}{640} = 0.0058 \text{ m/pixel.}$$

This calibration assumes a standard lane width and specific camera height and angle.
If the mounting configuration or road geometry (e.g., hill, banking) changes, the scaling becomes inaccurate. Consequently, the system may trigger false warnings or miss genuine departures, requiring manual recalibration for each setup.

Impact of fixed perspective transform on non-flat roads

The perspective transformation uses a homography matrix M to map four points from the camera view to a bird's-eye view. This mapping assumes all points lie on a single flat plane, the road surface. However, on hilly or uneven roads, this assumption fails.

Uphill Sections

When the road slopes upward, the actual surface lies above the assumed flat plane. The transformation still projects lane points as if they were at ground level, causing distant lane markings to appear compressed in the bird's-eye view. This leads to an underestimation of lateral distance, as the polynomial fit assumes points are farther than they actually are.

Downhill Sections

On downhill slopes, the road surface lies below the assumed plane. The perspective transform stretches the lane markings, making them appear wider apart. This results in overestimation of lateral distances and can suppress valid warnings.

Banked Curves

On banked roads, left and right lanes exist at different elevations. The homography matrix cannot handle this 3D variation, producing asymmetric distortions in the bird's-eye view.

Mathematical Consequence

The lateral offset is computed as:

$$\text{offset}_{\text{meters}} = (x_{\text{lane}} - 320) \times \frac{3.7}{640}.$$

This assumes pixel distance directly maps to real-world distance. On sloped terrain, this correspondence is violated; a lane marking appearing 100 pixels from the center could actually be 0.5 m closer or farther in 3D space, introducing systematic error into the warning logic.

Real-World Impact

On mountainous or uneven roads, the system alternates between over- and under-sensitivity, making lane warnings unreliable. With geometric errors of up to $\pm 10 - 20$ cm, a 20 cm departure threshold becomes meaningless. This inconsistency may cause drivers to ignore genuine warnings, reducing overall trust in the system.

IMU-assisted adaptive perspective correction

Proposed Improvement

Integrate an Inertial Measurement Unit (IMU) consisting of a 3-axis accelerometer and 3-axis gyroscope to dynamically adjust the perspective transformation matrix based on real-time vehicle pitch (forward/backward tilt) and roll (side-to-side tilt) angles.

Technical Implementation

- **IMU Measurements:** The IMU continuously measures vehicle orientation in 3D space:
  - Pitch angle (θ): Forward tilt on hills (positive uphill, negative downhill)
  - Roll angle (ϕ): Lateral tilt on banked curves (positive right bank, negative left bank)
- **Dynamic Homography:** Instead of using a fixed homography matrix M, compute an adaptive matrix $M(\theta, \phi)$ for each frame that compensates for non-flat road geometry.
- **Extract Pitch and Roll:**

$$\theta = \tan^{-1}\left(\frac{a_y}{a_z}\right), \quad \phi = \tan^{-1}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right)$$

- **Adjust Destination Points:**
  - For uphill (positive θ), the top edge of the BEV trapezoid shifts vertically.
  - For banked curves (non-zero ϕ), horizontal positions adjust asymmetrically.
- **Recompute Homography:**

$$M(\theta, \phi) = \text{cv2.getPerspectiveTransform}(src\_pts, dst\_pts_{adjusted})$$

  where $dst\_pts_{adjusted} = f(\theta, \phi)$ accounts for 3D road geometry.

Addressing Specific Limitation

This improvement removes the fixed perspective limitation by making the BEV geometrically accurate regardless of slope or banking. For a 10° uphill grade, the uncorrected system shows ≈ 15 cm lateral error at 30 m ahead, while IMU correction reduces this to under 3 cm.

From LDW to Lane Keeping Assist: functional upgrades

Lane Departure Warning is a passive system that alerts the driver during unintended lane departures but does not control the vehicle. Lane Keeping Assist is an active system that provides corrective steering inputs to maintain lane centring.

Key Comparison

- LDW: Are we leaving the lane? (binary detection)
- LKA: How far from the lane center are we, and what correction is needed? (continuous control)

Additional Hardware Requirements

- Electric Power Steering
- Steering Angle Sensor
- Vehicle Speed Sensor Integration
- Driver Monitoring System
- Enhanced Processing Hardware

Additional Software Components

1. Predictive Path Planning
2. Lateral Control Algorithm (PID Control)
3. State Machine Modes: Inactive, Standby, Active, Warning, Handoff, Fault.
4. Torque Limiting and Safety Monitoring:
5. Human-Machine Interface:
   - Dashboard indicators: Off / Standby / Active
   - Progressive alerts for driver inattention
   - Adjustable assistance levels (gentle / moderate / assertive)

Conclusion

- A complete camera-only lane detection pipeline was implemented: calibration, perspective transform, HLS + Sobel binary filtering, sliding-window tracking and polynomial lane modelling, all running in real time on ROS 2.
- The system can distinguish solid and dashed markings and trigger lane-departure warnings based on the vehicle's lateral distance to solid lane boundaries, making the warnings legally aware and less annoying for the driver.
- Temporal smoothing and metric-based thresholds (metres instead of pixels) significantly improve stability and interpretability of lane detection and warnings across video frames.
- The approach works well on roads with clear markings and moderate curvature but degrades under sharp curves, worn or occluded lines, strong shadows, and non-flat road geometry, highlighting key limitations of vision-only LDW.
- Integrating IMU-based pitch/roll compensation and extending the system towards Lane Keeping Assist would address several limitations and move the prototype closer to production-style active safety functions.

Appendix

Figure 5: μlab Robot Demo