Self Driving Challenge 2025
Visual Monocular Localization using a Map and Lane Lines

Thimo Veldema

Submisstion date: 22-07-2025

This paper presents a monocular localization system developed for the 2025 Self Driving Challenge, using a single forward-facing camera and a pre-defined map. The camera feed is transformed into a bird's-eye view, from which lane lines are extracted using HSV filtering. Vehicle motion is estimated by matching features on the road surface across consecutive frames. These motion estimates, combined with observed lane lines, are used in a particle filter to localize the vehicle. The resulting particle distribution provides pose estimates, which are fused with odometry data using a Kalman filter to produce a refined vehicle position and orientation. On straight road segments, the system achieves a mean absolute rotational error of 0.028 radians and a lateral error of 7 cm. While not highly precise, the system offers robust and reliable localization performance suitable for the requirements of the Self Driving Challenge.

Visual Odometry; Localization; Self Driving; Monocular

1. Introduction

The development of self-driving cars continues to accelerate. However, this progress must be approached responsibly. Autonomous vehicles need to operate safely under a wide range of conditions and respond correctly to dynamic traffic situations.

In the Netherlands, the RDW (Netherlands Vehicle Authority) is responsible for assessing the roadworthiness of vehicles[1]. Despite its central role in vehicle regulation, RDW acknowledges that self-driving technology presents new challenges beyond traditional automotive testing. To address this gap and gain practical experience, RDW organizes the Self Driving Challenge, an annual event initiated in 2019.

Each year, the challenge evolves in complexity. The 2025 edition focused on navigating a closed test track while demonstrating several key autonomous behaviors: obeying traffic signals, yielding to pedestrians, overtaking a stationary vehicle, and following a predefined route [2].

A crucial component in accomplishing these tasks is localization, the vehicle's ability to determine its position on the track. This year's challenge placed increased importance on localization, as the course featured more complex turns and decision points compared to previous years. In earlier editions, vehicles could largely rely on lane markings for guidance, as the tracks were simpler. While lane changes and basic maneuvers were tested, they did not require navigation decisions such as choosing between left or right paths.

To meet the demands of the 2025 challenge, localization techniques had to be implemented. The goal of this project was as follows: in the span of 4 months develop and implement a localization system that can be used for the self driving challenge. The research question is as follows:

How can monocular visual odometry and lane-line detection be combined to achieve accurate and robust vehicle localization using only one camera on the self driving challenge track?

2. Literature Review

2.1. Visual Odometry

Monocular visual odometry estimates a vehicle's motion using only input from a single camera. Existing approaches typically fall into two categories based on camera orientation: downward-facing and forward-facing methods.

Downward-facing approaches often rely on detecting keypoints, and calculating the transformation between consecutive images. For example, RANGER[3] is especially designed for cars, it detects keypoints using a StarDetector and matches features using ORB descriptors. Zhang et all[4] applies a similar technique, but is more focused on indoor robotics.

Foward-facing approaches tend to be more complex. Scaramuzza et all[5] uses one feature point in each image pair to track the movement of the vehicle, and Hamma et all[6] uses an uncertainty projection to track features on the ground plane.

2.2. Localization

Where as Odometry only says something about relative position, localization is used to determine the absolute position. This can be done using landmarks based methods, but also feature based methods.

Landmark based methods use identifiable points for localization, Bailo et al[7] method uses sequences of road markings as unique identifiers to determine the position of a car.

Feature based methods are more flexible, but also more computationally expensive, Yu et al[8] proposes a monocular vision-based approach that leverages the road structures like lane lines, other road markings and even poles and buildings for localization.

Monte Carlo Localization (MCL) [9] is a widely used method for robot localization. It employs a particle filter to represent the probability distribution over possible robot poses. Each particle represents a hypothetical pose, which is updated based on odometry data. The particles are then weighted according to how well their predicted sensor measurements match actual observations. These weights are used to resample the particle set, refining the estimate of the robot's pose.

A related approach, Markov Localization [10], represents the pose distribution using a discrete grid rather than a set of particles. Each grid cell stores the probability of the robot being in that specific location and orientation, and updates are performed over the entire grid using Bayesian filtering.

3. Methods

3.1. Bird Eye View Calibration

The bird's-eye view transformation is a crucial component of the entire system, as it serves as the foundation for all subsequent odometry and localization tasks. Any inaccuracies at this stage propagate through the pipeline, affecting overall performance. A particular challenge arises from the fact that

the cameras mounted on the vehicle are not rigidly fixed and can shift slightly during operation. Such shifts alter the top-down transformation, leading to distortions in the bird's-eye view. To address this, an on-the-fly calibration procedure is implemented.

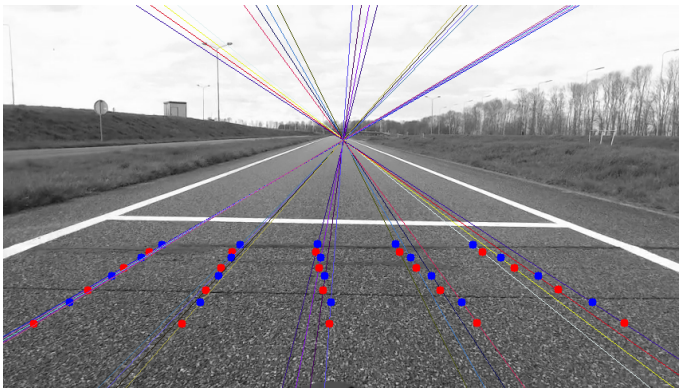


Figure 1.; Blue are the original points, Red are the same points after moving, the lines are lines between pairs of points

This calibration exploits the fact that as the vehicle moves forward, points on the road shift along their epipolar lines in consecutive frames. These epipolar lines converge at a vanishing point, which corresponds to the direction of motion, and can be used to get the BEV transformation[11]. In the top-down view, these lines should appear as straight lines. As shown in Figure 1, the vanishing point can be noisy due to factors such as camera vibrations and slight deviations in the vehicle's trajectory. Therefore, the procedure is repeated multiple times to gather several vanishing point estimates. Outliers are removed using Z-score filtering, and the final vanishing point is computed as the mean of the inlier set.

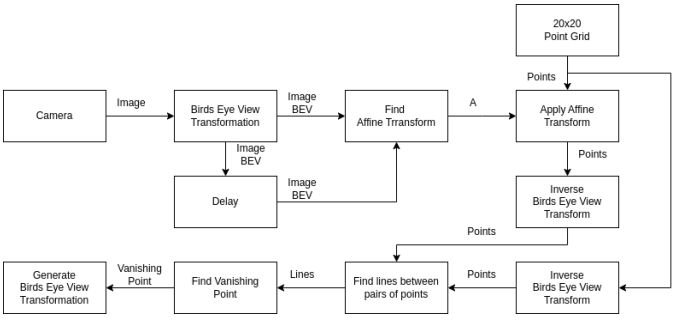


Figure 2.; Birds Eye View Calibration Diagram

Instead of explicitly detecting and matching keypoints between frames, the method utilizes the affine transformation between successive bird's-eye view (BEV) images from the Odometry. Let $B \in \mathbb{R}^{3\times3}$ represent the homography matrix that maps points from the camera frame to the BEV. $A \in \mathbb{R}^{2\times3}$ denotes the affine matrix defined in the BEV frame of reference. This transformation is computed using a regular grid of points $P \in \mathbb{R}^{3\times N}$, defined in homogeneous BEV coordinates and spanning the entire top-down view, their corresponding projections in the camera frame are:

$$P_c = B^{-1}P$$

These are shown as the blue points in Figure 1. After one frame of forward motion, an affine transformation A is estimated between successive BEV frames. Applying A to the original grid yields updated points, and applying B again projects them back into the camera frame:

$$P'_c = B^{-1}AP$$

These transformed points (in red) are visualized in Figure 1. Each pair $(P_{c,i}, P'_{c,i})$ defines a motion vector, and the line connecting these two points represents an approximation of the epipolar line.

To estimate the vanishing point v, we solve for the intersection point of these lines. Each pair of points defines a line in homogeneous coordinates via the cross product:

$$l_i = P_{c,i} \times P'_{c,i}$$

The vanishing point lies at the intersection of all such lines and can be found by solving the overdetermined system:

$$l_i^\top v = 0$$

for all i. This results in a least squares estimation problem, which can be written in matrix form as:

$$Lv = 0, \quad \text{where } L = \begin{bmatrix} l_1^\top \\ l_2^\top \\ \vdots \\ l_n^\top \end{bmatrix}$$

You can then get the vanishing point using least squares:

$$\min_{||v||=1} ||Lv||^2$$

Since individual estimates of the vanishing point can be noisy due to camera shake or uneven motion, the process is repeated multiple times. Outlier vanishing points are filtered using Z-scores, and the final vanishing point v_{final} is computed as the mean of the inlier set. This calibrated vanishing point is then used to refine the BEV transform.

This method assumes that the camera pitch is approximately zero. If the camera is tilted significantly, the affine model becomes insufficient, since affine transformations cannot capture the perspective distortions introduced by such tilt. Affine transformations are limited to translation, rotation, scaling, and shearing. In this calibration procedure, the translation component of the affine matrix effectively adjusts the horizontal position of the vanishing point, compensating for rotational offsets. The remaining parameters subtly modify the shape of the top-down transformation. This is why an initial BEV transformation must already be reasonably accurate, the calibration primarily fine-tunes rotational misalignments.

A potential improvement to this approach would be to combine it with a horizon detection step. First, the horizon could be estimated and the vanishing point initially placed at the center of the image width. A preliminary BEV would then be generated using this assumption. Afterward, the affine calibration could be applied to refine the vanishing point's x-position, and a final BEV transformation computed. This way the method could compute a BEV without an initial transformation matrix.

3.2. Lane Detection

Lane detection is an important component of autonomous driving systems. Although it may appear to be a straightforward task, it presents numerous challenges due to the variability in road and environmental conditions. Lane markings can differ significantly in appearance depending on the road surface, lighting, weather, and wear. Moreover, lanes are not always straight. They often contain curves, merges, or splits, and may be partially or completely occluded or faded[12]. These factors contribute to the complexity of reliable and accurate lane detection.

The implementation faced several practical limitations that significantly influenced design decisions. First, computational efficiency was a major concern due to limited processing power on the target hardware, necessitating lightweight algorithms that could operate in real time. Second, the output needed to be a pixel-wise segmentation of lane

lines and road markings. Finally, there were strict time constraints for development, which limited the complexity of the system.

Given the constraints outlined above, I opted for a straightforward HSV-based color filtering approach. In the environment used for testing, the road surface was consistently black, lane markings were white, and the surrounding area was predominantly green grass. This distinct color separation made it feasible to isolate lane markings using simple thresholding in the HSV color space. While this method lacks robustness in more diverse or unpredictable conditions, it offered the advantage of being easily tunable. Threshold values could be quickly adjusted to accommodate minor variations in lighting or surface appearance, which was sufficient for the scope of this challenge. Although HSV filtering performed the best on road regions closest to the vehicle, in this setup, the road lines near the car were not fully visible in the camera's field of view. As a result, a middle section of the image, where lane lines were more clearly captured, was selected as the region of interest (ROI) to ensure more reliable detection.

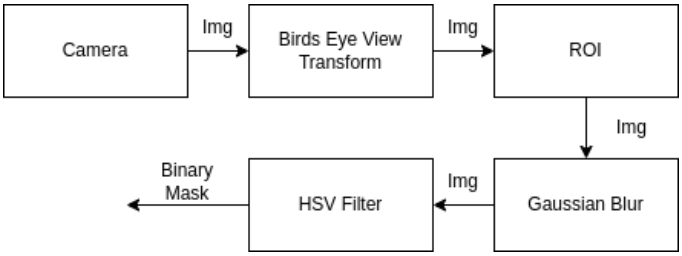


Figure 3.; Lane Line Detection Diagram

The processing pipeline begins by transforming the input image into a bird's-eye view. Next, a region of interest (ROI) is extracted, focusing on the section of the road directly in front of the vehicle, where lane visibility is typically highest. A Gaussian blur is then applied to the ROI to reduce high-frequency noise and smooth the image, which helps prevent false detections. Finally, an HSV color filter is used to generate a binary mask highlighting the lane lines and road markings.

(a).; Input image

(b).; Detected Lane Lines

Figure 4.; Input image to Lane Lines

3.3. Odometry

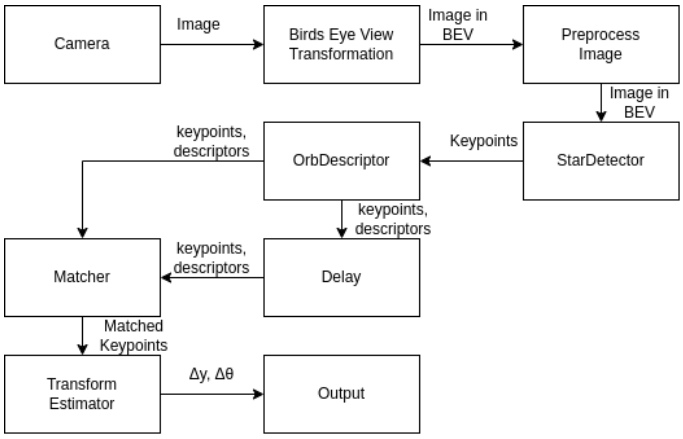
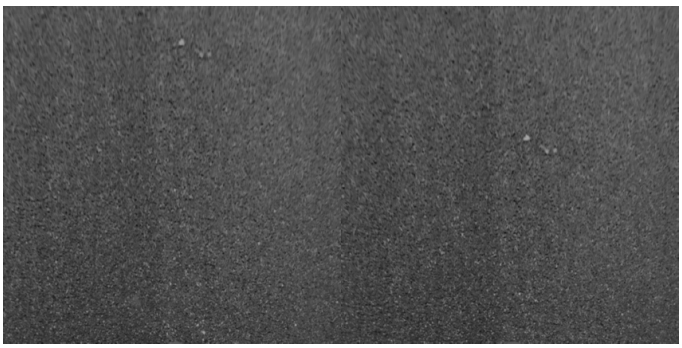


Figure 5.; Visual Odometry Block Diagram

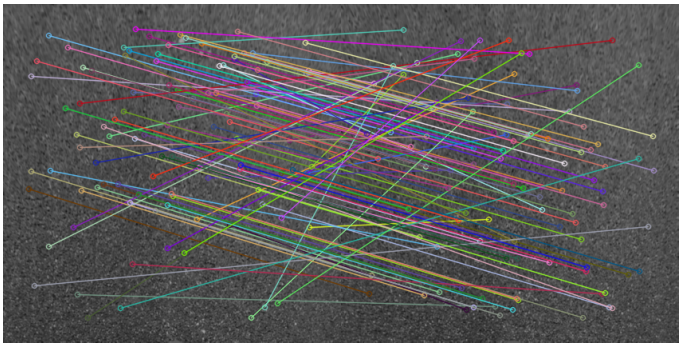
Odometry is used to estimate the position and orientation of a vehicle over time by integrating motion information. However, a key limitation of odometry is its tendency to drift. Since it estimates position indirectly, by integrating velocity rather than measuring absolute position, small errors in speed or heading can accumulate over time without bounds, leading to increasingly inaccurate position estimates[13].

Odometry is commonly derived from sensors such as an IMU (Inertial Measurement Unit) or wheel encoders. However, neither of these were available in this challenge and could not be used. Although the vehicle was equipped with a speed sensor and steering control, no data from these sensors was available during development, as only video recordings from the previous year were available. As a result, I opted to implement visual odometry.

One of the key advantages of this approach is that it relies solely on camera input, a sensor that is already essential for autonomous vehicles—making it highly adaptable and hardware efficient. Visual odometry estimates the vehicle's translation by analyzing motion between successive image frames. This is inherently more stable than acceleration-based estimation from IMUs, which require double integration and are therefore more susceptible to drift. Moreover, visual odometry avoids the pitfalls of wheel encoder-based odometry, particularly wheel slip. In situations such as rapid acceleration, wheel encoders may falsely register movement due to rotation without actual displacement, whereas visual odometry captures the vehicle's true motion by directly observing changes in the visual scene[14].



(a).; Top down view of a portion of the road at 2 different times (left is earlier)



(b).; Top-down view of the same road segment with matched keypoints overlaid. Only 100 of the total keypoints are shown for clarity.

Figure 6.; Keypoint matching between 2 images

The visual odometry system is based on the RANGER[3] system. Unlike RANGER, which uses a downward-facing camera, this implementation utilizes a forward-facing camera mounted on the vehicle. To compensate for the difference in viewpoint, a bird's eye view transformation is applied to the input images. Additionally, a Gaussian blur is used to bring the transformed image into the same scale space. This step is necessary because the apparent texture detail in the road surface varies significantly with distance from the vehicle, features closer to the camera appear sharper and more detailed than those farther away.

The top-down image is optionally resized to improve processing speed. Keypoint detection is then performed using the STAR detector, followed by ORB descriptor extraction for each detected keypoint. To estimate motion, descriptors from the current frame are matched to those from the previous frame using a brute-force matcher. These matched keypoint pairs are used to compute an affine transformation that describes the motion between the two consecutive frames. However, the matches often contains a significant number of outliers. To address this, RANSAC (Random Sample Consensus)[15] is applied to robustly estimate the transformation while discarding inconsistent matches.

From the estimated affine transformation, only the x- and y-translations are extracted. The y-translation represents the vehicle's forward motion, while the x-translation reflects lateral movement in the top-down view. Since the camera is forward-facing and the image is warped to a bird's-eye perspective, lateral motion in the image (along the x-axis) corresponds to rotational movement of the vehicle. Specifically, if the image shifts right, then the car has rotated to the left.

To compute the rotation angle, we model the car's motion as movement along the arc of a circle with radius r, where r is the distance from the camera to the center of the top-down view. The proportion of the circle traversed is given by $\frac{dx}{2\pi r}$, where dx is the lateral image displacement. Multiplying by 2π to convert this fraction to radians gives:

$$d\theta = \frac{2\pi dx}{2\pi r} = \frac{dx}{r}$$

Thus, the vehicle's rotation θ (in radians) can be approximated as the lateral image shift divided by the turning radius.

$$\theta_t = \theta_{t-1} + \frac{dx_t}{r}$$

Using this, the updated global position (x_t, y_t) is computed by rotating the local forward displacement dy:

$$x_t = x_{t-1} - dy_t \sin(\theta_t)$$

$$y_t = y_{t-1} + dy_t \cos(\theta_t)$$

This process is repeated for each incoming frame to incrementally estimate the vehicle's position and orientation. However, if the estimated displacement between two frames falls below a predefined threshold, the frame is discarded. This helps suppress updates based on unreliable or insignificant motion, thereby reducing the accumulation of noise and drift over time.

The radius r of the circle, defined as the distance from the car to the center of the birds-eye view (BEV), was not measured directly. Instead, it was estimated using odometry data. Specifically, as the car rotated 180 (or πradians), the total translation along the x-axis was recorded. Since the car moves along a circular arc during this rotation, the arc length corresponding to half the circumference is equal to the total x-translation. Using the circumference formula $c = 2r$, the half-circumference is r. Thus, the radius can be calculated as

$$r = \frac{x_{\text{total}}}{\pi}$$

3.4. Localization

One of the primary limitations of using odometry alone for localization is drift. Over time, small errors in motion estimation accumulate, leading to significant deviations from the vehicle's true position. Localization addresses this issue by continuously correcting the position estimate provided by odometry. There are various techniques for achieving this, including landmark-based, and feature-based approaches[13]. In this work, I focus on a feature-based method inspired by Monte Carlo Localization (MCL)[9], which uses particle filtering to maintain a probabilistic estimate of the vehicle's pose.

In this approach, the lane lines themselves serve as the primary features for localization. The system continuously adjusts the estimated position and orientation so that the projected lane lines from a reference map align with the observed lane lines in the camera feed. By constantly correcting the pose estimate in this way, the system effectively compensates for the drift present in the odometry.

Lane lines were chosen as the primary features for localization because they are consistently present throughout the track and exhibit sufficient distinctiveness to support pose estimation. While individual small segments of lane markings might lack uniqueness, longer continuous portions provide distinctive patterns that the particle filter can effectively leverage to differentiate between possible poses. Moreover, lane lines offer stability as they remain relatively unchanged over time, and the strong contrast between the road surface and the markings enables efficient and reliable segmentation from camera images. These characteristics make lane lines well-suited for robust and efficient localization.

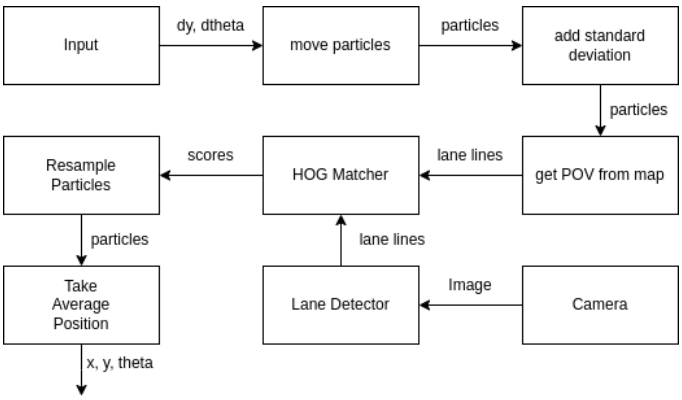


Figure 7.; Localization Diagram

To implement this localization method, two key components are required: a similarity metric for comparing observed lane markings to a reference map, and a particle filter to estimate the vehicle's pose.

For the similarity metric, Histogram of Oriented Gradients (HOG)[16] is used. HOG divides an image into blocks and computes a histogram of gradient orientations within each block. The resulting descriptors can then be compared using a distance metric to evaluate the similarity between the current lane marking observation and the expected lane pattern from a map. To reduce computation time, since this comparison must be performed hundreds of times per frame, all images are resized to 128×64 pixels before HOG computation.

An advantage of using Histogram of Oriented Gradients (HOG)[16] as a similarity measure is its tunable sensitivity through cell size adjustment. By increasing the cell size, the descriptor becomes more tolerant to small positional errors, an important property in this system, as the reference maps are not highly accurate. When lane lines fall within the same 8×8 pixel cell, they contribute equally to the gradient histogram, even if they are slightly misaligned. This allows the system to still recognize a match despite minor discrepancies between the map and observed image. However, this tolerance also introduces a trade-off: reduced precision. Because the HOG descriptor treats small variations within a cell as equivalent, it cannot distinguish between a perfectly aligned line and one that is slightly shifted, as long as both fall within the same cell. This makes the system more robust to noise but less sensitive to fine-grained alignment.

The particle filter models the vehicle's pose as a cloud of particles, each representing a possible position and heading (x, y, θ), and weighted by how likely that pose is given the current sensor input. During each iteration, all particles are propagated using the motion estimate from odometry, with Gaussian noise added to simulate uncertainty and model drift. This results in a distribution of particles representing plausible poses after motion.

Next, for each particle, the predicted view of lane lines, based on the particle's pose, is compared to the actual lane line observations using the HOG distance. This distance is converted into a weight, with lower distances resulting in higher weights. The particles are then resampled based on their weights, using systematic resampling[17]: low-weight particles are likely to be discarded, while high-weight particles are duplicated proportionally. This resampling step refines the particle set, concentrating it around the most likely pose estimates.

Monte Carlo Localization (MCL) [9] is commonly used to estimate a vehicle's position without requiring a known starting point. However, this typically demands a large number of particles to cover the full pose space, which can be computationally expensive. In our case, we benefit from a known starting location: each run begins at the start of the track, marked by a stop line. This stop line serves as a reliable reference for initial localization. A dense set of particles is first initialized around the expected starting position, with Gaussian noise applied to both position and orientation to account for uncertainty. The particle with the best alignment between observed and expected lane features is selected as the initial pose estimate. After this initial localization step, the particle set is reinitialized with fewer particles, allowing for efficient tracking during the rest of the run.

The mean of the particle distribution is used as the measurement input to a Kalman[18] filter, which fuses this information with the odometry data to produce a refined estimate of the vehicle's position and orientation. This approach combines the strengths of both methods: the particle filter provides a robust, observation-driven correction based on visual features, while the Kalman filter ensures smooth, continuous updates using odometry. Since the particle filter yields a probabilistic distribution rather than a single pose estimate, using its average as a measurement allows the Kalman filter to anchor the otherwise drifting odometry to observed lane features.

The Kalman filter employs a state vector **x** that includes the car's pose (x, y, θ), linear velocity v, and angular velocity ω. The state transition matrix **A** models the system dynamics and updates the state **x** over time based on the current velocity and rotation rate.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ v \\ \theta \\ \omega \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & -\Delta t \sin(\theta) & -v\Delta t \cos(\theta) & 0 \\ 0 & 1 & \Delta t \cos(\theta) & -v\Delta t \sin(\theta) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

There are two measurement matrices, H_A and H_B. The matrix H_A corresponds to measurements of linear velocity and angular velocity, while H_B corresponds to measurements of position and orientation. The measurement vector $\mathbf{z}_A$, associated with H_A, is obtained from odometry data. The measurement vector $\mathbf{z}_B$, associated with H_B, includes x, y, and θ values, which are provided by the particle filter. This allows the Kalman filter to update the estimated state using both odometry and particle filter data.

$$\mathbf{z}_A = H_A\mathbf{x}, \quad H_A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \Rightarrow \quad \mathbf{z}_A = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\mathbf{z}_B = H_B\mathbf{x}, \quad H_B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \Rightarrow \quad \mathbf{z}_B = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

The x, y, and θ of the state vector **x** are used as the output of the localization system. The performance of the Kalman filter can be adjusted by tuning the measurement noise covariance matrix **R**. By modifying **R**, it is possible to control the influence of the particle filter and odometry measurements in the state update, placing more or less trust in either measurements.

3.5. System

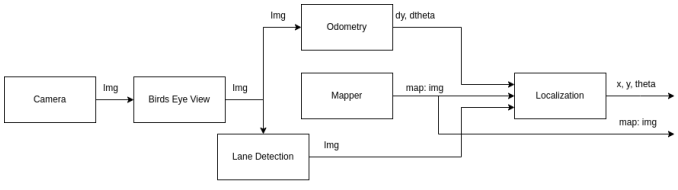


Figure 8.; High level overview of the entire system

In the final system, all components operate in an integrated pipeline to estimate the vehicle's position and orientation in real time. The process begins with the camera capturing sequential images of the environment. Each frame is first transformed into a bird's-eye view using a homography matrix. This bird's-eye image is simultaneously passed to two key modules: visual odometry and lane detection.

The visual odometry module analyzes consecutive frames to estimate the vehicle's relative motion. By detecting and matching keypoints in the bird's-eye view, it computes an affine transformation that reveals the difference in rotation and translation of the vehicle between frames. This incremental motion estimate is then accumulated over time to provide a rough trajectory. However, due to the inherent drift in odometry, these estimates alone are insufficient for reliable localization.

To correct for this drift, the same bird's-eye image is processed by the lane detection module, which isolates road markings using HSV-based color filtering. The extracted lane lines are passed to the localization module, where they are compared with a reference map using a particle filter. This module evaluates how well each hypothetical vehicle pose aligns the observed lane lines with the map, using Histogram of Oriented Gradients (HOG) [16] as a similarity metric. By weighting and resampling particles accordingly, the system maintains a probabilistic estimate of the vehicle's global pose.

Finally, the localization module fuses this estimate with odometry data using a Kalman filter. This allows the system to balance the short-term smoothness of motion estimation with the long-term stability of feature-based corrections. The result is a continuous estimate of the vehicle's global position, defined by (x, y) coordinates—and its orientation θ.

The system can encounter three distinct scenarios related to lane and feature visibility, each affecting the localization process differently. The most common scenario occurs in featureless environments along the Y direction, such as on straight road segments without turns or distinctive features like crossings or stop lines. In these cases, the system maintains the vehicle's position between the road lines, often relying on only one or two visible lane lines. Lateral drift is corrected based on this line, while longitudinal localization depends on odometry and a particle filter, resulting in an elongated particle distribution aligned with the road direction.

A second scenario arises when features are present in both directions, typically during transitions between road conditions, such as entering or exiting curves, or when crossings and stop lines are visible. The detection of these features causes the particle distribution to collapse into a compact, roughly circular cluster, significantly improving localization accuracy. Although this scenario provides the most reliable position estimate, it occurs less frequently than the previous case.

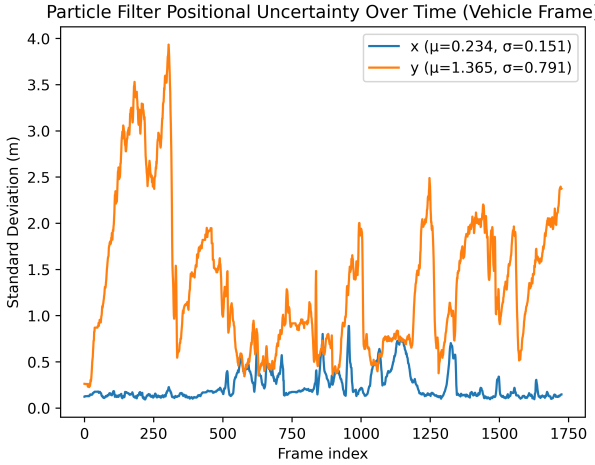
The third scenario is encountered when no lane lines are visible, which is relatively uncommon and usually occurs in short sections such as intersections where lane markings are absent. In these situations, the system relies fully on odometry for localization. Particles spread in multiple possible directions, diverging as the vehicle moves forward to represent uncertainty in the vehicle's position. Once new lane lines or features become visible, the system transitions back to the second scenario, causing the particle distribution to collapse and thereby refining the localization.

One of the modules not previously mentioned is the Mapper module, shown in Figure 8. Currently, this module is responsible for storing the map and providing it to the localization module. Its primary function is to return the point of view (POV) of a particle given a specific location and orientation. However, the Mapper module can be further extended to update the map using detected lane lines or even to generate the map from scratch using odometry data and loop closures.

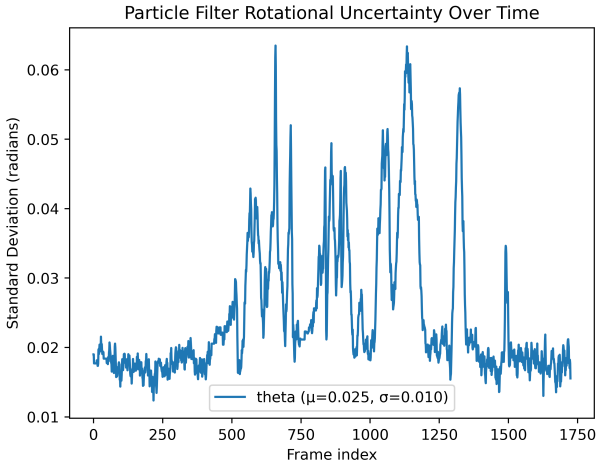
The entire system has been designed to be highly modular. For example, the Odometry component can be easily replaced with another method that provides the incremental translation (dy) and rotation (dθ). Similarly, the lane detection module can be swapped out for alternative methods that produce the same binary mask. This flexibility allows the system to avoid redundant lane detection in a self-driving car setup.

Moreover, the system is straightforward to integrate into other platforms. Simply include the localizer class in your codebase, initialize it, and pass it each frame; it will then output the current pose. The system does require calibration for each specific vehicle by adjusting the bird's eye view parameters and loading the corresponding map.

4. Results



(a).; Positional uncertainty represented by the standard deviation of the particle distribution in the x and y directions (in meters).



(b).; Rotational uncertainty represented by the standard deviation of the particle distribution in heading θ (in radians).

Figure 9.; Per-frame uncertainty estimates from the particle filter in the vehicle reference frame.

Figure 9 shows the positional and rotational uncertainty over time estimated by the particle filter, expressed in the vehicle reference frame. In this frame, the y-axis corresponds to the forward direction of the vehicle and reflects uncertainty in the estimated progress along the track. The x-axis is perpendicular to the direction of motion and represents how well the system estimates the car's position within the lane or on the road. The rotation gives the orientation of

the car on the map in radians.

Metric	Error (radians)
RMSE	0.037
STD	0.035
MAE	0.028

Table 1.; Rotational error metrics (in radians) on straight road segments.

The rotational errors reported for straight sections of the road were computed by estimating the true heading of the vehicle. The true orientation was derived by applying the Hough Line Transform to the detected lane markings. This direction was then compared to the expected orientation from the map; for example, when the vehicle is moving leftward on the map, the expected heading is $\frac{\pi}{2}$ radians. The difference between the expected heading and the detected lane orientation provides an estimate of the vehicle's true orientation in these sections. The rotational error was then computed as the difference between this estimated true orientation and the orientation reported by the localization algorithm.

Metric	Error (meters)
STD	0.10
MAE	0.07

Table 2.; Lateral (x-axis, in vehicle frame) error metrics (in meters) on straight road segments.

The relative error in the lateral direction of the vehicle is calculated using the lane lines. Thirty samples were collected on a straight piece of road, for each sample the center of the road was estimated using the lane lines. The error was computed as the difference between the vehicle's estimated X position and the calculated road center. The error was then subtracted by the mean of the error to get rid of the offset that was present.

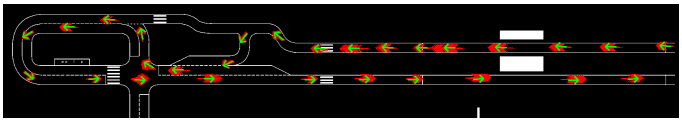


Figure 10.; Map of the track, green arrows are the estimated location of the car, red arrows are the particles used in the particle filter.

In figure 10 you can see the track at the RDW. You can see the path quite clearly if you follow the green arrows. This is the path we had to take for the challenge.

The green arrows in this figure represent the estimated location and orientation of the vehicle. The red arrows are all the particles that are used for the localization, you can see that they surround the estimated position.

5. Discussion

5.1. Birds Eye View

The bird's eye view (BEV) calibration method proved effective for correcting rotational errors. However, due to the inherent limitations of affine transformations, it could not correct for other distortions, such as those introduced by perspective. One way to address this would be to use a full perspective (projective) transformation instead, which can model a wider range of geometric distortions. However, this comes at the cost of increased computational complexity, which is why affine transformations were reused from the odometry system.

Another limitation of the calibration was its inability to estimate scale. Affine transformations assume that all points translate uniformly, which is not sufficient when scale variations are present due to perspective effects. If a perspective (projective) transformation had been used instead, it would have allowed the system to model these scale differences. The BEV could then be adjusted until all points appear to move consistently, an expected property in a true top-down view.

Ultimately, this calibration method was not used during the challenge. Instead, the BEV was manually calibrated prior to the race. This manual approach proved more accurate and allowed for the stitching of multiple camera views into a single, unified top-down image.

5.2. Lane Detection

The lane detection method used in this system was simple, but it proved highly effective. Although it wasn't very robust, requiring manual tuning of HSV filters under different weather or lighting conditions, it handled all relevant scenarios without additional complexity. Unlike many systems that detect lane lines and then model them geometrically, our approach relied solely on binary segmentation. Thanks to this simplicity, the system was able to handle situations like intersections and lane merges without issues. The segmented mask was sufficient for the system. While more advanced lane detection methods like ones based on machine learning might offer greater robustness, their computational cost would be too expensive to run.

5.3. Odometry

Odometry was a key component of the system and generally performed well. However, its main limitation was computational speed. For each frame, around a thousand points were sampled and matched to points from the previous frame, which required significant processing power. This could have been optimized by using a more efficient point sampling method or by replacing the approach with optical flow techniques such as Lucas-Kanade, which assume that points close to each other move similarly, reducing the computational load.

Another challenge with the odometry was accurately estimating rotation. Currently, the rotation estimate is based on the average x-translation. Since the birds-eye view (BEV) is not centered directly in front of the car, different sections of the view move at different rates during rotation, the areas closer to the car move less than those further away. This variation was not accounted for, which reduces precision in rotation estimation.

The odometry's accuracy might have been improved by using the method of Scaramuzza et al. [5], which tracks a single point but relies on a more complex vehicle model. I chose not to pursue this method because it requires tracking a point on the road surface, which can be difficult given the repetitive textures, making single-point tracking less reliable than sampling many points and rejecting outliers.

5.4. Localization

Localization using Monte Carlo Localization (MCL) worked well in practice. While it wasn't extremely precise—largely due to limitations in the odometry and the use of HOG for comparison, it provided a reliable estimate of the vehicle's position. The system required 200 HOG comparisons per frame, which introduced a significant computational burden.

An alternative approach could have been to use the detected lane lines directly for lateral localization in straight sections. This would have significantly reduced computational load, as the particle count could be reduced, but at the cost of increased implementation complexity. Due to this the alternative was not pursued.

Despite its computational cost, the MCL approach proved to be robust. The use of particles compensates for errors

in other components, incorrect particles gradually disappear while those aligned with the correct position remain, allowing the system to recover from small errors over time.

5.5. Results

In figure 9 you can see that there is quite a difference between the standard deviation of x and y. This is due to the road being quite featureless in the y direction most of the time. At the beginning the road is just straight, a piece of road 1 m further looks the exact same as the piece of road at the correct location. In the x direction this is a different story. The road lines provide enough information on where the vehicle is on the road, and it is therefore significantly smaller. The peaks in x are from more difficult pieces of road. Like where the car should make a turn or change lanes. The y direction on the other hand gradually climbs up, and then crashes down. This is due reaching a feature that can shows how far the car is along the track, such as stop-lines, pedestrian crossings or turns. In figure 10 this is easier to see. In the straight pieces of road in the beginning you see the particles spread out over time, then when it reaches a feature like stop-line or crossing it collapses.

The mean uncertainty of the vehicle's position in the Y-axis direction was found to be 1.37 meters, indicating a high degree of imprecision in localization along this axis. In comparison, X-axis direction showed a lower mean uncertainty of 0.23 meters, suggesting relatively better precision. However, this level of uncertainty in both directions remains significant.

However, these values represent the uncertainty of the particle filter, not the actual localization error. On straight sections, the absolute mean error along the X-axis is 7 cm, which is lower than the particle filter's estimated uncertainty of approximately 15 cm. For rotation, the uncertainty in straight sections is around 0.02 radians, while the absolute mean error is slightly higher at 0.028 radians.

The observed lack of precision can be attributed to two primary factors:

- 1. Inaccurate Map Data: The map used for localization was originally sourced from a PowerPoint presentation related to the SDC. Although it was manually edited to better reflect the actual track, it remains an imprecise representation. To compensate for the map's inaccuracies, the standard deviation of the particle filter was increased. While this adjustment improves robustness by allowing the filter to tolerate larger discrepancies between predicted and observed data, it also reduces localization precision.

- 2. Low-Resolution Lane Line Matching: The method used to match lane lines between the map and camera images also introduces imprecision. Input images are downscaled to a resolution of 128×64 pixels before comparison using Histogram of Oriented Gradients (HOG) features with 8×8 cells. This approach improves computational efficiency and robustness to noise but sacrifices localization precision.

6. Conclusion

The final system integrates odometry and lane-line detection using a particle filter. While the resulting localization is not highly precise, it is functional. In straight sections, the mean absolute error (MAE) in the X-axis is 7 cm, and the rotational MAE is 0.028 radians. The particle filter provides an estimate of the system's uncertainty: in the vehicle frame, the mean uncertainty along the X-axis is 23 cm (standard deviation: 15 cm), and along the Y-axis it is 1.37 m (standard deviation: 79 cm). The rotational uncertainty is 0.025 radians with a standard deviation of 0.01 radians.

The research question was: How can monocular visual odometry and lane-line detection be combined to achieve accurate and robust vehicle localization using only a camera on the self-driving challenge track?

This goal has been achieved. The final system is suitable for use in the self-driving challenge, though it may not provide sufficient precision for some applications, like staying within the road lines. Localization along the X-axis is relatively accurate in straight sections, while the Y-axis remains less precise.

7. Future Work

There are several ways to improve the current system. First, enhancing the odometry component could significantly boost overall accuracy. While the current visual odometry approach is functional, it struggles with rotation. One promising direction is to explore single-point visual odometry methods, such as the approach proposed by Scaramuzza et al[5].

The lane-line detection component is relatively simple and could be improved for greater robustness and reliability, particularly under varying lighting and road conditions.

Another limitation lies in the system's computational performance. Currently, Histogram of Oriented Gradients (HOG) features are computed for each particle, which is computationally expensive. Future research could investigate making this more performant.

Finally, instead of relying on a provided map, integrating a SLAM (Simultaneous Localization and Mapping) approach could allow the system to build the map dynamically, which would improve the reliability of the map.

8. AI

This document was edited and refined with the help of AI tools, such as ChatGPT, for clarity, grammar, and phrasing. All technical content, ideas, and structure were created by me. I have thoroughly reviewed the entire text and take full responsibility for the content.

References

[1] Rijksdienst voor het Wegverkeer (RDW), *About rdw – vehicle authority in the netherlands*, https://www.rdw.nl/, Accessed: 2025-06-23, 2025.

[2] Rijksdienst voor het Wegverkeer (RDW), *Self driving challenge*, https://www.selfdrivingchallenge.nl/, Accessed: 2025-06-23, 2025.

[3] K. Kozak and M. Alban, "Ranger: A ground-facing camera-based localization system for ground vehicles", in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016, pp. 170–178. DOI: 10.1109/PLANS.2016.7479698.

[4] L. Zhang, A. Finkelstein and S. Rusinkiewicz, "High-precision localization using ground texture", *CoRR*, vol. abs/1710.10687, 2017. arXiv: 1710.10687. [Online]. Available: http://arxiv.org/abs/1710.10687.

[5] D. Scaramuzza, F. Fraundorfer and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac", in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 4293–4299. DOI: 10.1109/ROBOT.2009.5152255.

[6] D. V. Hamme, W. Goeman, P. Veelaert and W. Philips, "Robust monocular visual odometry for road vehicles using uncertain perspective projection", *Eurasip Journal on Image and Video Processing*, vol. 2015, no. 1, 2015, Cited by: 15; All Open Access, Gold Open Access. DOI: 10.1186/s13640-015-0065-6. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84929000869&doi=10.1186%2fs13640-015-0065-6&partnerID=40&md5=3248d2212e7c052126e1dac45c614019.

[7] O. Bailo, F. Rameau and I. S. Kweon, *Light-weight place recognition and loop detection using road markings*, 2017. arXiv: 1710.07434 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1710.07434.

[8] Y. Yu, H. Zhao, F. Davoine, J. Cui and H. Zha, "Monocular visual localization using road structural features", Cited by: 15, 2014, pp. 693–699. DOI: 10.1109/IVS.2014.6856539. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84905407767&doi=10.1109%2fIVS.2014.6856539&partnerID=40&md5=5a831cfc71a8b891e2ffdf8a92c2be5d.

[9] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte carlo localization for mobile robots", in *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*, IEEE, vol. 2, 1999, pp. 1322–1328.

[10] D. Fox, W. Burgard and S. Thrun, "Markov localization for mobile robots in dynamic environments", *Journal of artificial intelligence research*, vol. 11, pp. 391–427, 1999.

[11] S. A. Abbas and A. Zisserman, "A geometric approach to obtain a bird's eye view from an image", in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 4095–4104. DOI: 10.1109/ICCVW.2019.00504.

[12] Z. Kim, "Robust lane detection and tracking in challenging scenarios", *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, Mar. 2008, ISSN: 1558-0016. DOI: 10.1109/TITS.2007.908582.

[13] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots", *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 869–880, 1996. DOI: 10.1109/70.544770.

[14] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan and N. B. Ismail, "Review of visual odometry: Types, approaches, challenges, and applications", en, *Springerplus*, vol. 5, no. 1, p. 1897, Oct. 2016.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. DOI: 10.1145/358669.358692.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *CVPR*, 2005, pp. 886–893.

[17] M. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking", *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002. DOI: 10.1109/78.978374.

[18] R. E. Kalman, "A new approach to linear filtering and prediction problems", *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.