Do It With No Hands: An Autonomous Vehicle Navigation System

Neelasha Bhattacharjee, Rachel Hawkins, Amina Irizarry, and Katherine Oung

I. Introduction

This project addresses the challenge of developing a fully autonomous vehicle navigation system capable of traversing urban environments. Our design is built on a simple one-lane road network and aims to route a vehicle to a series of loading zones while adhering to traffic rules.

We decompose the autonomous driving problem into its primary components: routing, localization, perception, and motion planning (decision making). Upon completion of these individual systems, we integrate them to create an autonomous vehicle controller that can interpret its environment, plan actions, and execute them safely.

II. Technical Background

Routing

Our routing system determines an optimal path for the vehicle using Dijkstra's Algorithm, a graph-based approach that finds the most efficient route between loading zones by adding a road segment to the path only if it maintains a minimum possible distance between our starting location and target destination.

We first use the predefined map segments to construct a graph, with roads as edges and intersections and loading zones representing nodes. After reading the vehicle's starting position and target destination, we compute the optimal route to our target zone using the process described by the pseudocode below.

```
function route(graph, start, target)

    1. Set start = 0, and set all other
       nodes = infinity

    2. Mark all nodes as unvisited

    3. Set current node = start

    4. Repeat until target is visited:
        a. For each unvisited neighbor of
           current node:
            - Calculate tentative distance
              through current node
            - If new distance is shorter,
              update neighbor's distance
              and mark current node as its
              predecessor

        b. Mark current node as visited

        c. Select the unvisited node with
           the smallest tentative distance
           as the next current node (if no
           reachable unvisited nodes
           remain, exit loop)

    5. To reconstruct the shortest path:
       - Start from target node and follow
         predecessors back to start node

    6. Return shortest path
```

Fig. 1. Pseudocode implementation of the No Hands routing algorithm.

Localization

Our localization system uses an Extended Kalman Filter (EKF) to fuse data from onboard sensors and estimate the vehicle's state. The EKF is a nonlinear estimation model that operates through two key steps, prediction and correction.

Prediction projects the vehicle's current state forward in time based on previous state estimates, control inputs (velocity, steering angle), motion model of the vehicle, and time elapsed since the last update. The Jacobian of the current state and process noise are used to recalculate the covariance matrix.

Correction will then update the prediction using new sensor measurements. A residual between the predicted sensor measurement and the actual sensor measurement is calculated. Then the Kalman gain is computed, factoring in the current covariance matrix, measurement noise, and the Jacobian of the current state. This value balances trust between prediction and the measurements. The predicted state from the measurement model is finally updated taking into account the Kalman gain and the measurement residual.

This approach leverages both measurements and a sound motion model while accounting for respective noise and provides the most accurate estimate of the vehicle's current location.

Perception

The perception system leverages camera-based measurements in conjunction with state estimation techniques to track surrounding objects, supplying essential input to motion planning's collision avoidance logic.

Cameras detect objects within their field of view and generate bounding boxes around them. The information transmitted to the perception module consists of the pixel coordinates corresponding to the top-left and bottom-right corners of each bounding box.

By combining the camera's extrinsic calibration parameters with the ego vehicle's estimated location, the perception algorithm transforms pixel-level bounding box data into global real-world coordinates, allowing for accurate estimation of surrounding object positions.

Once an object is detected, it is essential to track its trajectory over time to ensure continuous collision avoidance. This is achieved by maintaining a list of all tracked objects and their corresponding state estimates. When a new object enters the field of view, the system initializes estimates of its position, velocity, heading, and angular velocity. For objects previously observed, the system updates their state using an Extended Kalman Filter (EKF).

This perception algorithm continually reads the ego vehicle's state and imaging data and uses these as inputs to calculate the relative positions of all other objects.

Motion Planning

Motion planning translates localization, perception, and routing information into executable control commands.

Our controller initiates this process by generating a polyline for each road segment designated by the routing algorithm. These polylines are constructed as follows:

- If the length of the segment is over 80 meters or its curvature is non-zero, add a midpoint (waypoint) between its start and end points.
- If the road segment is a loading zone, get the loading center for the car to turn into.
- Use a flag to denote the presence of stop signs; if there is a stop sign at the end of the road segment, set the flag to TRUE.

In conjunction with polyline construction, we adapted a simple pure pursuit controller to direct the vehicle to its target destination. This is a tracking algorithm that attempts to select target points along the polyline based on a lookahead distance. In the final version of our controller, we modify our approach to test three points on the polyline as our next "best" location, selecting the closest to the lookahead distance. Between each waypoint, the controller computes adjustments in heading angle and linear velocity to minimize cross-track error. In particular–while navigating the route provided–our controller focuses on speed control and collision avoidance.

Regarding speed control, our decision making algorithm ensures the car's velocity is within the speed limit at all times, and it sufficiently slows the vehicle at curves to maintain stability. The controller commands the vehicle to a gentle stop upon both reaching a waypoint marked with a stop sign and approaching a target loading zone.The vehicle will remain stationary while reading the next road segment/target zone, respectively, and resume driving shortly thereafter– assuming there is not an object or stop sign ahead of us.

We implement collision avoidance as a separate channel for real-time responsiveness using the idea of a "collision cone".

Fig. 2. 30° "collision cone" for straight road segments.

Fig. 3. 90° "collision cone" for intersections.

As illustrated in the preceding diagrams, we construct a cone with vertex centered at our heading angle. If our perception system detects any object within this cone, the vehicle will stop until the object is no longer in view. On straight roads, this cone has a vertex angle of 30° (Fig. 1), and at intersections we increase the angle to 90° (Fig. 2) to account for increased traffic and possibility for collisions.

III. Quantitative Results

Routing: Optimized Traveling Distance

The success of our routing system was demonstrated by comparing across trials our path distance with the true shortest path distance, determined by a generic (verified) implementation of Dijkstra's shortest path algorithm.

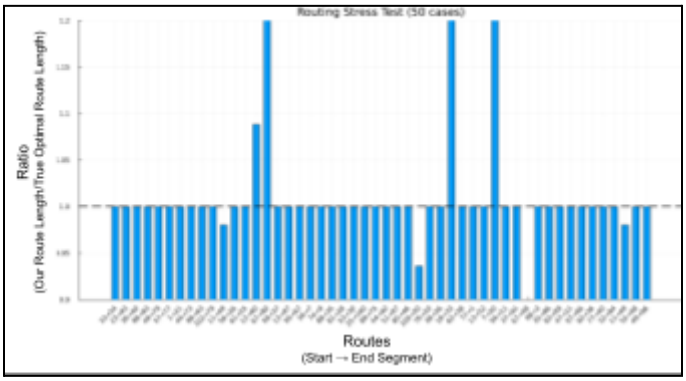
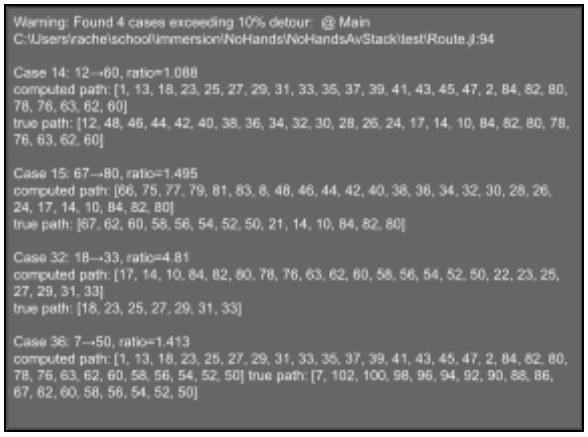


Fig. 4. Ratio of our route length against a generic shortest path algorithm; 50 random trials.

Figure 4 shows that our routing algorithm produced an optimal path in 46 out of 50 trials, indicated by a ratio of less than or equal to 1.0. The specifics of the four failing cases, given in Figure 5, indicate that the paths found by the generic algorithm would not adhere to our traffic guidelines– namely the ban on u-turns– so the paths produced by our algorithm in these cases are likely still optimal.



,Fig. 5. “Failing” cases; scenarios where the ratio between our path distance and the true shortest distance exceeds 1.0.

Motion Planning: Safety and Accuracy

…coming soon… 2 figures… a table?

Localization Accuracy

For testing purposes, data was collected from a simulation where the test vehicle lapped around the track while accelerating and decelerating. The EKF-based localization system was evaluated against ground truth data with the following results.

Positional Error

Average Position Errors

	Position Axis	Average Error
1	x_pos	3.508589353652426
2	y_pos	2.663504009068498
3	z_pos	0.005363020645858058

Table 1. Average error of x, y, and z position estimations for the ego vehicle against ground truth.

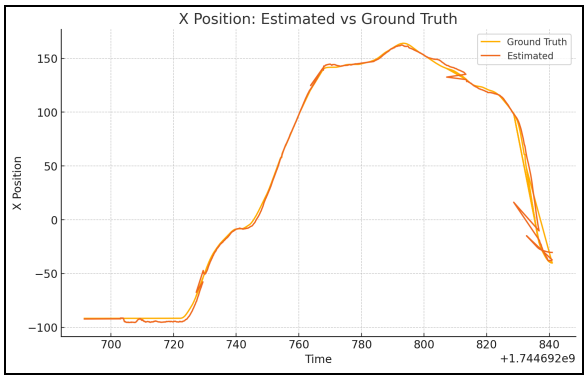


Fig. 8. x-axis positional error over time.

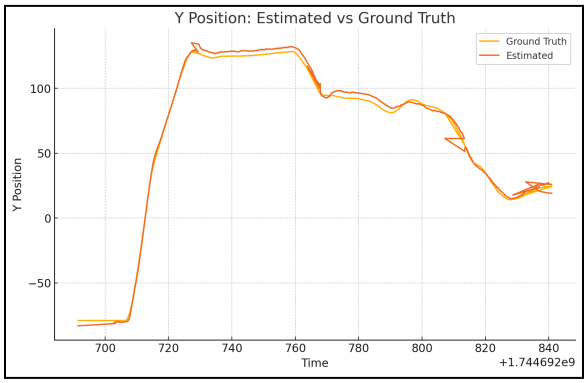


Fig. 9. y-axis positional error over time.

Figures 8 and 9 show that the localization system successfully estimates the ego vehicle’s position within a reasonable margin. The estimated coordinates generally align with ground truth measurements, aside from the deviations evident at around the 810 and 830 second marks in both graphs. As expected, error is more pronounced when the vehicle rapidly changes velocity and/or steering angle.

Orientation Error

Yaw Error Table

	Component	Average Error
1	Yaw	0.584

Table 2. Average yaw (heading angle) error against ground truth after adjusting process and measurement noise

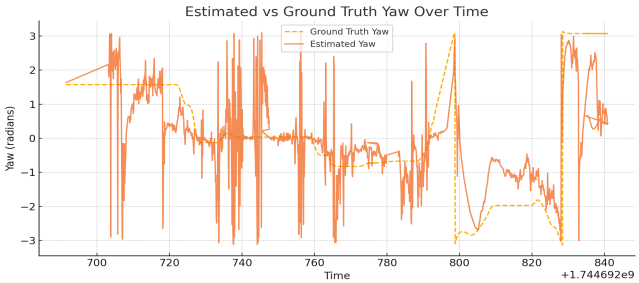


Fig. 10. Yaw error over time before adjusting process and measurement noise

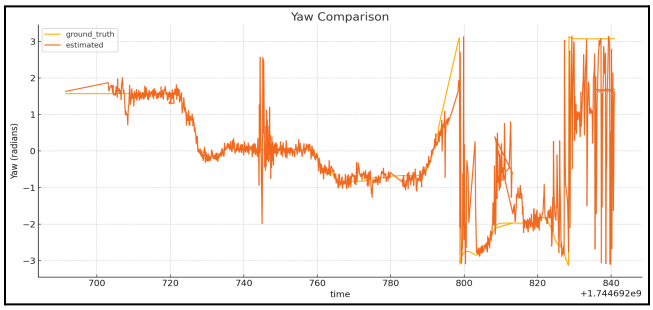


Fig. 11. Yaw error over time after adjusting process and measurement noise

When connecting the routing and motion planning systems to the localization and perception systems instead of using the ground truth channels, the car was only able to move in a continuous circle which indicated a potential issue within localization. Initial testing revealed a high error in yaw estimation with rapid oscillations that was identified as a potential cause of the car's inability to stop steering. Ultimately to mitigate the issue, measurement and process noise was adjusted with the lowest error resulting from collectively lowering both values while having the lowest measurement noise. The final orientation error remains relatively low throughout the duration of the trip. The average of 0.584 given in Table 2 is misleading, however, since our yaw estimations oscillate over the true measurements in a way that makes positive and negative signed errors cancel out (Fig. 10). In the future, it would be ideal to explore different ways to analyze test data and metrics that serve as a more accurate summary for the task at hand, such as circular mean squared error which is designed to handle angular data.

3. Positional Velocity Error

Velocity Component Errors		
	Component	Average Error
1	vx	2.078
2	vy	0.236
3	vz	0.234

Table 3. Average positional velocity error against ground truth.

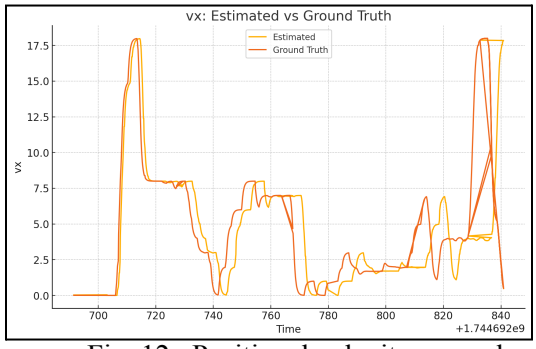


Fig. 12. Positional velocity error along the x-axis over time

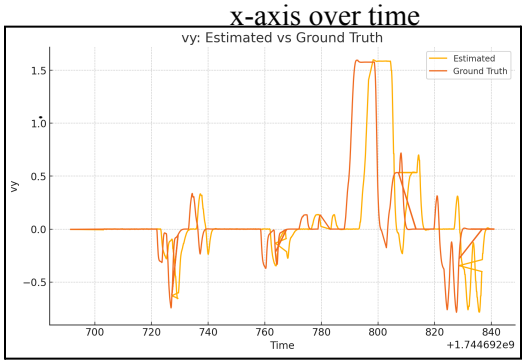


Fig. 13. Positional velocity error along the y-axis over time.

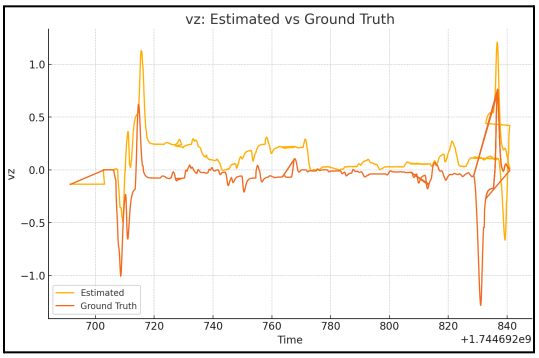


Fig. 13. Positional velocity error along the z-axis over time.

4. Angular Velocity Error

Angular Velocity Errors		
	Component	Average Error
1	wx	0.019
2	wy	0.018
3	wz	0.096

Table 4. Average angular velocity error against ground truth.

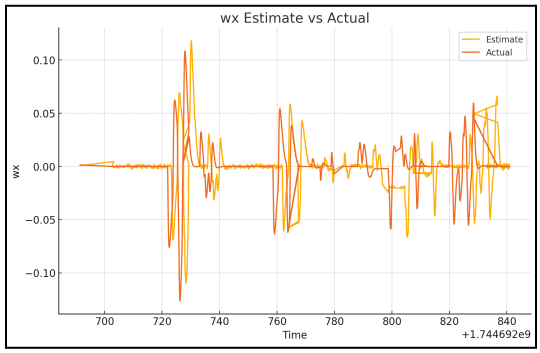


Fig. 14. Angular velocity error along the x-axis over time

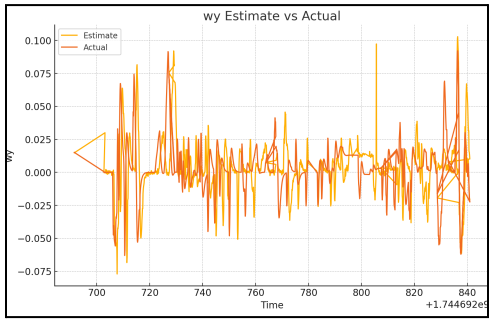


Fig. 15. Angular velocity error along the y-axis over time

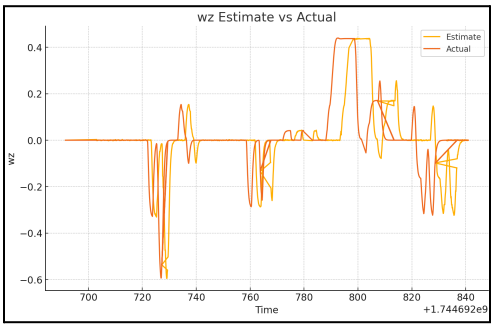


Fig. 16. Angular velocity error along the z-axis over time

Positional and angular velocity error remained relatively low throughout testing, with the localization system following ground truth measurements closely with a slight delay which appears to be trivial in effect. However, if it were ideal to resolve this delay it is likely that the way the localization system time stamps measurements would need improvement, one possible adjustment being using an average of recent measurements and using the average timestamp rather than just grabbing the most recent measurement from the channel and disregarding the rest. It would also be ideal to keep track of the discrepancy between the IMU and GPS measurements as currently the most recent measurement available in each channel is being used in both the prediction and correction step, but it is likely that there is a discrepancy between the timestamps. Angular velocity error also appears to spike temporarily when steering is applied, but this is both expected and within reason.

Perception Accuracy

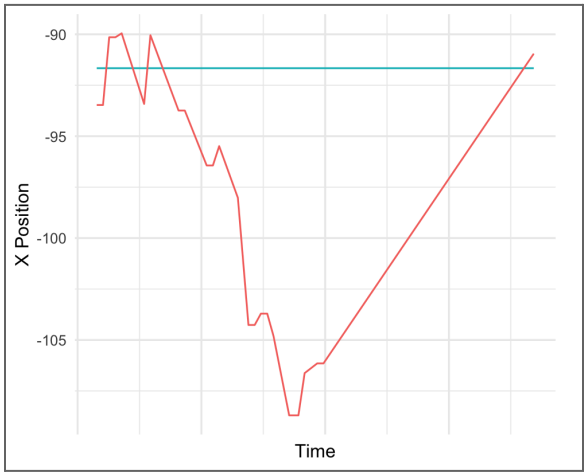


Fig. 20. Positional error along the x-axis over time; ego car approaches stationary vehicle.

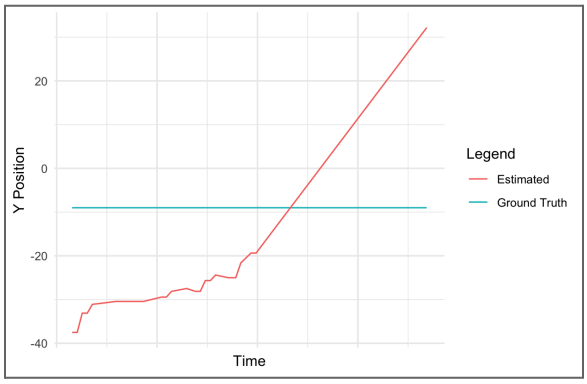


Fig. 21. Positional error along the y-axis over time; ego car approaches a stationary vehicle.

Three specific test scenarios were used to evaluate the perception system:

1. **Test 1 - Approaching Stationary Vehicle**
    - x position starts with a strong initial estimate
    - x position briefly deviates from the true value, but converges back to an accurate estimate
    - y-position's initial estimate is weaker and less stable

2. **Test 2 - Parallel Driving**:

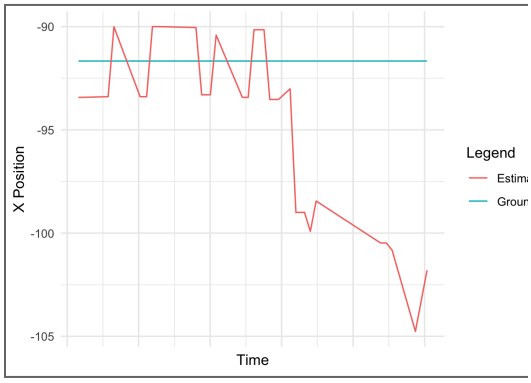


Fig. 23. Positional error along the x-axis over time; cars driving in parallel.

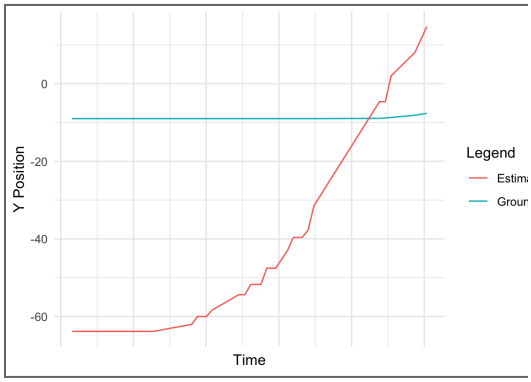


Fig. 25. Positional error along the y-axis over time; cars driving in parallel.

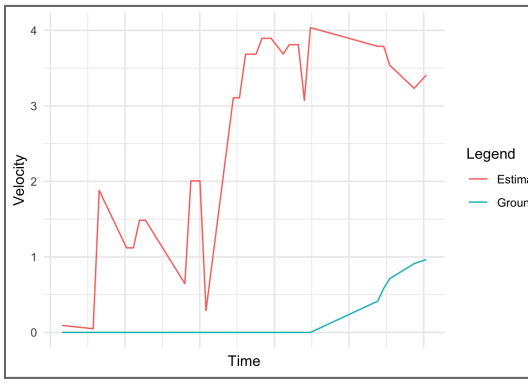


Fig. 22. Velocity error of non-ego vehicle over time; cars driving in parallel.

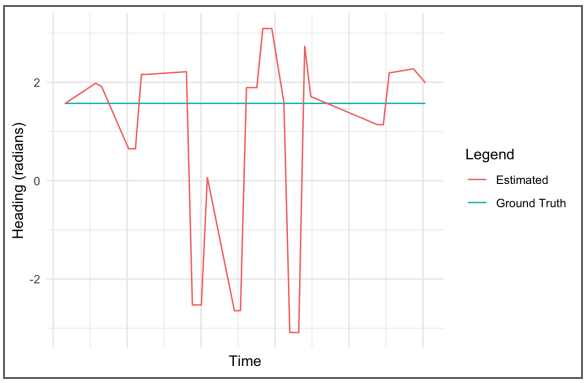


Fig. 24. Heading error of non-ego vehicle over time; cars driving in parallel.

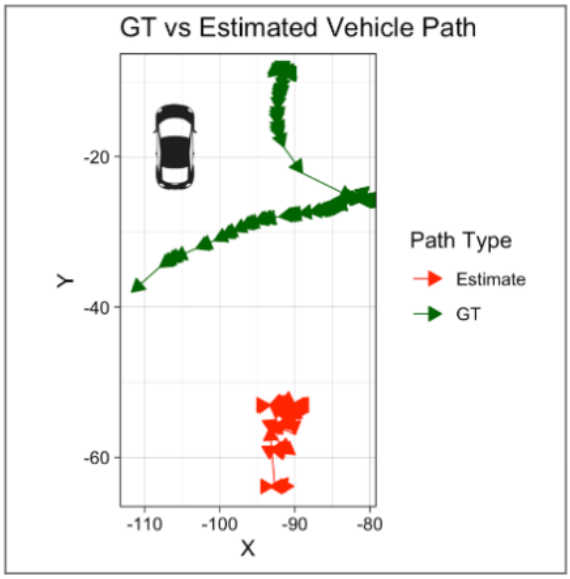


Fig. 26. Path for non-ego vehicle circling stationary ego vehicle; our perception estimation vs ground truth.

- Velocity estimates matched ground truth with less than 10% error
- Maintained tracking through varying relative speeds

3. **Test 3 - Circling Vehicle**:

- Successfully captured the general motion pattern of the second vehicle

- However, the estimated position of the second vehicle is inaccurate, resulting in spatial misplacement despite correct similar shape
- Demonstrated the system's ability to handle changing orientations and perspectives
- Maintained tracking through 360 degrees of relative position changes

IV. Qualitative Takeaways

What Worked Well

1. **Extended Kalman Filter for Localization**:
    - The EKF provided robust state estimation by effectively combining GPS and IMU data
    - Tuning the noise parameters significantly improved performance, especially for orientation estimation
2. **Pure Pursuit Controller**:
    - The modified pure pursuit approach with multiple candidate points improved path following
    - The controller handled both straight and curved segments effectively
3. **Perception System**:
    - Object tracking maintained consistent identification across frames
    - The system successfully handled various relative positions and movements

Challenges and Limitations

1. **Controller Oscillation**:
    - The pure pursuit controller exhibited some zig-zagging behavior, indicating potential for improved tuning
    - This oscillation increased at higher speeds
2. **Intersection Handling**:
    - Complex intersections required special consideration for collision avoidance
    - The expanded detection cone (90 degrees) was necessary but sometimes overly cautious
3. **Loading Zone Navigation**:
    - Precise positioning at loading zones sometimes proved challenging
    - Required specific handling to ensure proper alignment

Key Lessons Learned

1. **Sensor Fusion Importance**:
    - No single sensor provides sufficient information for reliable autonomous driving
    - The combination of different sensor inputs through methods like EKF is essential
2. **Parameter Tuning Impact**:
    - Small adjustments to parameters (like process and measurement noise) can dramatically affect system performance

- Iterative testing and refinement were crucial to achieving good results

3. **System Integration Complexity**:

   - The interaction between different components (routing, perception, localization, decision making) introduced unexpected behaviors
   - Thorough testing of the integrated system was as important as testing individual components

V. Team Contributions

Final Project Rubric:

Your project should be a minimum of 4 pages. You can format it however you like, e.g. single or double space, figures, tables, etc. Grading will depend on the content that is contained in your report. Your writeup should include the following:

1. Introduction (problem description and planned approach)

2. Technical Background (explanation of how methodologies work, e.g. EKF, particle filter, model predictive control, PID control, etc.)

3. Quantitative Results of implementation (metrics from your personal testing, e.g. ability to localize compared to ground truth, ability to identify state of another vehicle compared to ground truth)

4. Qualitative takeaways – what worked, what didn't, what did you learn

5. Team contributions – who did what, what % of total work did each member contribute

Rubric:

30%: Quality of writing (organized arguments, coherent flow and structure)

35%: Quality of technical background–accurate representations of methods, demonstrating a strong understanding of the concepts (while being concise given space limitations)

35%: Quality of evaluations (your methods don't have to work well, but you should be able to present clear evidence of them working or not)