Final Challenge Report: Synthesized Path Planning, Computer Vision, and Control for Autonomous Navigation in a Range of Racing Challenges

Team #14

Isabella Do Orozco
Spring Lin
Jace Lu
Kristoff Misquitta
Mark Razanau

RSS

May 14, 2024

1 Introduction (Spring)

The final challenge addresses previous labs, combining color segmentation, localization, path planning, and PID controls. These stem from the visual servoing, localization, and path planning labs. By building off all of these previous labs, this challenge produces the most sophisticated version regarding space navigation for our vehicle. The two parts of the final challenge is essentially how the vehicle perceives itself and navigates in an unknownspace.

Mario's Circuit addresses the challenge of color segmentation and having the robot respond to the local geometric features. This builds on the line following lab, however, this time the robot has to follow not only one line, but two. This demonstrates the ability to respond to local geometric features and track multiple lines, showcasing the practical application of color segmentation and PD controls in real-world scenarios. This not only enhances our understanding of the environment but also lays the groundwork for more complex navigation tasks.

When accounting for what methods would best work for our vehicle, we went through many iterations on line detecting and noise reduction which resulted in slicing the given image into two images to run color segmentation of the white of

the track lanes on. Originally post processing was implemented on the images, though considering our vehicle's processing limitations, the focus went more in depth on the control systems instead of the image processing. This leads to then stitching the two images together where the average of the two bounding boxes becomes the bounding box of the final image, which should be the center of the lanes. With the given point to follow, PD is then utilized to follow the given point. Though pure pursuit was initially the approach, given the sensitivity of pure pursuit, PD seemed to be the best approach given the easy sensitivity flexibility of the controller compared to pure pursuit.

Luigi's Mansion tackles the intricate challenges of localization and path planning with finesse. Path planning, in particular, offers a robust solution for navigating between arbitrary points within any given space. Our approach involved gathering location data along the Stata basement, constructing a comprehensive path, and establishing an efficient trajectory framework. By representing the path as a series of coordinate poses, the vehicle can discern its position relative to the closest points on the path and adeptly follow the sequence, adjusting its trajectory with an appropriate look-ahead distance to navigate bends and curves seamlessly.

To integrate goal nodes into our navigation system, we leverage the established trajectory framework and the vehicle's current location. By combining these elements, we create a pre-planned trajectory tailored to the specified goal points. Augmented by our localization capabilities, the vehicle gains a clear understanding of the trajectory within the real-world environment, enabling precise navigation along the designated path.

These methodologies draw inspiration from real-world deployments of autonomous vehicles, offering a navigation model that prioritizes robustness and independence from geometric or human-configured environmental features.

2 Technical Approach

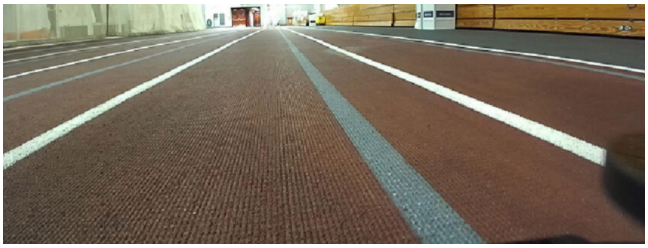
2.1 Race Challenge

For the Johnson Race component of the final challenge where the vehicle must complete a full lap around the track within its designated lane, our method can be divided into core modules: lane detection and lane following.

2.1.1 Lane Detection (Mark)

Before the vehicle is able to make a lap within its lane as fast as possible, it must first be able to identify the boundaries of its lane. In order to do this, we were heavily inspired to leverage similar color segmentation techniques utilized during the line following lab. As the lane is designated by a consistent sharp white shade, we can use a masking technique to extract all the white pixels from the raw input of the ZED camera feed. However, due to surrounding white noise

within the environment from walls and overhead lights, we took advantage of our ZED camera being at a fixed height by limiting our masking to a specific slice of the image that would only show the lane lines ahead.



(a) Raw image feed from the vehicle's ZED camera.

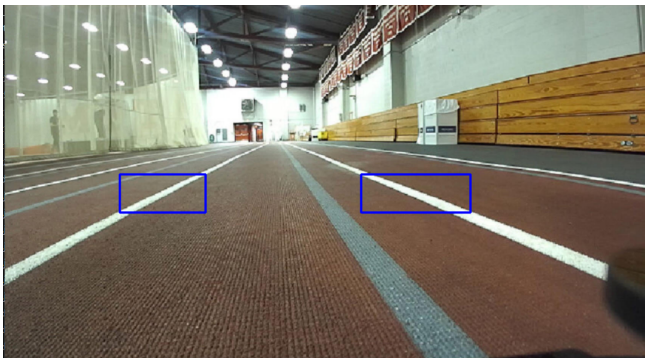
(b) Masked output where the white pixels are isolated in the image.

(c) Slice of raw image to isolate just the track.

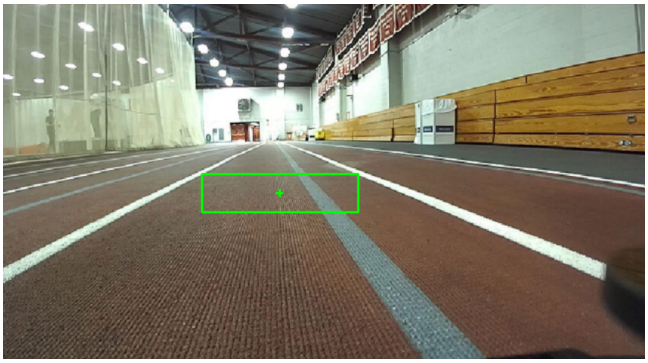
(d) Masked output of the slice images, highlighting only the white lanes.

Figure 1: Color segmentation techniques used to identify the white lane lines. While Figs. 1a-1b show the method used on the entire ZED camera feed, Figs. 1c-1d show the input utilized by the vehicle to understand the location of the white lanes ahead of it.

After segmenting the white pixels, we can identify the closest lanes and draw the respective bounding boxes by identifying the largest contours within the sliced mask image. In order to be able to identify both the left and right lanes separately, we split the image into two halves. With the two lanes identified and bounded, we can then estimate the middle of the lane as the center between these two bounding boxes.



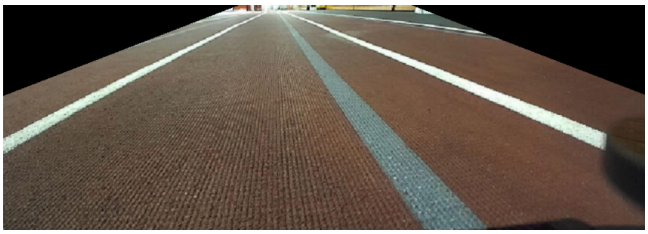
(a) Lane line bounding boxes.



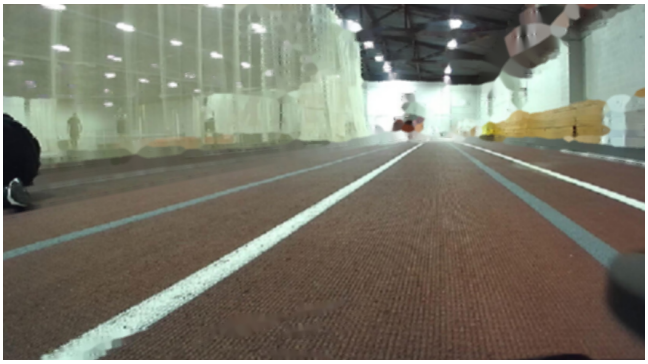
(b) Center lane bounding box

Figure 2: By taking the largest contours of the pixel, we can identify the corresponding bounding boxes for the left and right lane lines as seen in 2a. With the location of these two bounding boxes identified in the image, we can estimate the middle of the lane by taking the center point between these boxes as shown in 2b.

In order to reduce noise in the image from other neighboring lanes, we also tried to utilize additional post-processing techniques. Two techniques that we attempted to implement include a 1) trapezoidal image segmentation, where each raw image is segmented into a trapezoidal shape to "cut out" the lanes that are considered additional noise, and 2) Hough transforms, where edges and lines that fall within a given slope range are discarded.



(a)



(b) By designating the middle of the lane as the space between the two bounding boxes, we can estimate the center of the lane.

Figure 3: Using a trapezoidal segmentation, we can attempt to cut out the neighboring lane lines by shifting the angle at which we cut out the corners as seen in 3a. However, this often led to reduced robustness as oscillations would "blind" our robot of the closest lane lines. On the other hand, while the Hough transforms were more successful at removing neighboring lines as seen in 3b, these were also vulnerable to oscillations in our movement as it affected the calculated slopes of the lines.

While both showed promise on the straights of the track with a perfect controller, they were liable to significant error around corners and oscillations as the segmentations were not dependent on changes in the orientation of the robot. As a result, this would either cut out the left or right lane lines if the robot began to face the other direction, or failed to distinguish the neighboring lane lines as their relative angle changed. We also faced additional issues with the Hough transform as it reduced our performance and estimations due to added blur placed upon the original image. Ultimately, as these methods lacked the robustness we required for our lane follower, we shifted our efforts towards perfecting our PD controller that would allow us to remain within the lane as we navigated the course.

2.1.2 Lane Following (Jace)

Once we were able to consistently find the lines of the lane, we implemented a lane following algorithm using a combination of pure pursuit and PD controller.

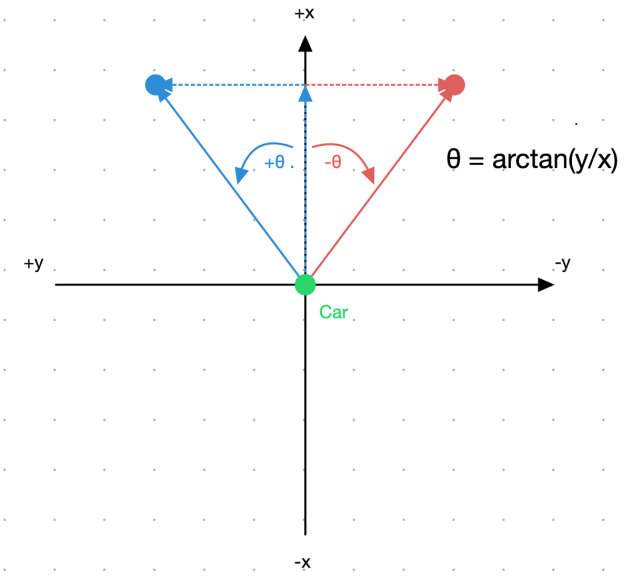


Figure 4: Math used to find the steering angle during pure pursuit.

Figure 4 depicts the mathematical approach initially used to compute the steering angle for our pure pursuit algorithm. This method was quickly revised due to the algorithm's heightened sensitivity to noise in data points at elevated velocities.

We opted to use the steering angle generated by our pure pursuit algorithm as the error input for a PD controller. The output from this controller is subsequently issued as a steering command to the robot.

```
def PID(error, prev_error, dt, Kp, Kd):
    P_out = Kp*error
    derivative= (error-prev_error)/dt
    D_out = Kd*derivative
    steering_angle = P_out + D_out
    return steering_angle
```

While the PD controller mitigated the sensitivity issue inherent to the pure pursuit algorithm, we observed that the robot tended to veer right due to the wheel alignment. To address this, we introduced a constant positive angle to the steering angle produced by the PD controller. This adjustment effectively decreased the extent of right turns, characterized by negative steering angles,

and enhanced the magnitude of left turns, which are denoted by positive steering angles.

```
steering_angle = PID(error, prev_error, dt, Kp, Kd) + C
```

2.2 City Driving

For Luigi's Mansion, the robot must be able to navigate a simulated city landscape to reach designated goal nodes while adhering to traffic laws.

2.2.1 Building Base Trajectories (Kristoff)

When possible, it's always good to precompile trajectories to minimize time and risk associated with planning online. In this regard, rather than run A* or RRT without optimality guarantees on race day, the car was driven systematically down both lanes of the Stata basement, with points collected from the particle filter at regular intervals. These collected poses were then aggregated to form two straight-line trajectories the car could safely use to navigate the basement. This transformed the problem of staying on the right side of the road into one of designing an accurate enough localizer to follow a given path (which would already be on the right side of the road). The two trajectories collected are displayed in Fig. 5 and Fig. 6.

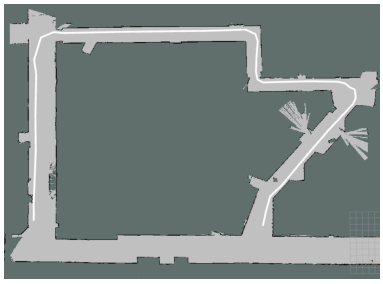


Figure 5: Precomputed trajectory from start of Stata basement (right side of the road).

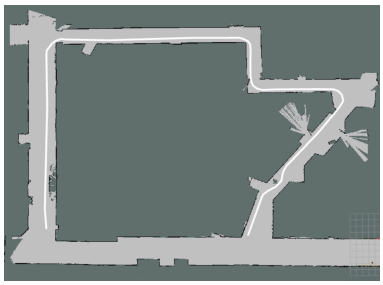


Figure 6: Precomputed trajectory from end of Stata basement (left side of the road).

2.2.2 Creating Trajectory to Reach Goals (Isabella)

After the base trajectory paths were created, they were used in planning paths to reach the given goal nodes.

The two main pieces of information we kept for the robot were which trajectory it currently is on and at which index. Because some of the nodes in the trajectory were far apart, we first extrapolate extra points between them. Then, we find the minimum distance between the goal node and each extrapolated trajectory. With this, we can determine which path the robot would need to be on in order to pick up the shell. With the side the robot is on and by comparing indices, we can see which of the following four cases this falls under. Then, the goal node is always added between the 2 closest nodes on that trajectory.

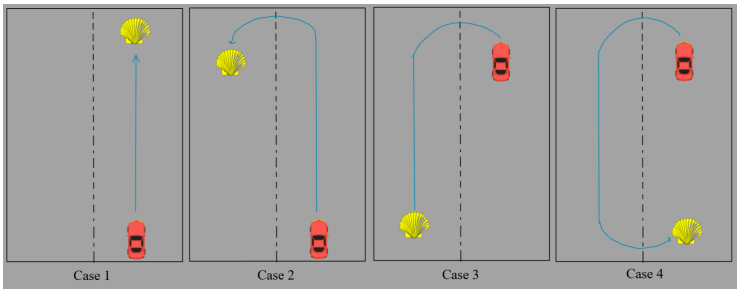


Figure 7: Four possible cases of shell location

If Case 1:
If the shell is on the same side as the robot and the index of the shell is greater than the robot, then the robot simply has to keep moving ahead on the trajectory to reach the shell. All nodes between the robot's current position and the shell are added to the overall path plan as well as the node directly following the shell. The current index of the robot is then updated to this new point.

If Case 3 or Case 4:
If the shell is either behind the robot on its same side or is on the opposite side but ahead, the first move needed is a U-turn. By finding what indices on the opposite trajectory that the robot is between, we chase the point closer to the point aka further along the trajectory by adding it to the planned trajectory. Then, we update which trajectory the robot is now on as well as its index to this turn point.

If Case 2, 3, or 4:
For all three of these cases now, there is simply a straight away. This will follow the same logic as Case 1. However, for Case 2 and 4, the goal node and preceding node are not yet added to the trajectory. The current index is then updated.

If Case 2 or 4:
In these cases, one final turn is still necessary. To ensure the robot goes far enough forward before attempting to turn onto the goal node, it goes one further node in its current path. Then the goal node itself is added to the planned trajectory to be chased. The current trajectory of the robot is updated and its current index is updated.

This is then repeated for all goal nodes. This final stitched together trajectory is then published.

Figure 8: Example Generated Trajectory

2.2.3 Localization Challenges (Kristoff)

Maintaining an efficient and accurate localizer was paramount for the challenge, which featured long navigation tasks with no opportunity to re-initialize the car's pose without a penalty. Accurate localization meant reduced drift accumulation with time, while efficient localization meant the particles kept up with

the car's real-time position so decisions could reflect the current state, not a past one. The accuracy and efficiency of localization were modulated primarily by setting the number of particles and the noise associated with the motion model. Specifically, it was found that in long, featureless hallways, conditionally amplifying x-noise could distribute state estimates more widely and force the localizer out of false stagnancy. While a large number of particles (1000) increased the accuracy of the state estimate, it also increased computational burden. Lag introduced into the system was more than enough to offset any computational gains. Further refinement and testing of these parameters pointed towards 400 particles for a good tradeoff of speed and accuracy. Despite extensive testing, the localizer still failed in the vicinity of the water fountain, a unique environmental feature. On analayis of a rosbag collected later, it was observed that the particles froze in place, so the car lost all sense of position. This rendered collecting the second shell impossible and, without good safety controller design, limited the car's ability to directly return for the third shell.

3 Experimental Evaluation

3.1 Race Challenge (Jace)

We attempted to chart the x error, y error, and distance error of the car using the rqt graph tool. Unfortunately, rqt consistently shut down before the runs were complete, preventing us from collecting these error plots.

Our robot exhibited strong performance during the race, successfully navigating the racetrack at speeds of 2.6 m/s. We subsequently recorded it achieving a higher speed of 2.8 m/s on the same track later.

3.2 City Driving (Isabella)

While the robot was able to consistently reach the first shell with ease, because of issues with the localizer in the long South hallway, it had trouble stopping at the second shell. In our best run, we were able to manually correct which allowed it to stop at the third shell.

4 Conclusion (Spring)

In conclusion, the culmination of our efforts in the final challenge represents a significant leap forward in the sophistication of our vehicle's navigation capabilities. By integrating key concepts from previous labs such as color segmentation, localization, path planning, and PID controls, we have created a robust and adaptable navigation system capable of tackling complex real-world scenarios.

Mario's Circuit showcased our ability to respond to local geometric features and track multiple lines, laying the groundwork for more advanced navigation tasks.

Through iterative refinement, we optimized our approach, focusing on enhancing control systems to overcome processing limitations and improve accuracy.

Meanwhile, Luigi's Mansion demonstrated our proficiency in addressing intricate challenges such as localization and path planning, enabling precise navigation through dynamic environments. By leveraging trajectory frameworks and localization capabilities, we developed a comprehensive solution capable of navigating seamlessly in real-world spaces.

Looking ahead, there are several key areas that warrant further exploration and development. Firstly, continued refinement of our control systems will be essential to enhance the vehicle's adaptability and robustness in diverse environments. Additionally, ongoing research into advanced color segmenting of the lanes as well as racing control systems beyond PD into perhaps a solid pure pursuit. Secondly, the refinement of algorithms regarding path planning and localization due to the dead zones found resulting from the lack of distinctive features in the environment.

Furthermore, expanding the scope of our testing to include more complex environments and challenging scenarios will provide valuable insights into the performance and limitations of our navigation system. Collaborating with experts in related fields such as computer vision and robotics can also provide valuable expertise and perspectives for further advancement.

5 Lessons Learned

5.1 Isabella

I'm very thankful I decided to take this class as it was truly my first (and only) class where I've gotten to work on something so hands on. As a Course 6 who is graduating in a few weeks, I feel like this class really taught me what it's like to work with a real working piece of hardware like a robot. I'm very thankful for this hands on experience and for having such a great team, even if it did take many hours. I feel as though this class embodies the MIT experience and I'm very happy I get to take that with me before I go.

5.2 Spring

Working on this final challenge proved to be quite demanding. From a technical perspective, immersing myself in color segmentation, various processing techniques used in real-world scenarios has been incredibly enlightening, and different control sequences. The hands-on experience of working directly with the robot and observing how our code directly impacts its behavior has been deeply frustrating but as well as immensely gratifying. On the aspect of collaboration, I am deeply grateful for the chance to operate within a team environment where I could rely on my teammates to execute their responsibilities. Given the

magnitude of the task at hand, it was imperative to divide into specialized sub-teams, each focused on distinct components. This division of labor necessitated a high level of trust and accountability among team members to ensure completion by the challenge deadline. I take immense pride in our team's group work and am thankful for our collective dedication.

5.3 Jace

Getting the robot to navigate the racetrack for the final challenge proved to be one of the most taxing experiences I've encountered recently. Along the way, we faced numerous obstacles and spent many late nights tweaking our approach to refine the algorithm before the deadline. The tight schedule also impacted our ability to fully charge the car's batteries, resulting in considerable downtime waiting for them to recharge. Furthermore, shutting down the robot to charge meant additional delays, as the camera required time to warm up after being turned back on. Variations in lighting conditions also posed challenges, affecting the consistency of the algorithm's results. Despite these challenges, I remained focused and motivated, thanks to the unwavering support from my teammates Mark and Spring, who were instrumental in the racetrack challenge. A special thanks to Kristoff and Isabella as well, who were pivotal in refining our localization algorithm and ensuring our robot could successfully navigate the city. Without my team, I couldn't have made it through to the end of this demanding challenge.

5.4 Kristoff

Working with my team was a true joy, as much fun as it was to learn the ins and outs of our car. I grew to rely increasingly on our collaborative efforts, and because of that, I saw excitement in otherwise mundane tasks like debugging. I also realized that keeping the long-term implementation goals in mind made a big difference in mustering the energy to push forward on code. I'm excited by the potential applications of the techniques I learned in the future.

5.5 Mark

Completing the final challenge was definitely one of the msot extensive projects I've completed in a class at MIT. From a technical perspective, I really learned a lot about color segmentation and the variety of processing techniques employed in the real-world to complete very similar tasks. Being able to work hands-on with the robot and see how our code affects its behavior was a very rewarding experience. From a CI perspective, I really appreciate the opportunity to work in a team environment where I could trust my teammates to complete their tasks. Due to the scale of the challenge, it was necessary to split into dedicated sub-teams that focused on each of their components. This separation of tasks required us to place a lot of trust and responsibility in each other to be able to complete the work by challenge day. I am very proud of the performance of our

team, and appreciate our willingness to rally together for over 12 hours straight in order to perform well on race day.