Lab #3: Wall Follower

Team #7

Edward Cheng
Audrey Lee (Editor)
Christopher Liem
Drew Tufto

Robotics: Science and Systems

March 15, 2025

1 Introduction

Drew Tufto

The transition from a simulated environment to real-world deployment is a huge obstacle in system development, especially for systems designed to interact with humans, such as our car. Now, we are unable to fix a bug or stop a collision by simply refreshing the simulator. This lab addresses this essential transition through two objectives. The lab first aimed to optimize each team member's independently developed wall-following approaches into a single algorithm capable of operating reliably under varying conditions. Second, the lab tasked us with designing and implementing a safety controller to prevent our racecar from colliding with obstacles or people, ensuring operational safety. These objectives are foundational steps toward creating a fully autonomous racecar capable of navigating complex environments safely and efficiently. Moreover, our safety controller will be utilized in all future labs, which compounds our need for a good design.

Our wall-follower utilizes 2D LiDAR data and a PD control strategy to maintain the desired behavior at variable speeds and poses. Its capabilities were tested methodically in different environments. The implemented safety controller functions concurrently with this algorithm, overriding driving commands as necessary to prevent collisions without hindering the cars functionality. Both the wall-follower and safety controller yielded satisfactory metrics, and our team is confident we have laid a solid foundation for future labs.

2 Technical Approach

2.1 Wall Follower

Edward Cheng

The goal of the wall-following algorithm is to enable the vehicle to follow a wall on either side at any distance, regardless of how the wall's shape or turns change. To achieve this, the vehicle first needs to determine the wall's position relative to itself, which is done by analyzing LiDAR data to estimate the walls location. Once the wall is identified, a Proportional-Derivative (PD) controller adjusts the vehicle's steering to maintain the desired distance, ensuring smooth and consistent wall following.

2.1.1 Technical Approach

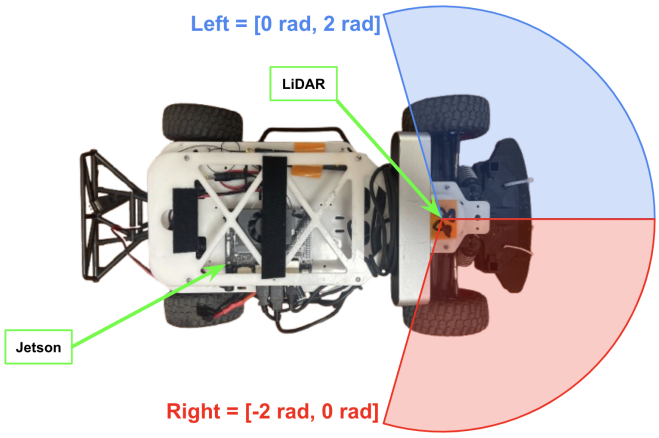

Figure 1: **LiDAR data segmentation for wall following**. The key components of our robot are the Jetson computer and the LiDAR. The LiDAR data is split in half depending on which side the wall is on.

LiDAR measurements are provided in polar coordinates (θ, r), representing the angle and range, respectively. As illustrated in Figure 1, LiDAR data is divided into two halves based on the angle relative to the car's orientation. The left half ranges from 0 to 2 radians, while the right half ranges from 0 to -2 radians. Depending on whether the vehicle is following the left or right wall, only points from the corresponding half are considered.

To ensure robustness, LiDAR points exceeding ten times the desired following distance are filtered out. This prevents the vehicle from reacting prematurely to distant obstacles or variations in wall geometry. Figure 2 visualizes this filtering, with white points representing the complete LiDAR data set and red points indicating those collected for wall estimation.

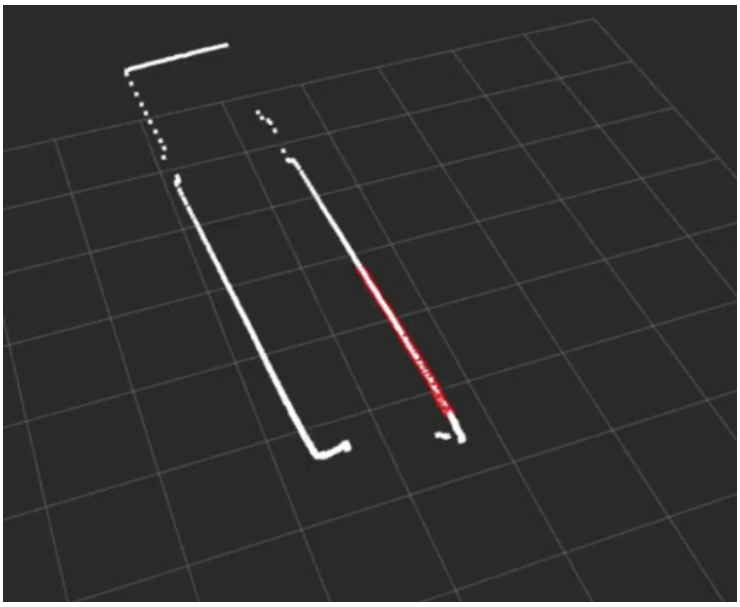

Figure 2: **Visualization of filtered LiDAR points** The white points make up the complete LiDAR data set while the red points are our points of interest for wall estimation.

Each retained LiDAR point is converted from polar coordinates to Cartesian coordinates using:

$$\begin{aligned} x &= r\cos(\theta) \\ y &= r\sin(\theta) \end{aligned} \tag{1}$$

A linear regression is performed to estimate the wall's position, represented by:

$$y = mx + b \tag{2}$$

The perpendicular distance between the vehicle and the estimated wall line is calculated using:

$$d = \frac{|b|}{\sqrt{m^2 + 1}} \tag{3}$$

Thus, the error indicates how far the vehicle deviates from the desired distance:

$$\text{error} = \text{desired\_distance} - d \tag{4}$$

The steering command that minimizes this error is determined by PD control:

$$\text{steering} = K_p \cdot \text{error} + K_d \cdot \frac{d(\text{error})}{dt} \tag{5}$$

Here, K_p and K_d represent proportional and derivative gains, respectively.

After experimentation and tuning, the parameters were finalized as $K_p = 2.5$ and $K_d = 0.5$.

The described method was selected for its flexibility, efficiency, and effectiveness in real-time control scenarios. Unlike the hardcoded method (if-else logic) that requires manually specifying steering behaviors for each type of turn, the PD-controller dynamically adapts to various wall configurations and distances, providing robust performance across diverse conditions. Although our vehicle exhibits some difficulty during inside turns, it performs well during outside turns, demonstrating precise and reliable control performance. Moreover, the vehicle quickly corrects its orientation, reducing deviations efficiently and exhibiting minimal oscillation.

2.1.2 ROS Implementation

The ROS Node **WallFollower** consists of one subscriber and two publishers: `lidar_subscriber`, `drive_publisher`, and `line_publisher`.

1. **lidar_subscriber** subscribes to the `/scan` topic and receives `LaserScan` messages. Upon receiving data, its callback function (`compute_least_squares_line`) processes the incoming LiDAR data by selecting points on the relevant side of the vehicle and then filtering out points beyond a specific distance threshold. Next, the function fits a linear approximation (using one-degree `np.polyfit`) to these filtered points. Using the derived slope (m) and intercept (b), the function calculates the distance between the vehicle and the estimated wall, computing an error relative to the desired distance. Finally, the function employs tuned proportional (K_p) and derivative (K_d) gains to determine the appropriate steering angle.
2. **drive_publisher** publishes an `AckermannDriveStamped` message with the calculated steering angle and a predefined speed to the `/vesc/high_level/input/nav0` topic.
3. **line_publisher** publishes the filtered LiDAR points obtained in `compute_least_squares_line` for visualization and further tuning.

2.2 Safety Controller

Audrey Lee

The goal of the safety controller was to reliably stop the racecar at a safe distance from any obstacle while still preserving the robots high-speed driving and sharp turning capabilities. We prioritized a minimal initial implementation for easy fine-tuning, with adjustable parameters to enhance flexibility and adaptability.

2.2.1 Technical Approach

Our basic implementation is based on predicting collisions using kinematics:

$$d_{pred} = \text{AckermannDrive.speed} \times \Delta t \tag{6}$$

We calculate the index i of the LaserScan value of interest via (7), and stop if the range registered at i is closer than d_{pred}, indicating that we would run into some obstacle.

$$i = \left\lfloor \frac{\text{AckermannDrive.steering\_angle} - \text{scan.min\_angle}}{\text{scan.angle\_increment}} \right\rfloor \tag{7}$$

This initial implementation had two major issues:

1. **Due to the latency of the robot coming to a full stop, using d_{pred} alone still caused collisions.** We modified the stopping condition to use $d_{safe} = d_{pred} + d_{tolerance}$, where the added tolerance term provided the necessary buffer.

2. **A single LaserScan datapoint is susceptible to noise**. We instead check a range of points determined by (8) and, instead of comparing distances, we compare the resulting percentage ("rating") to a tunable threshold.

$$\begin{aligned} i_{\text{start}} &= \frac{(\text{AckermannDrive.steering\_angle} - \text{ang\_range}) - \text{scan.min\_angle}}{\text{scan.angle\_increment}} \\ i_{\text{end}} &= \frac{(\text{AckermannDrive.steering\_angle} + \text{ang\_range}) - \text{scan.min\_angle}}{\text{scan.angle\_increment}} \\ \text{rating} &= \sum_{i=i_{\text{start}}}^{i_{\text{end}}} (\text{scan.ranges}[i] < d_{\text{safe}}) \;/\; (i_{\text{end}} - i_{\text{start}}) \end{aligned} \tag{8}$$

Our final design decisions were intended to ensure more realistic motion prediction:

1. **When the robot receives a new command, there is a latency period before this new command takes effect, in which the robot continues executing the previous command**. Thus, at time t, we make decisions based on $\text{AckermannDrive}_{t-1}$ instead of AckermannDrive_t.

2. **The steering angle may require an unrealistically sharp turn (e.g. 0° to 100°).** To obtain a more realistic heading estimate and reduce the effect of sudden turn commands, we take a weighted average of the last five steering angles.

$$\angle_{\text{pred}} = \sum_{i=1}^{5} (w_i \times \angle_{t-i}) \;/\; \sum w_i \tag{9}$$

A diagram of our final safety controller architecture is shown below in Figure 3. The variables highlighted in blue represent tunable parameters. We determined their final values through manual parameter sweeps, evaluating each set qualitatively. Our final parameter values were: $\Delta_t = 0.25s$, ang_range $= 0.1$ rad, $d_{\text{tolerance}} = 0.25m$, threshold $= 10\%$, and weights $= [0.1, 0.2, 0.3, 0.4, 0.5]$

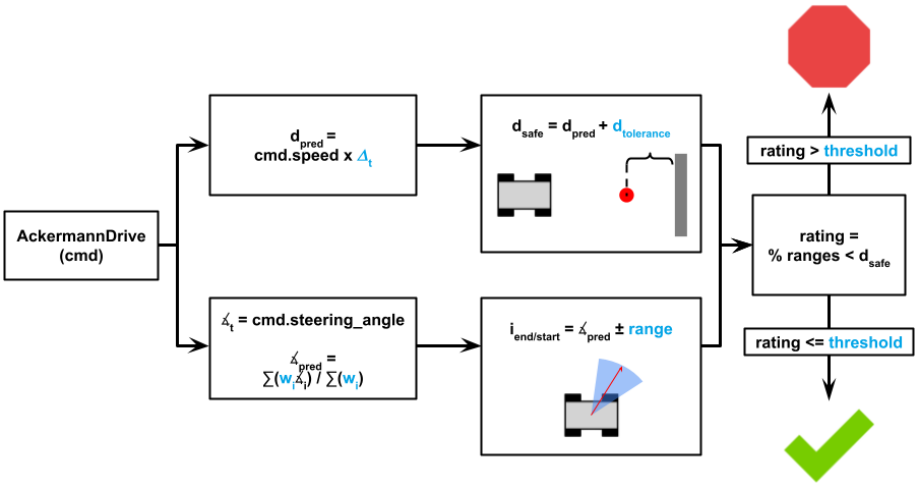

Figure 3: **Flowchart of "Are We Safe?" decision-making process**. The arrows show the flow of variables from our input steering command to our stop decision. The variables in blue are our tunable parameters.

2.2.2 ROS Implementation

The ROS Node **SafetyController** consists of two subscribers and one publisher: `command_subscriber`, `lidar_subscriber`, and `safety_publisher`.

1. **lidar_subscriber** subscribes to the `/scan` topic and receives `LaserScan` messages. Upon receiving data, its callback function saves the received angles and ranges as attributes to be used when parsing a drive command.

2. **command_subscriber** subscribes to the `/vesc/low_level/ackermann_cmd` topic and receives `AckermannDriveStamped` messages. Upon receiving data, its callback function determines whether the robot needs to stop via the system architecture in Figure 3.

3. The **safety_publisher** publishes `AckermannDriveStamped` messages to the `/vesc/low_level/input/safety` topic when the safety controller is triggered. The key field of this message is that the `speed` is set to 0.0.

3 Experimental Evaluation

Chris Liem

3.1 Wall Follower

Our initial setup used different parameters for following the left and right wall. In particular, $K_p = 0.3$ and $K_d = 0.25$ if we were following the right side and $K_p = 10.0$ and $K_d = 0.1$ otherwise. We believed we could improve our robot's

robustness by identifying parameters with uniform performance on both sides. To test this hypothesis, we conducted tests while following a straight wall at 1 m/s.

We initialized $K_d = 0$ and the desired distance $D_d = 1.0$m to choose a value of K_p.

K_p	Avg. Error (L) (cm)	Avg. Error (R) (cm)
0.5	5.50	8.88
1.0	5.50	6.50
1.5	5.27	6.18
2.0	6.34	5.81
2.5	5.37	5.40
3.0	6.01	6.28
4.0	5.81	4.61

Figure 4: **Table for tuning K_p.** This table shows the average error of the robot following a wall from one meter away at different K_p values. $K_p = 2.5$ provided the best results

According to Figure 4, $K_p = 2.5$ is optimal. There is a downward trend in average error on both sides until we reach a local minimum of 5.37/5.40 cm at $K_p = 2.5$. For $K_p < 2.5$, the robot generally struggled to adjust quickly which makes clearing turns difficult. At $K_p = 3$, the average error spikes to 6.01/6.28 cm, likely due to the robot beginning to overshoot. Moreover, the average error between the left and right sides is nearly symmetric at $K_p = 2.5$, which is desirable for the robot's robustness to the environment and trajectory. K_p values like 2.0 and 4.0 are undesirable due to the extreme differences in the left and right performance.

After choosing $K_p = 2.5$, we tuned the K_d value with $D_d = 1.0$.

K_d	Avg. Error (L) (cm)	Avg. Error (R) (cm)
0.00	5.37	5.40
0.25	3.60	5.62
0.50	4.53	4.70
1.00	4.89	3.86

Figure 5: **Table for tuning K_d.** This table shows the average error of the robot following a wall from one meter away with $K_p = 2.5$ at different K_d values.

Although $K_d = 0.25$ yielded the best performance on the left side, it also yielded the worst performance on the right side. On the other hand, increasing K_d to 0.5 led to the most symmetric behavior with a lower average error than $K_d = 0$,

so we chose $K_d = 0.5$ as our final value.

Tuning K_i was unnecessary as we had already converged to zero error. We now compare our tuned controller with our initial one.

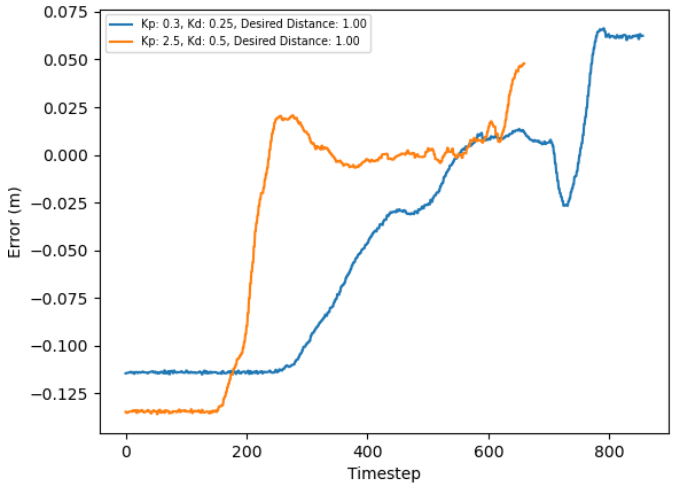

Figure 6: **Error of the tuned controller and error of the initial controller over time while following a straight wall one meter away.** Notably, our tuned controller adjusts to the initial error quicker and is steadier once it reaches zero error.

Our robot's performance improved after tuning. Our new controller converges to steady-state zero error in approximately 150 timesteps, even with a higher initial error whereas our initial controller never converged. Our final product is capable of following a straight wall.

The following figures demonstrate our robot's performance while following a loop in Stata basement. The loop was chosen for its narrow path and variety of turns.

Figure 7: **Robot following an open turn**. The robot successfully completes this open turn.

Figure 8: **Robot in a narrow corridor**. The robot is robust to narrow corridors with little room to turn.

Figure 9: **Real-time error as our robot traverses a loop at 1.5 m/s.** The histogram plots the robot's final errors and scores, where scores are calculated using the metrics given in Lab 2 (score $= \frac{1}{(1+(\alpha * \text{loss})^2}$ where α is a tuned parameter).

Desired Distance (cm)	Avg. Error (L) (cm)	Avg. Error (R) (cm)
76	5.37	5.98
100	3.60	4.53
128	4.53	8.48

Figure 10: **Average error of the robot's path at different distances**. The errors are consistently low with a velocity of 1.0 m/s

3.2 Safety Controller

To evaluate the robustness of our safety controller, we conducted experiments starting from various initial robot orientations and velocities. Figure 11 and Figure 12 illustrate the effects of orientation and velocity, respectively.

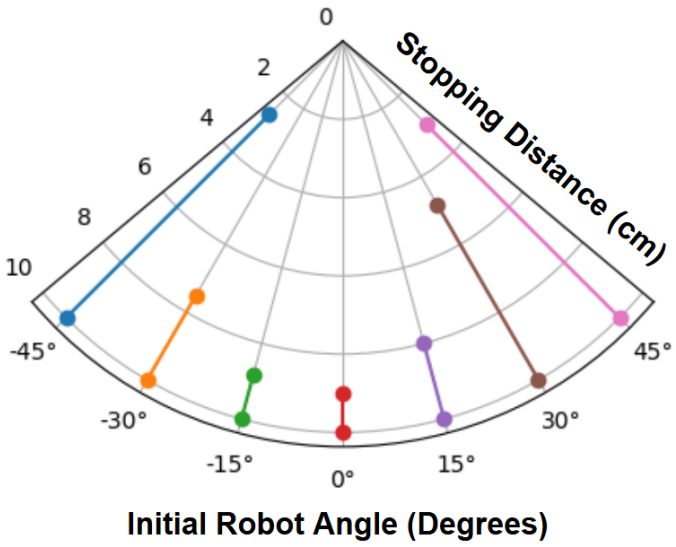

Figure 11: **Stopping distances vs. initial robot orientation**. The wedge represents the robot's possible starting angles, ranging from $-45°$ to $45°$. The robot performs well from $-30°$ to $30°$ but barely avoids collision at more extreme orientations.

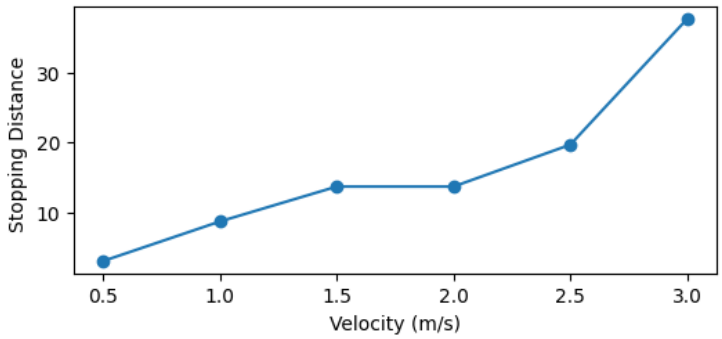

Figure 12: **Stopping distances vs. initial robot velocity**. The robot performs well up to 2.0 m/s but becomes too cautious at higher initial velocities.

Although the robot successfully stopped in all conditions, this evaluation highlighted several areas of improvement. When our robot is placed into real race scenarios, it will often encounter extreme angles and high velocities. As such, future work on our safety controller must improve performance in these extreme cases.

4 Conclusion

Drew Tufto

In this lab, we successfully transitioned our wall-following and safety controller algorithms from simulation to real-world deployment. Our wall follower, utilizing LiDAR distances and a PD control approach, demonstrates effective performance across varying speeds and wall configurations. Although some challenges persist during inside turns, the algorithm reliably maintains desired distances and swiftly corrects deviations with minimal oscillation. Moreover, our safety controller effectively prevents collisions by stopping the vehicle at safe distances from obstacles, all while maintaining our wall-follower's performance.

Moving forward, there remains room for further optimization. Improving our wall follower's responsiveness during quick inside turns would enhance overall performance. Additionally, our safety controller logic could continue to be refined to reduce its restriction on the wall-following algorithm. Future labs will also integrate feedback provided during our presentation, particularly regarding our programming methodology. We plan to be more modular and descriptive in our code. These critiques, along with technical refinements, will ensure that our racecar continues to perform robustly and safely and that our team functions cohesively in upcoming labs.

5 Lessons Learned

5.1 Edward Cheng

I learned more about controlling the car using PID values. Previously, in the individual lab, I brute-forced the three parameters by testing different combinations. This process took a lot of time, and the results werent great either. However, in this real car lab, with my teammates' help, I gained a deeper understanding of how each parameter contributes. For example, Kp helps correct errors, while Kd stabilizes the car.

In a project of this scale, with multiple components, I also realized the importance of unit testing. Our car consistently had different PID values for the left and right sides, and none of us understood why. We kept tuning the PID values without making progress. Eventually, we discovered that dust on the LiDAR sensor was causing inconsistent readings between the left and right sides. If we had conducted unit tests early on, we could have quickly identified and fixed

the issue.

I also appreciate how responsible and passionate my team was about the lab. Everyone was willing to invest time and effort to achieve strong performance, even though we only met as a full group twice. One area we could improve is delegating tasks more efficiently to maximize productivity. At times, we found ourselves overlapping in certain areas while neglecting others. If we had clearly assigned rolesfor instance, having one person focus on hardware while another on software could have streamlined our workflow and made faster progress.

5.2 Audrey Lee

This lab was particularly challenging for me as I was out of the country for most of the allotted work time, which required me to learn how to coordinate with my team to ensure meaningful contributions. I am incredibly grateful to my team for being so accommodating and allowing me to take charge on the remote/simulated components. Also, given the time difference, we were rarely working at the same time, so learning how to leave clear and actionable questions and detailed progress reports for them to read during their working hours was crucial for ensuring that we were all on the same page.

In terms of the technical aspects, this lab really highlighted the importance of easily replicable metrics and evaluation methods. For example, our team resorted to a guess-and-check method for tuning the parameters on the right side of our wall follower when it didn't behave as expected. However, after taking the time to implement a graphical error log, it was clear that the issue was not in our code or parameters but with the LiDAR itself. Im sure that this would have been identified much earlier if we had adhered to good evaluation methods from the beginning.

While I have a lot of experience working with teams in highly structured environments, I believe this project was a key lesson in how effective communication and robust experimental methods can significantly improve a team's efficiency.

5.3 Christopher Liem

From a technical standpoint, I learned how to translate software that worked in simulation into the real world. Simulations are idealistic environments, so code that works in theory may produce stochastic behaviors in the real world. This was the case for my wall follower. When the team first met, we compared scores for lab 2 to decide whose code to use as a baseline for lab 3. I had one of the higher scores for lab 2, yet it resulted in awful performance on the real robot. We used Edward's code as a baseline and tweaked his parameters for our deliverable instead, which I am grateful for because it saved us time.

From a communications perspective, I learned how to work on a big project with a team. Prior to this lab, I was always scared of working with a team because I usually worried about my teammate's capabilities and willingness to work. This lab was a "breath of fresh air" for me. My teammates are capable, responsible, and experienced in diverse fields. We were confident in assigning tasks to each other knowing that they would get would get it done. Moreover, we were always ready to answer each other's questions and willing to work outside of class time when we were available. We had complications with business trips and availability misalignments, but we successfully submitted a working prototype on time.

Overall, developing a wall-following robot in real life with a team was an enjoyable experience, and I look forward to rest of the labs this semester.

5.4 Drew Tufto

I believe that our team performed well in all aspects of the bag, but the feedback we received from our presentation, both in technical and communication areas, will be very helpful in future labs. For example, we did not have large issues with our debugging processes, but implementing unit tests will surely streamline the debugging process as we move further in the course.
A specific area where our team can improve is making our code more readable, specifically with comments and docstrings. This would benefit us in technical, communicative, and collaborative aspects, as readable code is easier to debug and translate into a presentable form.