Optimizing Soil-Based Robotic Navigation Through PID Control

Aayush Anand
aayush3198@gmail.com

Abstract

As the need for agricultural processes grows with an increasing population, robotics presents the opportunity to improve efficiency through precise soil navigation. The integration of robotics in agriculture offers the potential to develop a more efficient navigation system that relies less on human involvement and more on robots to not only monitor crops but also execute tasks that would otherwise be tedious or impractical. By reducing human intervention and utilizing robotic systems to address these agricultural needs, resources can be managed more effectively as well. Unfortunately, many environmental factors such as soil composition, temperature, vegetation, salinity, and changing weather patterns pose many challenges to accurately navigating a soil terrain. To address these challenges, we propose the implementation of a Proportional-Integral-Derivative (PID) controller to enhance adaptability in soil navigation systems through continuous monitoring. We will simulate a soil-like terrain using Gazebo and test the PID parameters on a ROS-based TurtleBot to find what is most suitable to improve the performance of the robot and achieve stable navigation.

I. Introduction

Autonomous crop monitoring through robotics has emerged as a transformative approach in agricultural settings that allows for precise data collection and efficient execution of tasks that would otherwise require human involvement, which may be insufficient [8]. These robotic systems can perform a wide range of vital functions, including environmental data gathering, seed planting, mowing, spraying, weed control, crop readiness detection, crop harvesting, etc. [4] Furthermore, they can also be applied to mapping terrains and soil inspection, creating a comprehensive foundation for agriculture which would be useful for farmers when it comes to planning [1]. However, ensuring reliable and accurate navigation from an initial starting position to a target destination poses considerable challenges. Robotic systems must not only avoid obstacles but also navigate the complexities of soil-based movement, which introduces additional issues. Soil conditions can vary widely with factors like moisture content, composition, compaction, and surface irregularities (e.g. rocks, plant material, debris) [3] which affect a robot's stability and precision. These environmental variables mean that robotic systems must continuously adapt to maintain effective navigation, highlighting the need for advanced control algorithms and sensing mechanisms to accommodate a dynamic terrain.

Navigation for robots includes four key components: perception, localization, cognition, and motion control [2]. Together, these components enable robots to effectively interpret sensor data, position themselves accurately, make informed decisions, and maneuver safely within their environments. For successful navigation, robots rely on sensor information to determine safe and efficient trajectories while

avoiding obstacles [3]. This becomes imperative as their surroundings change while in motion. Perception includes collecting and interpreting this sensory information which allows robots to estimate the distance between themselves and obstacles in their way so that they know where to avoid and can compute a new trajectory. Localization which refers to the robot's awareness of its position and orientation within a given environment, is necessary to compute these trajectories as well so the robot is aware of its position [1], [2]. Cognition allows the robot to take in its surroundings to make instantaneous decisions regarding navigation. Adapting to new information is critical, especially in uncertain environments that can't be predicted ahead of time. Agricultural robots are often times larger and move slower due to the nature of their capabilities. Therefore, motion control becomes very important to ensure that these robots remain stable and avoid tipping over [2].

The control system of a robot commonly consists of certain features. These include reactivity, robustness, sensor integration, modularity, expandability, and adaptability [21]. The control system is what allows robots to function in complex, dynamic, and partially unknown environments [21]. A key component of the control system that can be utilized is PID control, which is a closed-loop controller that consists of three parameters that help maintain stability in a robot's movement and minimize overshoot [12], [13], [14] ensuring reliable performance. These parameters are the proportional, integral, and derivative values which are used for feedback.

II. Problem Description

Unknown terrain may interfere with the functionality of a robot in an agricultural environment because current approaches don't often take into account the density of the soil and its structure which has an effect on how these robots need to move [5]. Therefore, to improve robustness, we integrated a proportional-integral-derivative (PID) controller [6], [12] with a turtlebot using ROS and Gazebo simulations to test what PID values result in the smoothest navigation. In a PID controller, the proportional term is used to calculate the current errors, the integral term gives information about previous errors that have occurred, and the derivative term is able to give a prediction of future errors that may occur [10]. This controller will enhance robot navigation which is crucial for accurately determining where the robot should collect its information. By fine-tuning the robot's movement, the PID ensures precise navigation, which will lead to more accurate path following and obstacle avoidance. The PID controller works by continuously collecting data and finding the error between the actual data and what it was expected to be. This helps to reduce the variability of sudden changes that may occur in the field and helps prevent future errors that may occur based on past trends [6], [7]. We will need to fine-tune the PID parameters to find the most optimal settings for navigation on a soil-like terrain.

To determine optimal PID parameters for soil navigation, we utilize a ROS-based TurtleBot in a simulated Gazebo environment. The simulation provides a controlled but realistic setting to test various PID values on a model of uneven terrain, replicating the physical conditions that may affect movement on soil. By adjusting the proportional, integral, and derivative gains, we aim to observe the robot's performance metrics, such as linear and angular velocity, response time, and accuracy in navigation. Through iterative testing and data analysis, we will identify the PID settings that enable the TurtleBot to achieve smooth, stable navigation with minimal deviation from its intended path and target position. This approach allows us to systematically refine the control parameters by seeing how varying values cause different trajectories, ultimately developing a robust, soil-specific navigation strategy suitable for real-world applications in agriculture.

To simulate the soil-like terrain in Gazebo, we used an existing heightmap to represent varying heights, which will simulate the unevenness of the soil [23]. Using heightmaps in Gazebo enables us to approximate soil-like conditions, capturing the irregularities found in agricultural environments [16]. The TurtleBot is also encapsulated in a hexagonal area with nine obstacles placed evenly throughout to simulate crops in a field. By looking at previously implemented PID controllers, we were able to make modifications for our specific application [24], [25] which we have linked in the appendix. Additionally, when the TurtleBot comes within 0.5 meters of an obstacle or the wall, it will use an obstacle avoidance trajectory generation of continuously turning right until the obstacle is no longer present instead of using the PID values. For each test scenario, the TurtleBot is placed at a starting location and the target position is defined, so we can observe its ability to maintain a stable path on the irregular surface. By running multiple iterations, each with different PID values, we analyze how the adjustments affect the robot's speed, stability, and error correction during navigation [15], [17].

This study tries to find the optimum PID parameters for soil-based navigation in agricultural robots, establishing the precision benchmark for crop monitoring tasks. The project will support more reliable data collection in varied outdoor conditions through increased stability and accuracy of movement, improving the robot's capacity for tasks like moisture sensing, weed detection, and crop analysis [18]. This will also support future work of testing in the field on actual soil terrain to validate the results and optimize the parameters under real-life conditions [19]. The ability to have a simulated environment will afford fast prototyping since changes to the PID controllers can always be directly applied and iteratively tuned for much more efficient calibration before deployment into real life [20].

III. Related Work

Previous works have sought to address these navigation challenges through various approaches, including through the use of sensors. These include the Inertial Measurement Unit (IMU), GPS, GNSS, LiDAR, and wheel encoders [1], [4]. These have been found to be reliable for tracking the position and movement of a mobile robot. For seed placement, differential GPS units and an IMU were used for precision [1]. However, when it comes to outdoor applications, these approaches are less desirable due to the constantly changing environment. Additional technologies such as computer vision and depth cameras have been utilized to enhance localization precision, especially in scenarios where the robot is navigating through confined or obstacle-heavy spaces that demand high spatial accuracy. Although these methods improve a robot's ability to estimate orientation and position in these challenging spaces, agricultural settings often include irregular slopes and vegetation as well. These can obscure sensors and distort data, leading to errors in depth perception and providing inaccurate results [4]. In addition to this, machine vision is greatly used for navigation, however, in an outdoor environment, it becomes susceptible to natural elements [8]. For example, machine vision can be used to map crop rows improving localization [2].

Furthermore, PID control has been previously used along with computer vision to improve line-tracking robots. It does so by using the images to make sure the robot stays on track with the line while the PID control helps maintain a suitable speed [9]. Various path-finding algorithms, aside from A* which is most commonly used, have also been implemented in conjunction with PID parameters to find optimal robotic navigation [13]. For instance, a modified version of the A* algorithm, called IA* uses a heuristic function and weight penalties to reduce the time taken for the path as well as create a more stable trajectory [11]. Additionally, the Genetic Algorithm, which is a method for solving problems based on

natural selection and evolution [22], was used to tune the parameters by using the mean square error as the fitness function. This algorithm was shown to be efficient in terms of being near the desired path [13], however, it was only tested for straight-line paths. This same study attempted to use an artificial bee colony algorithm to tune the parameters as well. This algorithm works similarly to how bees forage in a colony. This means that in each iteration, the bee finds its food source and evaluates it to determine its fitness. Through this approach, they were able to find the optimal path. Compared to the genetic algorithm, they found that the artificial bee colony algorithm is more efficient and results in the smallest distance error [13].

$$(t) = K(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt})$$

Figure 0: PID control equation [17].

The PID controller uses feedback as a way for robots to improve their performance by continuously making changes [17]. Figure 0 shows the equation for the PID controller. More specifically, the proportional term means that the action is proportional to the error. The integral action ensures that the output is correct with the setpoint and eliminates any errors that may have occurred in the steady state with just the proportional term. The derivative action helps the system notice changes in error that will help make future errors more easily detectable [17]. Together these three components can work together to achieve stable robot navigation.

IV. Results & Insights

To determine the success of our results, we outputted certain statistics to a CSV file so we could create line graphs to visualize the trends in error over time. These included the time, x position, y position, distance error, linear velocities, and angular velocities. In addition to the trends in error, we were also able to visualize the incremental changes in linear and angular velocity, which helped with determining which PID parameter values resulted in the smoothest trajectories. The process for finding these PID parameters was initially trial and error. We started with low values for the linear and angular proportional parameters while keeping everything else at 0. This allowed us to see how fast the turtlebot was moving and turning before introducing any other variables. This data-driven approach allowed us to incrementally change the parameters to achieve a more stable trajectory and see how each component was affecting the navigation. In this section, we will show three scenarios of varying PID values to support our selection of the optimal values.

Figures 1A-C show the distance error, linear velocities, and angular velocities, respectively. Looking at these graphs, we can see that the turtlebot was unable to reach its goal destination as it remained at a distance error of about 4.0 for over five seconds before we terminated the process. This can likely be attributed to the robot being stuck in a position it was unable to get out of due to varying ground heights. The oscillation of the linear and angular velocities corroborates these findings by appearing stable at a nonzero value. This indicates that the robot was attempting to move and turn around but was unable to actually do so.

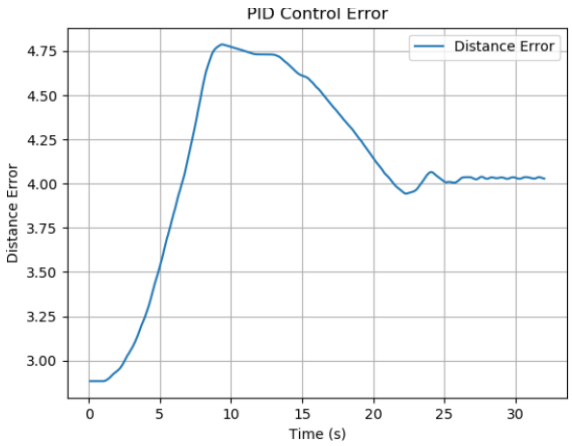

Figure 1A: PID control distance error for linear and angular proportional values of 0.05 and values of 0 for all other parameters.

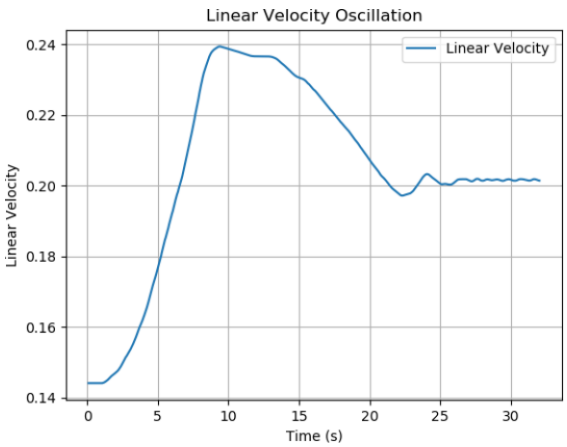

Figure 1B: Linear velocities for linear and angular proportional values of 0.05 and values of 0 for all other parameters.

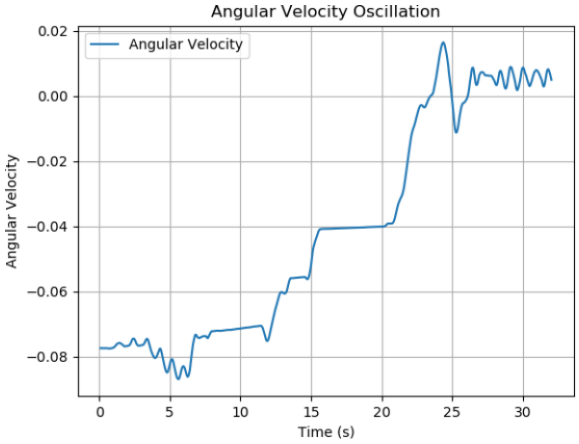

Figure 1C: Angular velocities for linear and angular proportional values of 0.05 and values of 0 for all other parameters.

Using these results from the initial PID parameters, we increased the proportional parameter for both the linear and angular values. With this, we observed that the turtlebot moved very quickly in large circles which resulted in it crashing into the wall. Once it crashed into the wall, it was unable to turn to free itself. From here, we decided to lower the values and make the derivative value nonzero for the linear component. The robot appeared to still crash into obstacles however it was not moving quite as fast. We were able to deduce that we needed a small p-value and d-value to ensure that the robot would move at a reasonable pace. We then decided to change the i-value to see what effect it would have on the movement.

Figures 2A-C show the output for PID values where linear p=0.10, i=0.02, d=0.05 and angular p=0.10, i=0, d=0.02. Figure 2A shows that the error distance during the trajectory is not consistent and the robot never reaches the target position even after an extended period of time. Figure 2C shows spikes in the angular velocity where the robot is hitting obstacles. During this scenario, the robot was moving in circular motions instead of linearly while only turning when an obstacle was in its path which made it more prone to collisions. It also appeared that the robot never slowed down as shown by Figure 2B, even when the error distance was becoming smaller, indicating that we needed to increase the derivative values for both the linear and angular components.

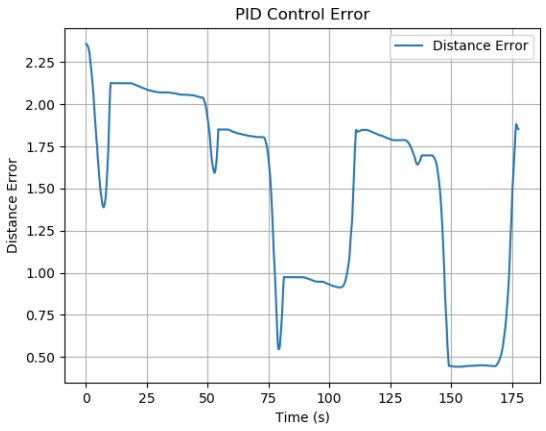

Figure 2A: Distance error for linear values of p=0.10, i=0.02, d=0.05 and angular values of p=0.10, i=0, d=0.02

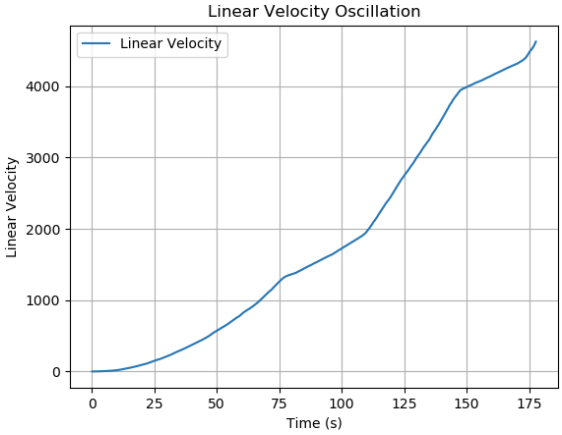

Figure 2B: Linear velocities for linear values of p=0.10, i=0.02, d=0.05 and angular values of p=0.10, i=0, d=0.02

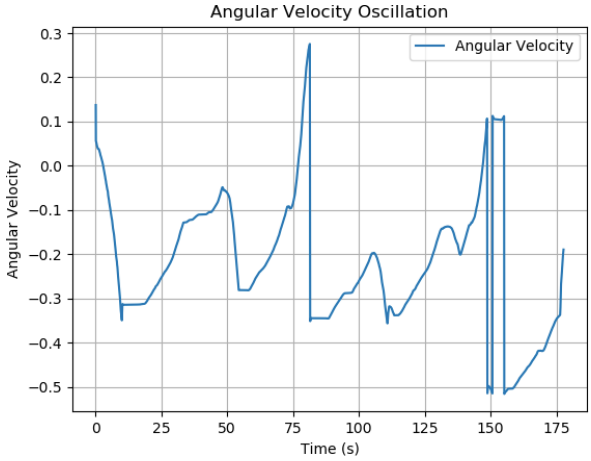

Figure 2C: Angular velocities for linear values of p=0.10, i=0.02, d=0.05 and angular values of p=0.10, i=0, d=0.02

After running many simulations, we were able to find certain trends that occurred and which sets of values worked the best. Figures 3A-C show what we found to be the optimal PID values where linear p=0.14, i=0, d=0.08 and angular p=1.10, i=0, d=0.04. First, looking at 3A, we can see that the distance

error becomes 0 after about 85 seconds, however, it appears to be approaching an error of 0 for quite a bit of time. This is due to the derivative value helping to slow the robot down as it approaches the target position so it does not move too fast and pass the target. In Figure 3C, we can see the angular velocity fluctuates a lot around 15 to 30 seconds. During this time, the robot found an obstacle in its path, so it was continuously turning and attempting to move forward until the obstacle was no longer in its way. The distance error around this time also helps to show that the robot was continuously attempting to take the shortest path to the target position instead of simply turning more and continuing in a different direction.

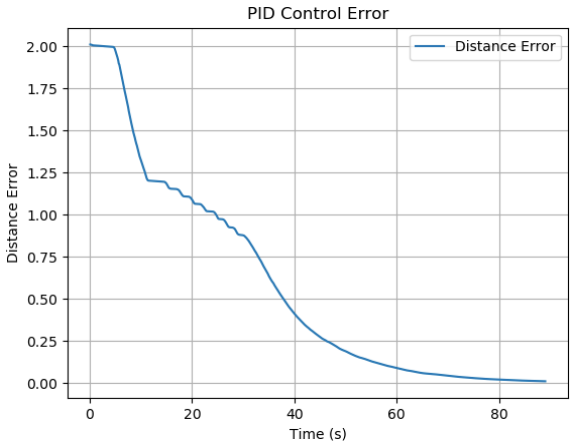

Figure 3A: PID distance error with linear values of p=0.14, i=0, d= 0.08 and angular values of p=1.10, i=0, d=0.04.

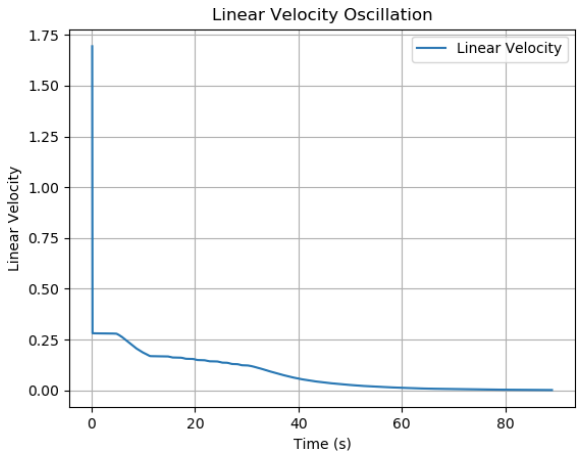

Figure 3B: Linear velocities with linear values of p=0.14, i=0, d= 0.08 and angular values of p=1.10, i=0, d=0.04.

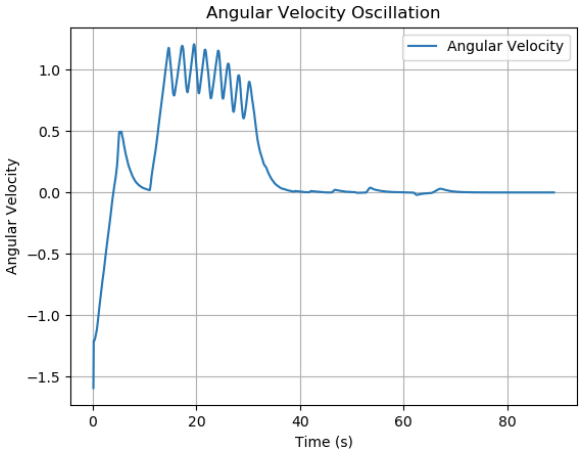

Figure 3C: Angular velocities with linear values of p=0.14, i=0, d= 0.08 and angular values of p=1.10, i=0, d=0.04.

Based on our extensive testing, we ultimately found that only the proportional and derivative parameters needed to be changed. Introducing a nonzero value for the integral value almost always caused

the robot to either crash into the wall or an obstacle and also overshoot the target position if it was able to even get to it at all. Changing this value also tended to cause the robot to travel in a path that was not efficient by moving in directions away from the goal.

While modifying just the linear proportional parameter, we found that the robot did not change orientations except when it was coming in contact with an obstacle or the wall. This is attributed to the obstacle avoidance node we incorporated that overrode the PID node when applicable. By raising the angular proportional value, we found that the robot changed directions based on what would get it closest to the target. When this value was too small, we found that the robot wasn't changing directions often enough and was deviating from the optimal path. However, making it too large caused the robot to turn too much and crash into obstacles.

The linear proportional value was determined by seeing how fast the robot was moving. We found that when the value was too small, the robot moved very slowly, but when it was large the robot would cause it to move too fast and overshoot the target. The robot would then try to correct itself, but it was often unable to get to the precise target position. Because of the varying heights, as well, we found that the robot appeared to be falling at some positions. If the proportional values were too small, the robot wouldn't be able to overcome these height differences and gather enough momentum to keep moving, which would result in it getting stuck in one position.

V. Problems Encountered

We encountered several challenges during the course of this project. Due to our lack of prior experience with TurtleBot and Gazebo, the initial setup took a bit longer than expected. We also ran into some issues with creating our own height map. Our original plan was to create a heightmap model that we could import into our environment that would represent the varying heights in a soil-like terrain. We were able to use Matplotlib to create a height map, however, adding it to the turtlebot3_world.world and model.sdf files turned out to be a challenge as the model wouldn't load. Therefore, we decided to use a preexisting sand terrain model that we found from gazebosim.org [23]. Since we had to use a model that was already made, we couldn't configure it to our needs specifically, however, because it was a sand terrain, it was very close to what we would have originally created. The terrain contained obstacles (rocks, gravel, etc.) which worked well for the uneven terrain simulation.

In addition to this, we also found that trying to find the most optimal PID parameters was quite challenging. Because there were six values we were changing, it became rather difficult to tell which values were causing certain movements especially since small changes could cause significant movements. However, as we began to run more and more simulations we were able to see how the parameter values were causing different movements and also see certain patterns occurring within the data.

VI. Lessons Learned

Throughout this project, we were able to learn a lot about the importance of control systems in robotics and how the PID control works to achieve precise and stable movement. We were able to expand on our knowledge of ROS and Gazebo and how it works, more so by trying different models to represent soil before we decided on what we were going to use for our simulations. By doing our testing through a

data-driven approach, we were able to learn about how the PID parameters work together to affect a robot's movement in different ways and also see which values worked best for our specific application. Our findings also helped support the information we found in our initial research about how PID control is used.

VII. Future Work

Even though we were able to find the PID parameters that work best on uneven terrain, there are still additional factors that would need to be considered to accurately find which parameters would be ideal. Since we only used a TurtleBot in a Gazebo simulation, we didn't simulate these environmental conditions such as vegetation and weather patterns which would likely have a somewhat significant impact on the values. For example, vegetation such as grass, weeds or plant stems can exert resistance and friction, changing the dynamics of the robot. The interaction between the robot wheels and the plant can affect power and traction, possibly requiring adjustments to the PID parameters. The impact of these factors can be better analyzed by including real plant models in the simulation or testing in real terrain.

Factors such as rain, wind, and temperature changes can affect soil and sensor performance. For example, rain will make clay or cause water to accumulate, so tighter control should be applied to prevent falling and stabilize. High winds might disturb lightweight robots or affect sensor readings, while extreme temperatures could impact the electronics and actuator response times. Future work will include simulation or real-world testing of these conditions to ensure that PID is efficient and flexible. The current study focuses on uniformity but does not consider other soil properties such as moisture content, compaction, and grain size. These properties directly affect traction and maneuverability. For example, compacted soil will provide more traction but also increase friction, while loose, sandy soil will cause the wheels to slip.

Future experiments will include field trials where the robots move on different types of soil at different moisture levels. This will help refine the PID parameters to adapt to changes in real-time. Farms are typically large and have many terrains, obstacles, and environmental conditions. Ensuring that the PID parameters remain effective across diverse sections of a field will be crucial for practical deployment. For this purpose, large-scale field tests will be conducted where the robot will navigate a heterogeneous environment including slopes, rocky areas, and flooded areas.

The development and implementation of adaptive PID control will be an important avenue for exploration. This controller uses data from the robot's sensors to adjust the PID parameters according to the environment. For example, if the robot detects an increase in slippage, it could modify the proportional term to regain stability. Machine learning techniques such as reinforcement learning can be used to estimate adaptive parameters based on historical data and real-time data. By combining this information with the robot's onboard sensors, the system can predict and adapt to environmental changes before they occur. For example, a robot that knows rain is in the forecast can adjust its navigation strategies to cope with rainy conditions.

While the current study provides a strong foundation for PID optimization on uneven terrain, future work will focus on expanding the scope to include environmental conditions such as vegetation, weather patterns, and soil variability. Through simulation enhancements, field testing, and the development of dynamic parameter adjustment mechanisms, the system can be refined to achieve reliable and efficient navigation in real-world agricultural environments.

VIII. Conclusion

This project successfully implemented a PID controller to enhance the navigation of a TurtleBot on an uneven soil-like terrain simulated in Gazebo. The smooth and much more stable navigation attained by systematic fine-tuning of the parameters of the PID testifies to the great capability that control systems can proffer on challenges thrown up by dynamic environments. The study provided a robust framework for optimizing soil-based robotic navigation, emphasizing adaptability in agricultural robotics.

Iterative testing led us to the optimal proportional and derivative gains, while the integral term was less effective for this application. Our results emphasize how data-driven methods play a crucial role in fine-tuning navigation strategies for better path following and obstacle avoidance. This project also pointed out the limitations of simulation-based testing, mainly the absence of real factors such as vegetation, weather patterns, and soil variability. These findings provide a basis for further work on field trials and adaptive PID controls. Ultimately, this research is intended to further develop robotics in agriculture, proposing practical solutions to enhance efficiency and reduce human intervention in farming operations.

References

[1] Post, M. A., Bianco, A., & Yan, X. T. (2017). Autonomous navigation with ROS for a mobile robot in agricultural fields. Paper presented at 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Madrid, Spain.

[2] Gao, X., Li, J., Fan, L., Zhou, Q., Yin, K., Wang, J., ... & Wang, Z. (2018). Review of wheeled mobile robots' navigation problems and application prospects in agriculture. Ieee Access, 6, 49248-49268.

[3] Shalal, N., Low, T., McCarthy, C., & Hancock, N. (2013). A review of autonomous navigation systems in agricultural environments. SEAg 2013: Innovative agricultural technologies for a sustainable future.

[4] Galati, R., Mantriota, G., & Reina, G. (2022). RoboNav: An affordable yet highly accurate navigation system for autonomous agricultural robots. Robotics, 11(5), 99.

[5] Luo X, Yin C, Sun Y, Bai W, Li W, Song H. A Real-Time Prediction Approach to Deep Soil Moisture Combining GNSS-R Data and a Water Movement Model in Unsaturated Soil. Water. 2024; 16(7):979. https://doi.org/10.3390/w16070979

[6] R. K. Megalingam, D. Nagalla, K. Nigam, V. Gontu and P. K. Allada, "PID based locomotion of multi-terrain robot using ROS platform," *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, 2020, pp. 751-755, doi: 10.1109/ICISC47916.2020.9171152.

[7] Yuzhu Cheng, Yong Chen and Hongxing Wang, "Design of PID controller based on information collecting robot in agricultural fields," 2011 International Conference on Computer Science and

Service System (CSSS), Nanjing, 2011, pp. 345-348, doi: 10.1109/CSSS.2011.5974664.
[8] Jia W, Tian Y, Duan H, et al. Autonomous navigation control based on improved adaptive filtering for agricultural robot. International Journal of Advanced Robotic Systems. 2020;17(4). doi:10.1177/1729881420925357
[9] Farkh, R., & Aljaloud, K. (2023). Vision navigation based PID control for line tracking robot. Intelligent Automation & Soft Computing, 35(1), 901-911.
[10] Barsan, A. (2019). Position control of a mobile robot through PID controller. Acta Universitatis Cibiniensis. Technical Series, 71(1), 14-20.
[11] Liu L, Wang X, Wang X, Xie J, Liu H, Li J, Wang P, Yang X. Path Planning and Tracking Control of Tracked Agricultural Machinery Based on Improved A* and Fuzzy Control. Electronics. 2024; 13(1):188. https://doi.org/10.3390/electronics13010188
[12] Huang M, Tian M, Liu Y, Zhang Y, Zhou J. Parameter optimization of PID controller for water and fertilizer control system based on partial attraction adaptive firefly algorithm. Sci Rep. 2022 Jul 16;12(1):12182. doi: 10.1038/s41598-022-16425-7. PMID: 35842470; PMCID: PMC9288466.
[13] Ali, R. S., Aldair, A. A., & Almousawi, A. K. (2014). Design an optimal PID controller using artificial bee colony and genetic algorithm for autonomous mobile robot. International Journal of Computer Applications, 100(16), 8-16.
[14] Balaji, V., Balaji, M., Chandrasekaran, M., & Elamvazuthi, I. (2015). Optimization of PID control for high speed line tracking robots. Procedia Computer Science, 76, 147-154.
[15] Nevludov, I., Sychova, O., Reznichenko, O., Novoselov, S., Mospan, D., & Mospan, V. (2021, September). Control system for agricultural robot based on ROS. In 2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES) (pp. 1-6). IEEE.
[16] Li, C., Xu, H., & Liang, X. (2018). "The Application of Vision-Based Navigation in Agricultural Robots: A Review." Computers and Electronics in Agriculture, 151, 366-379.
[17] Åström, K. J., & Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning*. ISA – The Instrumentation, Systems, and Automation Society.
[18] Fennimore, S. A., & Doohan, D. (2019). "The Importance of Robotics for Field Crop Weeding." *Pest Management Science*, 75(7), 1761-1764.
[19] Cruz, D., et al. (2021). "Autonomous Navigation Systems for Agricultural Robots in Outdoor Environments." *Agricultural Engineering International: CIGR Journal*, 23(4), 224-235.
[20] Koenig, N., & Howard, A. (2004). "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator." *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149-2154.
[21] Tzafestas, S. G. (2018). Mobile robot control and navigation: A global overview. Journal of Intelligent & Robotic Systems, 91, 35-58.
[22] Ayala, H. V. H., & dos Santos Coelho, L. (2012). Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator. Expert Systems with Applications, 39(10), 8968-8974.
[23] hmoyen. “Model Insertion from Fuel#.” Model Insertion from Fuel - Gazebo Ionic Documentation, Open Robotics, gazebosim.org/docs/latest/fuel_insert/.
[24] UATeam. “Python PID Controller Example: A Complete Guide.” Medium, Nov. 2024, medium.com/@aleksej.gudkov/python-pid-controller-example-a-complete-guide-5f35589eec86.
[25] “Lab 3: Closed Loop Control.” Lab 3: Closed Loop Control - UCR EE/ME144/EE283A Fall 2023 1.0 Documentation, ucr-ee144.readthedocs.io/en/latest/lab3.html.