The CleanBot 3000

Nethania Morales
Advisor: Professor Shahnam Mirzaei, Ph.D.
California State University, Northridge

***Abstract*—In collaboration with NASA Jet Propulsion Laboratory (JPL), CSUN undergrad students were tasked with the creation of the CleanBot 3000. The CleanBot 3000 is an autonomous robot with the aim/purpose to sanitize through the cleanrooms at the JPL establishment. In order to properly sanitize such rooms, the CleanBot will navigate through every square inch utilizing ultraviolet-c lights. The project consisted of a total of two sub-teams this Spring 2024 semester: the Power team and the Systems team. Both teams played an important role in the progression of the project. The Power team was in charge of power distribution while maintaining the intactness of the bot as well as keeping within JPL's guidelines. The systems team was in charge of incorporating all components through micro-controller operation, algorithm implementation, etc. As a part of the Systems team, it was vital that the team set a strong foundation at the start of the semester that would eventually help the entire group reach all their goals. Although this was a group project, it was important that each individual followed the sub-team lead, as well as the overall project lead to push the project forward. The team had to evaluate how our specific engineering solution affected one or more of the following: public health, safety, welfare, global impact, cultural aspects, social dynamics, environmental concerns, and economic effects. The team also had to evaluate the impact the CleanBot 3000 would have in global, economic, environmental, and societal contexts. Individually, I was able to contribute to the team in many ways that allowed for the group to reach its main goals.**

I. Introduction

THE CleanBot 3000 is an autonomous robot that is responsible for the sanitization of cleanrooms at JPL. The cleanrooms at JPL contain all types of flight hardware/equipment. There is constantly a high amount of personnel working in and out of the facility, as well as a high risk of microorganisms being introduced to all flight hardware stored in the cleanrooms. Because the flight hardware being worked on will travel to space and come into contact with extraterrestrial environments, these rooms must maintain a high level of cleanliness. It is important to not introduce foreign microorganisms that can disturb these extraterrestrial environments. Such microorganisms could ultimately cause a lot of harm and alter the ecosystems to a state that is irreversible. The CleanBot 3000 would help mitigate the exposure of such harmful microorganism due to its autonomous nature. The bot would substantially decrease the amount of microorganisms that could be brought in by cleanroom personnel.

This CSUN senior design project was split to have two subteams tackle all design issues pertaining to the bot. The Power subteam was tasked with designing the power distribution of the bot that maintained all of JPL's guidelines. The System subteam was tasked with incorporating all components (Lidar, motors, etc.) using micro-controller operations, a variety of algorithms, code implementations, etc. There was a leader for the entire group project, as well as a leader for each team. Both teams worked in conjunction with each other to have the bot get one step closer to reaching full autonomy. As a part of the Systems team, the goals for this semester were implementing Nav2 into the current ROS2 framework. Once Nav2 was implemented, the next goal was to research and implement a full coverage algorithm that would allow for the bot to navigate through an entire room autonomously. Lots of challenges were faced within the Systems team this semester; however, the team was able to navigate through these challenges and accomplish a substantial amount of work this semester.

II. Individual Work

One of the first goals this semester was to implement Nav2 into the current CleanBot ROS2 framework. Each person on the Systems team worked on their own personal computer to try and get Nav2 up and running. This process proved to be harder than imagined. Each team member had a different process for trying to get RVIZ and Nav2 functioning on their personal computer. Because of this, there were lots of setbacks. The solution to this was whenever team members had issues within the implementation process, we would all communicate with each other on what exact issue we were having. We would collectively try to work and troubleshoot through these issues.

Because the implementation process of Nav2 was taking longer than planned, our team lead designated the group to research the implementation of a camera as well as incorporating the ability to read QR codes. This would give the bot the ability to read and save loaded maps, cutting navigating time down drastically. Dr. Mirzaei provided us with a OV7670 camera module. [7]

Fig. 1: Utmel Electronics OV7670 Camera Module [7]

Because of the convenience of already having this camera, I started researching ways on how to connect the camera

and set it up properly. On the Utmel Electronics website, the article "OV7670 Camera Module: Datasheet, Specifications, and Comparison" was used to learn more about this camera. The article states that some applications for the OV7670 are as follows: facial recognition, document scanning, obstacle avoidance, etc. Therefore, for the purpose of scanning QR codes, this camera module was perfect. However, the same article also states that this camera is only compatible with the Arduino Uno. More research was conducted to see if there was any way implement this camera with our microcontroller. Implementing this camera into our system proved to be more troublesome, and another team member suggested we use a USB camera instead. The team did not proceed with the implementation of the camera module OV7670.[7]

To continue, a goal that was set the previous semester was to start tackling PID tuning. In the current ROS2 framework,there source code file that utilizes PID parameters. The PID parameter constants are as follows: the proportional term Kp was set to 20, the derivative term Kd was set to 12, the integral term Ki was set to 0, and the output scaler Ko was set to 50. For any robot to function properly, PID tuning is essential. Although the current parameters did not pose any issues, I believe it is important to run multiple tests changing specific parameters and see how each parameter affects the movement of the bot. The plan I had was to individually increment one of the PID parameters and see how it affects the movement of the bot. Once enough tests were accomplished, then we could have more accurate PID parameters. Before coming into this project, I had no idea what PID was. I was able to gain extensive knowledge on what each parameter means and how an increase/decease in such parameters can affect the system. Unfortunately due to timing constraints, the team decided that PID tuning was not as essential as our other goals. PID tuning was decided to be left for the next semester team members to tackle. Fortunately, the knowledge I gained about what PID is very beneficial to future project members.

The next issue that I tackled collectively with my team members was bugs within Nav2. The bot seemed to display navigation issues whenever it got to close to objects. At first this issue seemed to have no fix. With extensive research on how Nav2 functions, I was able to come up with a solution on how to fix this. Nav2 is very customizable. There are a variety of costmap layers as well as filters. There are a total of eight layers within a single costmap: voxel, range, static, inflation, obstacle, spatio-temporal voxel, non-persistent voxel, and denoise layer[4]. There are a total of three filters: keepout, speed, and binary[4]. Once seeing that the costmap has lots of settings to alter and change as desired, the next part of the process was simpler. The costmap filter descriptions did not give an immediate solution to the issue. I proceeded to read what each costmap layer consists of. The first two layers that looked the most promising to solve the issue were the obstacle layer and the inflation layer[4]. The obstacle layer has a parameter can be defines the range being scanned and can be changed[4][5]. I thought that if the range was smaller than the default value, then the bot wouldn't be able to detect objects unless at a certain proximity. This could temporarily solve the issue of the robot not being able to get close enough to certain objects in the room. Although this seemed to provide a temporary solution, I did consider this to not be the best engineering solution. I think changing the range to a smaller value allows for other bugs to enter the system. The more scanning range the bot has the better costmap will be displayed. Therefore, the next solution I had in mind was to change the parameters in the inflation layer. The inflation layer adds a safety buffer to objects, essentially making objects appear larger than they already are[4][1]. If there were two objects close to each other, then the inflation radiuses of the objects were either touching or just really close to each other. This in turn causes there to be a narrow path for the bot to go through. This further established the notion that the bot was having issues going through these objects because the path between the objects was too narrow. The default inflation radius in Nav2 was set to .55[1]. I was unsure of the units of measurement Nav2 was using for this value. I did do research to see if there was any information on this topic in their documentation website but found nothing. I suggested we just reduce the inflation radius in general in hopes that the issue we were facing would subside. Fortunately, this was all there was to it. Decreasing the inflation radius allowed the bot to get from point A to point B. However, it should be noted that the bot would still run into the same issue wherever two objects are to close to one another. Further research continued on what else Nav2 has to offer.

The main focus I had this semester was the research aspect on finding a full coverage algorithm to have the CleanBot3000 reach full autonomy. An idea I had was to use planners that are already a part of Nav2. There are a total of five planners: NavFn Planner, SmacPlannerHybrid, SmacPlanner2D, SmacPlannerLattice, and ThetaStarPlanner[4]. Each planner does focus on point to point mapping through different algorithms. For instance, the NavFnPlanner uses an A* algorithm to navigate. The SmacPlannerLattice uses a state lattice for its navigation. Fig. 2 displays a sample of how the SmacLatticePlanner works[4].

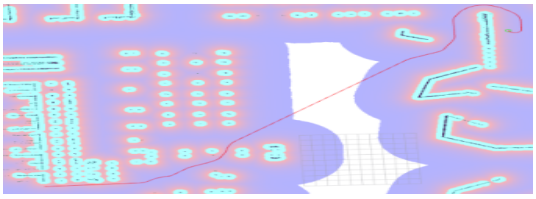

Fig. 2: Sample SmacLatticePlanner [8]

Most of these planners are plugins. I researched an learned how to create a planner plugin. The team came to the consensus to feed Nav2 different way points, one after another, that would allow for the full coverage/navigation of a room. This is a great step towards reaching autonomy, yet I still kept researching for better full coverage algorithms.

Another team member was able to find a great alternative to provide full coverage for a room: Open Navigation's Complete Coverage Algorithm[3][2]. Although I did not personally find this algorithm, I took it upon myself to study how this

algorithm works, what libraries it uses, and how to implement it into our current ROS2 framework. I took a look into what libraries this algorithm uses, specifically focusing on what Fields2Cover is and how it functions within the algorithm[3][2]. Me, as well as other team members, were able to set the foundation to for future team members to reach full autonomy. Fortunately, this coverage algorithm is compatible with both ROS2 and Nav2; it is an algorithm that is still currently being worked on by the developers of Nav2. Because this is still in development, there are some setbacks. I was able to research and read multiple forums that showed that there were still some bugs present when trying to implement this algorithm into the ROS2 framework. [3][2][6][10]

Although this algorithm is still in the works, I wanted to make sure that this algorithm was a suitable option and would give us the tools needed for full coverage via autonomy. I was able to learn about the framework of the Complete Coverage algorithm as well as an important library it implements: Fields2Cover[3][2][6][10]. There is a possibility to install just Fields2Cover into the system. I was able to find out the installation process as well as all its capabilities. This library works on the basis of headlands and swaths. It is mainly used within agriculture; however, it can definitely be applied in this project. There are many tutorials on how get this up and running on the documentation website[9]. Fig. 3 showcases an example of a generated field using Fields2Cover.

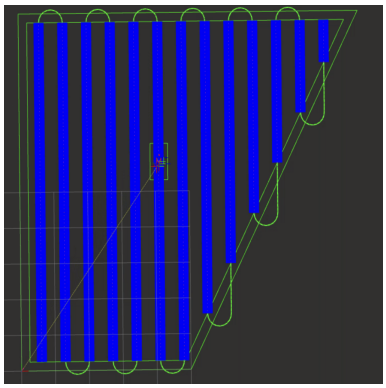

Fig. 3: Example Fields2Cover Field with Generated Swaths from Steven Macenski[2][6]

III. Overall Impact

Overall, my individual contribution in pair with my team members contributions were able to push the CleanBot3000 project in a forward direction. The main goal for this Spring semester 2024 was to have the bot get steps closer to being fully autonomous. We were able to achieve significant improvements towards autonomy. When it comes to evaluating our engineering solution, we took lots of factors into consideration. Our engineering solution does have a global impact, it expands more than just the local level. Because our robot tackles sanitization within cleanrooms at JPL to reduce microorganism exposure to extraterrestrial, our project helps protect other ecosystems outside of the scope of Earth. Not only can our design be used within JPL, but it can also be used within other organizations. It not only can be applied within cleanrooms but can also be applied at hospitals or other areas that require a high standard of cleanliness. Our team did also consider environmental concerns. In general, it is easy to produce excess amounts of waste. Our team made sure only use materials that were necessary. This in turn allowed us to not emit large amounts of waste as well as keep our project low budget. The project does cause implication to public health in the sense as it reduces the expansion of microorganisms being transferred to extraterrestrial systems. Because of this, the team took lots of time and care to produce a bot above expected standards. The team set lots of goals and tasks for each member to make sure that all areas of the project were covered. It was important to constantly remind ourselves why our project is essential to the engineering industry.

With the current engineering design of the CleanBot3000, it has the potential to significantly impact the globe as well as the environment. This project can be applied to so many other industries, not just be used in the cleanrooms at JPL. Our project can also be customized to fit the needs of others. Therefore, our engineering solution is very versatile. The team considered effects this project had overall. The scope of this project goes beyond just having a robot clean floors through autonomous navigation. It helps protect all ecosystems in space by reducing the amount of microorganism exposure to JPL flight hardware.

Overall, working as part of a team has many responsibilities. To make sure this team was successful, I created a collaborative and inclusive team environment by teaming up with others whenever I did not understand how to solve an issue. I also helped other team members trouble shoot issues they were having. I also made sure to always keep busy throughout the semester and constantly asked my team leader what other tasks I could do. Any free time that I had, I made sure to continue research on topics in relation to the overall team goals. I also took initiative to suggest new tasks/goals for the team to better improve our system. I also helped the team meet objectives by being present to every meeting, even extra meetings throughout the week. All the extra hours contributed to this project allowed for lots of improvements and achievements.

REFERENCES

[1] *Inflation Layer Parameters*. Available at https://docs.nav2.org/configuration/packages/costmap-plugins/inflation.html.

[2] Steve Macenski et al. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. URL: https://github.com/ros-planning/navigation2.

[3] Gonzalo Mier, João Valente, and Sytze de Bruin. "Fields2Cover: An Open-Source Coverage Path Planning Library for Unmanned Agricultural Vehicles". In: *IEEE Robotics and Automation Letters* 8.4 (2023), pp. 2166–2172. DOI: 10.1109/LRA.2023.3248439. URL: https://ieeexplore.ieee.org/document/10050562.

[4] *Navigation Plugins*. Available at https://docs.nav2.org/plugins/index.html.

[5] *Obstacle Layer Parameters*. Available at https://docs.nav2.org/configuration/packages/costmap-plugins/obstacle.html.

[6] *open-navigation/opennav$_c overage$*. Available at https://github.com/open-navigation/opennav_coverage?tab=readme-ov-file (2024/04/15).

[7] *OV7670 Camera Module: Datasheet, Specifications and Comparison*. Available at https://www.utmel.com/components/ov7670-camera-module-datasheet-specifications-and-comparison?id=797.

[8] *Smac Planner*. Available at https://docs.nav2.org/configuration/packages/configuring-smac-planner.html.

[9] *Tutorials*. Available at https://fields2cover.github.io/source/tutorials.html#.

[10] *Tutorials — Fields2Cover latest documentation*. Available at https://fields2cover.github.io/source/tutorials.html (2024/04/15).