Autonomous Occupancy Grid SLAM using Hector Quadrotor

Anthony Ambrose
Rochester Institute of Technology
Kate Gleason College of Engineering
Rochester NY, USA
axa6841@rit.edu

Nikhil Deshmukh
Rochester Institute of Technology
Kate Gleason College of Engineering
Rochester NY, USA
nxd4573@rit.edu

***Abstract*—As technology has evolved, there is a significant movement towards autonomous systems that are both more efficient and more versatile than human labor. A problem that one encounters when trying to design a mobile autonomous robot is: how does the robot know the environment that they are in? How does the robot navigate the environment when the environment is not a known factor or when there are obstacles, stochastic or deterministic, in the surroundings of the robot? Simultaneous Localization and Mapping is the solution to mobile autonomous robots. However, there are several types of SLAM algorithms with varying levels of accuracy for various environments. This project aims to explore a specific type of SLAM, Occupancy Grid SLAM, with a quadcopter.**

***Index Terms*—Hector, Quadrotor, autonomous, drone, SLAM, Occupancy Grid, Octomap**

I. Problem Statement and Goals

This project utilized ROS packages such as Hector and Octomap in order to create an occupancy grid based SLAM approach on a quadcopter. The system was simulated in gazebo and RViz using the ROS Kinetic environment in a virtual machine. Another goal of this project was to have the quadcopter be able to map the Z-direction in addition to the X-Y directions. The robot would first map the entire current plane and then elevate to a new Z level and remap. A Kinect is attached to the quadcopter and is the primary means of data input to the algorithm. The drone also uses information from the attached sonar sensor to control Z-axis mapping.

II. Literature Review

The topic of drones or unmanned aerial vehicles (UAV) is heavily researched. Additionally, SLAM is a constantly evolving methodology, with various implementations and necessary improvements. To begin, in [2], Alex Chandykunju and Aditya Vijaychandra proposed the idea of offloading tasks to multiple UAVs to execute during times of natural disasters. Specialized drones would allow for optimal efficiency with weight, battery life, and flight time. One drone would handle mapping the environment using laser scanners, stereo cameras, and IMU. SLAM is used to create the map which is then fed to another drone. The 3-D map would be sent to a cloud server which would have to be independently maintained to avoid downtime during natural disasters. The created map will then be used by another set of drones outfitted with infrared scanners/ thermal cameras to detected stranded people and then log their GPS coordinates. These drones would communicated with each other using RF transmitters [2].

There are various types of SLAM algorithms, however one of the most well known is the open source ORB-SLAM2 from [3]. It is a complete SLAM system for monocular, stereo, and RGB-D cameras. It also includes features such as map re-utilization, loop closing, and relocalization capabilities. One important aspect of visual SLAM is Place Recognition to close loops. The system detects when a sensor returns to a mapped area and then corrects the accumulated error in exploration, or relocalizes the camera after a tracking failure. To achieve greater accuracy, ORB-SLAM2 utilizes bundle adjustment (BA) instead of traditional methods such as interactive closest point (ICP) or depth error minimization [3]. One of it's main strengths is that it features a lightweight localization mode that can reuse the map with mapping disabled, and achieve zero-drift localization in already mapped areas [3].

Another project uses convolutional neural networks (CNN) for depth prediction in combination with monocular SLAM [4]. This solves two primary issues with monocular slam; low map completeness, and scale ambiguity. One of the downsides to using CNNs is that they can sometimes give inaccurate depth predictions which lead to large tracking errors in SLAM. CNNs are classified as weakly supervised depth prediction networks. They are designed to learn the correlations between visual patterns/cues and absolute depths from a large amount of training pairs. Tracking poses from the direct monocular SLAM are applied to pairs of motion stereo images to tun the CNN model in real-time, improving accuracy and generalization ability progressively for various environments [4].

Visual Planar Semantic (VPS) SLAM is another observed technique where visual odometry/ visual inertial odometry are used to map planes in the environment [5]. VIO is considered a special instance of SLAM as the camera and IMU data are used to estimate the robot location and map the environment [8]. VIO does not have loop closure, and as a result the system can output the estimated motion of the robot at a significantly higher throughput compared to traditional SLAM [8]. Most semi-dense SLAM techniques rely on low level characteristic features of the environments such as points lines and planes. Illumination changes, and repetitive

patterns affect the performance as a result. VPS is a dense 3-D mapping SLAM technique which does not pose these same shortcomings. It uses object detection to aid in planar mapping. It uses integral normal estimation to perform real time extraction of normals at each 3-D point to aid in planar mapping [5].

Air SLAM from [1], uses a stereo camera configuration in contrast to the traditional monocular SLAM. The stereo configuration is used to mitigate issues with scale definition and 3-D information. First, keypoints are extracted from images. These keypoints are described by rotary-binary robust-independent elementary features (rBRIEF). Keypoint descriptors are used to determine the environment map for the drone. To estimate robot's position, the keypoints from each camera view are matched to determine the transformation matrix between two different timed views [1]. It was learned that many ground based drone control systems choose parallel tracking and mapping (PTAM) over visual SLAM due to complexity and response time.

Another project discusses the use of SLAM in conjunction with AR. SLAM is used to map the environment and create markers for the AR to easily recognize objects and overlay a virtual augmentation [6]. Positional instant tracking is used as the relative position is analyzed along with scene features to determine the robot's alignment. It is mportant to note, SLAM is not used for marker based tracking. After the mapping is done, the models are created on the now known environmental ground plane [6].

Another project operates under the idea that SLAM consists of two parts; a FRONT-END aiming to detect, identify, and track different features, and a BACK-END, which uses detected feature projection on the camera plane and then calculates camera position, orientation, and feature positions in the real world [7]. Due to SLAM being traditionally based on relative information, there tends to be accuracy drift (decrease) over time. The goal was to use global landmarks to navigate in global coordinates instead of relative coordinates, and ultimately reduce drift [7]. However, a large database of global landmarks is needed to refer to, which also increases processing time [7].

Another project created a low cost, lightweight SLAM solution using only a monocular camera and IMU [9]. Through trace correlation analysis between the camera motion and object motion, a 3-D tracking system is developed, and neither scale nor motion assumption of the dynamic object are needed prior [9].

Major challenges of visual tracking on UAVs include target representation, target appearance change, target detection and localization in real time computation as mentioned in [10]. A hybrid solution was developed where additional data are added to scenes in conjunction with object tracking. This data includes describing places, natural features or general points of interests. Additional features in the scene will allow the drone to better identify and label tracked objects [10].

One obstacle to SLAM robots is the complications that occur when the environment is dynamic and the sensor measurements are noisy. In order to address these issues, methods such as the "extended Kalman Filter SLAM, Fast SLAM, Graph based SLAM, and Occupancy grid SLAM" have been designed [18].

The most common type of occupancy grid maps are 2D slices of a 3D environment. However, a 3D occupancy map is more effective than its 2D counterpart. The heavy computational load of 3D occupancy grids has resulted in it being less popular than 2D grid maps. A paper [12] submitted to the 2020 12th International Conference on Intelligent Human Machine Systems and Cybernetics proposed utilizing the 2D range image to accelerate the processing of a 3D occupancy grid map, and using stixels as the processing primitive. This is in contrast to Octomap using voxels and octrees as the hierarchical data structure. Experiments showed that their proposed approach can run faster than Octomap as well as eliminate the influence of dynamic objects, which is something that can be implemented in future work.

Once SLAM is fully realized on mobile robots, the next step will be fully autonomous systems or collaborative robots. Before that point, there must be some man-machine interface in place to allow for communication between autonomous robots and people. Engineers from Zurich University of Applied Sciences submitted a paper earlier in 2020 [13] attempting to create that man-machine interface, specifically in regard to drones. Their work focused on human-drone interfacing using marshalling signs and low cost drones, with minimal addition of extra sensors/cameras. They managed to create an interface that can interpret simple Yes/No commands as well as a command to gain the drones attention.

Jessica Cauchard et al. [15] attempted to tackle the same problem as the engineers from ZHAW, the man-machine interface, specifically in regard to drones. They introduced drone.io, "a projected body-centric graphical user interface for human-drone interaction". Through use of onboard projectors on the drone, they were able to create a fully functional prototype. Drone.io has both input and output capabilities with a drone. This could be implemented with Hector and our SLAM algorithm in order to create a responsive drone that can map the environment then proceed to interact with the user within the environment, which is a good idea for future work.

Previously mentioned graph based SLAM makes a return appearance in a paper submitted to ICCAS 2018 [19], in which the authors seek to combine RGB-D cameras and magnetic SLAM in order to create a more robust SLAM algorithm. In their paper, they cite better performance in feature-poor environments such as hallways. In general, feature-poor environments or noisy sensor measurements is the main issue with SLAM algorithms. There is almost always a use case in which the algorithm struggles to keep its head above water, so to speak. Papers like the aforementioned combine different SLAM methods as well as different sensors in order to get a holistic view of the environment. Moving forward, we will definitely try to design our robots to be as robust as possible by incorporating sensor fusion.

In order to explore remote areas that are normally inaccessible, drones are outfitted with cameras and sent into the environment. However, the limitation of a camera is that only 2D images or videos are supplied. In order to provide a more complete spatial environment to the user, a 2019 paper [11] from students at the University of Toronto proposed combining haptic force feedback with camera vision. The paper seeks to establish that through the use of haptic rendering and Generative Adversarial Networks, a user can observe and interact with their environment through the use of a drone. This is once again a project that tries to create a man-machine interface which is something that could be turned into a stretch goal for our project.

One industry trying to take advantage of autonomous systems is the parcel delivery system, such as Amazon. However, the issue drones have with parcel delivery is payload weight limitation and battery capacity. Seeking to overcome these challenges, a paper [14] submitted to the 2019 IEEE IoT journal proposes a three-pronged approach of trucks, truck-carried drones, and independent drones. Their experimental results strongly suggest that their hybrid parcel delivery system would outperform the current system as well as previously proposed systems of independent drones or truck carried drones and trucks. The relationship between this paper and our project is that swarm capability is a stretch goal for us, and a drone swarm with SLAM capabilities would be a requirement in any delivery system.

In relation to the swarm stretch goal, a paper published by Yosuke Kishimoto, Kiyotsugu Takaba, and Asuka Ohashi [17] details a multi-robot SLAM based on the C/GMRES method. They developed a distributed multi-robot SLAM method in which robots share their past and present sensor measurements with adjacent robots. This is represented as a moving horizon estimation, or MHE, problem. The MHE problem is then solved using the SLAM algorithm in conjunction with the continuation/generalized minimum residual, C/GMRES, method. The C/GMRES method has its roots in real-time optimizations in nonlinear predictive control systems. [b21]. This paper illustrated how SLAM algorithms can incorporate complex functions through use of mathematical representations and how computational efficiency plays a large role in mobile robot SLAM algorithms. This is a large takeaway from the project, inefficiencies in SLAM implementation is unacceptable since the margins are so thin with technology where it currently is.

A paper by Qiannan Cui, Peizhi Liu, Jinhua Wang, and Jing Yu [20] provides insight into drone swarm communication methods. At a base level, a drone swarm is a wireless mesh network. The challenge is keeping the many-to-many communication decentralized and limited to simple interactions. This reduces the error in the network as well as increases the speed of information flow within the network. For our stretch goal of implementing swarm capability, this paper is extremely helpful in designing a communication network.

III. Methodology and Implementation

A. ROS Packages

There were a number of different ROS packages used. An itemized list is provided below.

- Hector Quadrotor [22]
- Octomap [23]
- Move Base (part of ROS navigation stack) [24]
- TEB Local Planner [25]
- ROS Kinetic on Ubuntu 16.04
- Python 2.7: numpy, rospy, math, etc.

Hector Quadrotor is an extremely helpful ROS package containing the URDF for Hector, along with several tutorials and a SLAM algorithm implemented with GMapping. One of the Hector models came with a Microsoft Kinect The package also contained some prebuilt world maps but we found that they did not function that well and routinely crashed, so we made our own using Gazebo Building Editor. We did not use the prebuilt Hector SLAM functions either.

Octomap is a ROS library that implements a 3D occupancy grid mapping approach using octrees as hierarchical data structures. As mentioned in Section II, 3D occupancy grid mapping is more efficient but more computationally expensive than normal 2D occupancy grid mapping. However, Octomap's use of octrees is both accurate and efficient in terms of SLAM methods and provides more than enough capability for our project.

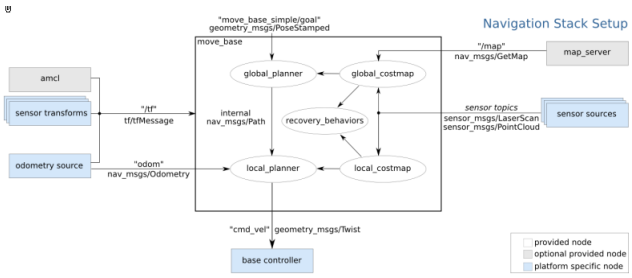

Fig. 1. ROS move base detailed description

The move base ROS package is a part of the overall ROS navigation stack repository. This package links together a global planner and local planner and completes navigation tasks. It is modeled by the above diagram, Figure 1, which was provided in the ROS documentation. The global planner used is a global planner included in the navigation stack, "navfn". The local planner was the TEB Local Planner [25], which is a plugin to the base local planner in the navigation stack. The global and local planners were selected after reviewing the data that other projects had as to select an accurate and efficient planner.

One of the custom worlds we built is shown in Figure 2. There were two that we designed, one to test the X-Y mapping capabilities and one to test the Z-axis mapping capabilities. The one shown in Figure 2 was designed to test the Z-axis, which is why it is tower-like.

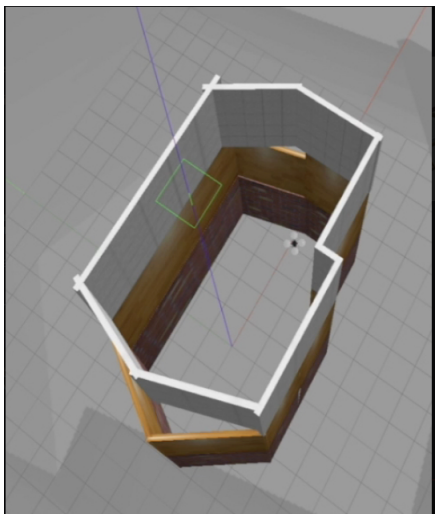

Fig. 2. Vertical World to test Z-axis mapping

B. Implementation

The overall code flow is displayed in Figure 3. The algorithm starts off by creating all the necessary drone objects since our code was structured in a C++ object oriented style. Then, the program waits for the costmap evaluation data and the frame transform data, which comes from the navigation stack. The program then proceeds to takeoff and begin exploring the level.

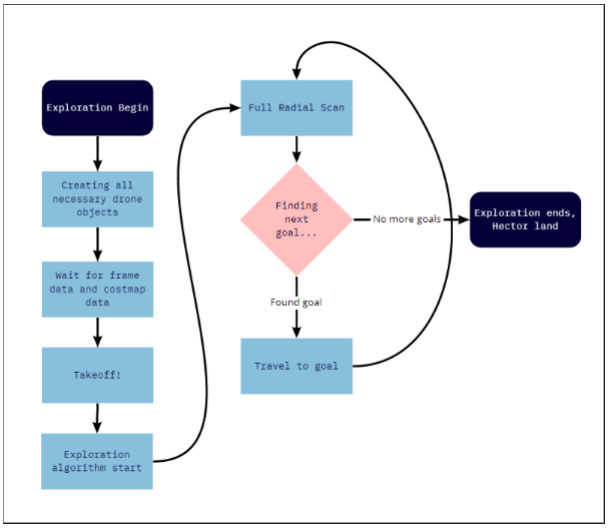

Fig. 3. Overall Code Flow (Larger image in Appendix)

Once exploration has begun, the program loops between a Full Radial Scan, searching for goals, and traveling to goals. If there are no more new goals to be found, the exploration ends and Hector lands. If a goal remains, Hector will proceed to the goal and complete a full radial scan, adding whatever new information found to the costmap. The goal is chosen by using a find_goal function designed by Timur Kuzhagaliyev and Qingzhuo Aw Young [26].

C. Validation

In order to validate the results of the project, a binary pass/fail observation of RViz is enough. Fortunately, Octomap displays extremely nicely on RViz, so the occupancy grid mapping is clear to the user. If the occupancy grid does not appear as the quadcopter is moving around the world, then it does not work.

Some data that was being observed during the operation of the drone was the sonar sensor topic and the RGB-D camera visualization. Hector also came with a Kinect attached to it and the pointcloud data was visualized on RViz.

IV. Results

The virtual machine was modified to allow full graphics acceleration. This allowed the simulation to run much smoothly at around 30 fps and also reduced run time. Using the custom written scripts, the program was executed in the custom environment depicted in Figure 4. The final result of the completed SLAM algorithm is illustrated in Figure 5.

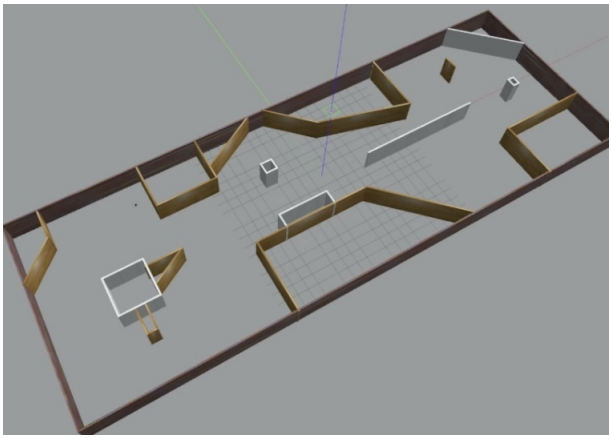

Fig. 4. Single Plane Environment

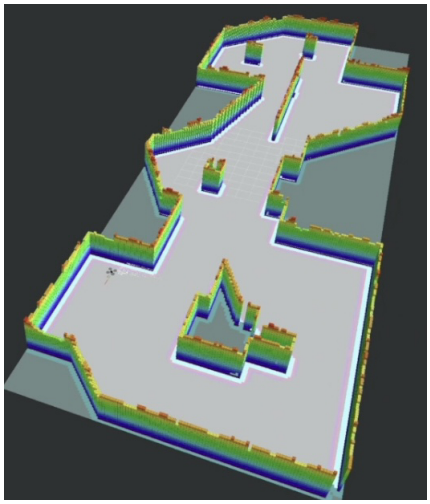

Fig. 5. Mapped Single Plane Environment

Figure 4 illustrates the custom gazebo world while Figure 5 represents the resulting map in RViz. The simulation took approximately one hour and 32 seconds to complete. The long run time is due to how the planning function works. The planner explores the world outwards from the origin on a horizontal. Visualizing a standard x-y plane, the drone will move from the origin and outwards to positive and negative x values equidistantly. It will not move to a new y value unless the entire current y value has been explored. Due to this inefficiency, the drone travels throughout many of the explored areas repetitively when trying to explore new regions.

A modified version of the exploration script was made to test the drone mapping different elevations. The custom world was altered and the one depicted prior in Figure 2 was used. The resulting RViz map is shown below in Figure 6.

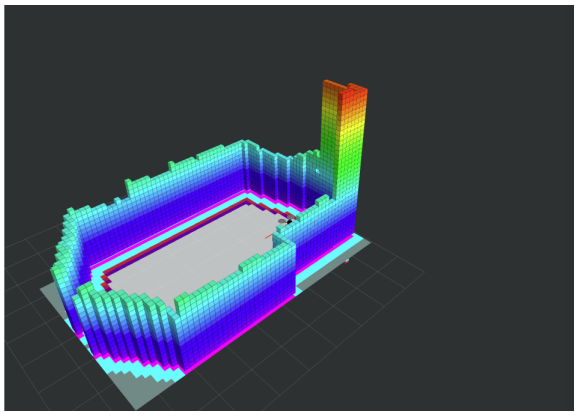

Fig. 6. Mapped Elevated Environment

The drone was able to start mapping the environment at higher elevations, however an issue arose. The sonar sensor that is pre-packaged with Hector begins to fail above two meters. The max sonar range value was even adjusted in the URDF to 10 meters, however the same issue arises. The problematic data can be seen in the snippet shown below, Figure 7, which displays the ROS echo of the sonar plugin.

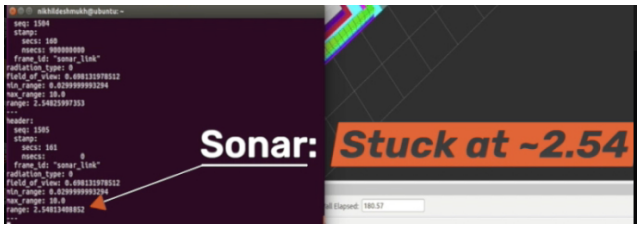

Fig. 7. ROS Sonar Output

The current implementation elevates the drone 2 meters from the current position, meaning it moves from Z=1 to Z=3. Since the sonar never reaches 3 meters, the drone continues to rise resulting in the map above. The logic was written to store the goals from the previous elevation and iterate through them to map the new elevation. However, due to the sonar limitations, this could not be properly visualized.

V. Conclusion

The extensive ROS documentation on packages like Octomap and Navigation Stack proved invaluable in completing the project. There were also several open source tutorials for running Hector on ROS Indigo, however this is a pretty outdated version. An updated version was later found for ROS Kinetic which was found to run better. However, minimal documentation for running Hector on ROS Kinetic was found, so setup time was increased. Unfortunately, there were no other updated packages found to run Hector on later versions of ROS.

A future goal for us is to give this project swarm capability and perhaps some other ROS integrations for more robust SLAM and more functionality. As it currently stands this is a bare-bones occupancy grid SLAM using Hector and swarm capability would be an extremely interesting project.

References

[1] P. Araújo, R. Miranda, D. Carmo, R. Alves and L. Oliveira, "Air-SSLAM: A Visual Stereo Indoor SLAM for Aerial Quadrotors," in IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 9, pp. 1643-1647, Sept. 2017, doi: 10.1109/LGRS.2017.2730883.

[2] C. Alex and A. Vijaychandra, "Autonomous cloud based drone system for disaster response and mitigation," 2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA), Kollam, 2016, pp. 1-4, doi: 10.1109/RAHA.2016.7931889.

[3] F. Zeng, W. Zeng and Y. Gan, "ORB-SLAM2 with 6DOF Motion," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, 2018, pp. 556-559, doi: 10.1109/ICIVC.2018.8492909.

[4] H. Luo, Y. Gao, Y. Wu, C. Liao, X. Yang and K. Cheng, "Real-Time Dense Monocular SLAM With Online Adapted Depth Prediction Network," in IEEE Transactions on Multimedia, vol. 21, no. 2, pp. 470-483, Feb. 2019, doi: 10.1109/TMM.2018.2859034.

[5] H. Bavle, P. De La Puente, J. P. How and P. Campoy, "VPS-SLAM: Visual Planar Semantic SLAM for Aerial Robotic Systems," in IEEE Access, vol. 8, pp. 60704-60718, 2020, doi: 10.1109/ACCESS.2020.2983121.

[6] S. Sreeram, K. K. Nisha and R. Jayakrishnan, "Virtual Design Review and Planning Using Augmented Reality and Drones," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 915-918, doi: 10.1109/ICCONS.2018.8662919.

[7] R. Ronen, A. Jigalin and Z. Berman, "Development Challenges and Performance Analysis of Drone Visual/Inertial SLAM in a Global Reference System," 2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), Portland, OR, USA, 2020, pp. 125-136, doi: 10.1109/PLANS46316.2020.9110128.

[8] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman and V. Sze, "Navion: A 2-mW Fully Integrated Real-Time Visual-Inertial Odometry Accelerator for Autonomous Navigation of Nano Drones," in IEEE Journal of Solid-State Circuits, vol. 54, no. 4, pp. 1106-1119, April 2019, doi: 10.1109/JSSC.2018.2886342.

[9] K. Qiu, T. Qin, W. Gao and S. Shen, "Tracking 3-D Motion of Dynamic Objects Using Monocular Visual-Inertial Sensing," in IEEE Transactions on Robotics, vol. 35, no. 4, pp. 799-816, Aug. 2019, doi: 10.1109/TRO.2019.2909085.

[10] D. Cavaliere, V. Loia, A. Saggese, S. Senatore and M. Vento, "Semantically Enhanced UAVs to Increase the Aerial Scene Understanding," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 3, pp. 555-567, March 2019, doi: 10.1109/TSMC.2017.2757462.

[11] T. Duan, P. Punpongsanon, S. Jia, D. Iwai, K. Sato and K. N. Plataniotis, "Remote Environment Exploration with Drone Agent and Haptic Force Feedback," 2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), San Diego, CA, USA, 2019, pp. 167-1673, doi: 10.1109/AIVR46125.2019.00034.

[12] H. Fu, H. Xue and R. Ren, "Fast Implementation of 3D Occupancy Grid for Autonomous Driving," 2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 2020, pp. 217-220, doi: 10.1109/IHMSC49165.2020.10127.

[13] H. D. Doran, M. Reif, M. Oehler, C. Stöhr and P. Capone, "Conceptual Design of Human-Drone Communication in Collaborative Environments," 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Valencia, Spain, 2020, pp. 118-121, doi: 10.1109/DSN-W50199.2020.00030.

[14] D. Wang, P. Hu, J. Du, P. Zhou, T. Deng and M. Hu, "Routing and Scheduling for Hybrid Truck-Drone Collaborative Parcel Delivery With Independent and Truck-Carried Drones," in IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10483-10495, Dec. 2019, doi: 10.1109/JIOT.2019.2939397.

[15] J. R. Cauchard et al., "Drone.io: A Gestural and Visual Interface for Human-Drone Interaction," 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Daegu, Korea (South), 2019, pp. 153-162, doi: 10.1109/HRI.2019.8673011.

[16] X. Chen, H. Zhang, H. Lu, J. Xiao, Q. Qiu and Y. Li, "Robust SLAM system based on monocular vision and LiDAR for robotic urban search and rescue," 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, 2017, pp. 41-47, doi: 10.1109/SSRR.2017.8088138.

[17] Y. Kishimoto, K. Takaba and A. Ohashi, "Moving Horizon Multi-Robot SLAM Based on C/GMRES Method," 2019 International Conference on Advanced Mechatronic Systems (ICAMechS), Kusatsu, Shiga, Japan, 2019, pp. 22-27, doi: 10.1109/ICAMechS.2019.8861681.

[18] P. Sasithong et al., "4G Enabled PhoneBot with Cloud based SLAM," 2020 International Conference on Electronics, Information, and Communication (ICEIC), Barcelona, Spain, 2020, pp. 1-4, doi: 10.1109/ICEIC49074.2020.9051151.

[19] H. Kim, S. Song, J. Hyun, S. H. Hong and H. Myung, "RGB-D and Magnetic Sequence-based Graph SLAM with Kidnap Recovery," 2018 18th International Conference on Control, Automation and Systems (ICCAS), Daegwallyeong, 2018, pp. 1440-1443.

[20] Q. Cui, P. Liu, J. Wang and J. Yu, "Brief analysis of drone swarms communication," 2017 IEEE International Conference on Unmanned Systems (ICUS), Beijing, 2017, pp. 463-466, doi: 10.1109/ICUS.2017.8278390.

[21] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control,"Automatica, vol. 40, pp. 564-574, 2004.

[22] Meyer, Johannes Sendobry, Alexander Kohlbrecher, Stefan Klingauf, Uwe Von Stryk, Oskar. (2012). Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo. 7628. 400-411. 10.1007/978-3-642-34327-8_36.

[23] Hornung, A., Wurm, K.M., Bennewitz, M. et al. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. Auton Robot 34, 189–206 (2013). https://doi.org/10.1007/s10514-012-9321-0

[24] Marder-Eppstein, Eitan. "Ros-Planning/Navigation." GitHub, github.com/ros-planning/navigation.

[25] C. Rösmann, F. Hoffmann and T. Bertram: Integrated online trajectory planning and optimization in distinctive topologies, Robotics and Autonomous Systems, Vol. 88, 2017, pp. 142–153.

[26] Young, Qingzhuo Aw, and Timur Kuzhagaliyev. "CS/ME/EE 134 Project Report." GitHub, 15 June 2018

APPENDIX

A. *Overall Code Flow (Zoomed In)*