UAV Autonomous Navigation and Target Identification in ROS Simulation

Michael F. Smith III
Electrical Engineering Undergrad,
Rochester Institute of Technology
Rochester, NY, USA
mfs1036@rit.edu

Abstract—**Unmanned aerial vehicles or UAV's have many applications and uses across a varied of industries. For a while they were primarily used for military applications. According to Britannica, an unmanned aerial vehicle is a "military aircraft that is guided autonomously, by remote control, or both and that carries sensors, target designators, offensive ordnance, or electronic transmitters designed to interfere with or destroy enemy targets" [Britannica]. Over the past few decades, UAV's have began to have a wider variety of uses and applications. Today, UAV's and drones are used in anything from photography, to surveying, to search and rescue, to drone racing. For many of these applications some form of human interaction is required in order to pilot the drone. Despite this, there are many possible applications that could benefit from autonomous navigation of UAV's. A large field of UAV flight is that of drone racing. In FPV (first-person view) drone racing, a pilot is required in order to navigate the UAV around a 3D course which is usually layed out with cones, walls, flags or hoops. The focus of this paper details the process of utilizing the Robotic Operating System or ROS to simulate and emulate the autonomous navigation of a UAV's and the design of an algorithm to detect objects or targets in their environment through the use of cameras. Once the object or target is detected a path planning algorithm will determine the route in which the UAV should take to navigate around or through the found target. The goal is to develop an algorithm which permits a UAV to autonomously navigate a drone race course through the identification of course obstacles such as hoops or flags.**

Index Terms—**Unmanned Aerial Vehicles, Autonomous Flight, Path Planning, Drone Racing, Camera Recognition, RGB Color Recognition, ROS (Robotic Operating System), Gazebo**

I. Introduction

The unmanned aerial vehicle is a descendant of the target drone utilized in the years following World War II. Since the early 1980's, UAV's became a standard military weapons system after they were first implemented by the Israeli military [Britannica]. These uav's were mostly if not all guided and piloted by human interaction. In 2006 non-military companies began research and development in utilizing UAV's for commercial applications. Then in 2013 Amazon announced that it would begin utilizing drones for package deliveries. This announcement brought UAV's into the public eye for the first time and a more recreational hobbyist industry began. Since 2013, the industry has vastly grown and the applications of UAV's are much more numerous. Despite this growth, the field of autonomous flight and navigation is one that is often a challenge. With the widespread commercialization of UAV's, drone and quadcopter kits have become much more affordable for use in research and education. Despite this, there are still many "risks of damage to the flight system in the event of failures" as many of the various components that make up a UAV are designed to be light and without protective casings or housings to make them lighter or faster [Quadcopter UAV using ROS and Gazebo]. To overcome the chances of damaging multiple UAV's, A UAV or quadcopter can be imported into a simulation software such as Gazebo in ROS and analyzed and studied virtually. Taking into account the social distancing requirements as a result of the Covid-19 pandemic, this approach also permits demonstration of the UAV in a virtual environment remotely.

Additionally, through simulation a variety of obstacles, environments, and objects can be simulated and analyzed through the navigation of a UAV. To best analyze an object / target identification algorithm, the virtual environment in which the UAV would reside could be modified to analyze its approach to maneuvering around its environment and locating its targets. In addition to a easily modifiable environment, utilizing a simulation to analyze a UAV would permit the use of a variety of virtual sensors which would physically be quite expensive and unachievable. Many existing UAV's and their corresponding sensors have been implemented into ROS and Gazebo.

With the widespread interest in UAV's and autonomous flight, there have been a variety of methods to implement said autonomy. Many researchers in the field of autonomous UAV flight have explored utilizing Artificial Intelligence, especially Neural Networks in the training of drones and UAV's.

This paper proposes a method of implementing autonomous UAV flight as well as target identification in simulation through the use of the Robotic Operating System and Gazebo. The Robotic Operating System is "a set of software libraries and tools that help build robot applications" [ROS.org]. A pre-designed UAV as well as its ROS packages will be utilized to simulate a UAV in ROS. Examples of these include the hector_quadcopter as well as the DJI quadcopter or the AR Parrot drone. The included packages and sensors in these drones will be utilized to implement an autonomous navigation algorithm. After this autonomous navigation algorithm is implemented, a target identification algorithm will also be implmented so that the robot can identify targets or hoops that it could fly through. These hoops could be thought of as similar to those that drones fly through in Drone Racing.

The goal of this method is to design an autonomous navigation algorithm as well as an target identification algorithm so that a UAV can autonomously navigate through a virtual course of hoops based on their color (autonomously navigate through a drone racing course). Section II. of this paper will focus on the comparison of this method to research that has been done previously. Section III. will discuss the means of implementing the proposed method. Simulation softwares will be analyzed and ROS packages will be discussed. Section IV. details how the method will be validated and deemed successful.

II. Literature Survey

Much research has been conducted within the field of Unmanned Aerial Vehicles and autonomous navigation and flight. Many papers exist that implement autonomous flight, navigation, and landing of a UAV utilizing cameras or Time-of-Flight sensors such as LiDaR.

A paper titled the "Autonomous Navigation in Drone Racecourse" details a method of implementing autonomous navigation of a AR Parrot 2.0 drone as it flies around a drone race course. This paper details that this UAV contains "five sensors: a forward-looking camera, [a] downward-looking camera, [an] ultrasonic (US) rangefinder, [a] pressure sensor, and [an] inertial measurement unit (IMU)" [7]. The paper further outlines their method of implementing the autonomous navigation. Their depiction of this system architecture can be seen in Figure 1.

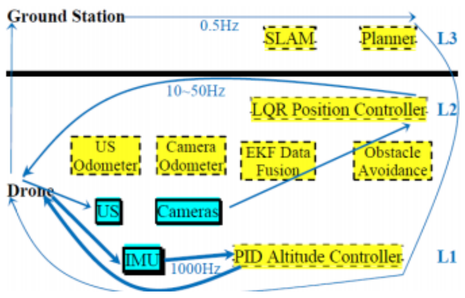

Fig. 1: Overview of System Architecture

In addition to a high level overview of the system architecture, the paper also delves into the mathematical models required to implement flight control. Through utilizing RPY (roll-pitch-yaw) angles and their corresponding rotation matrices as well as matrices to represent the lift forces generated

$$\textit{parallel motion:}\ \begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{1}{m}\begin{bmatrix}F_x\\ F_y\\ F_z - mg\end{bmatrix} = \begin{bmatrix}\frac{\sin\theta}{m}\sum_{i=1}^{4}F_i\\ \frac{-\sin\emptyset\cos\theta}{m}\sum_{i=1}^{4}F_i\\ \frac{\cos\emptyset\cos\theta}{m}\sum_{i=1}^{4}F_i - g\end{bmatrix}$$

Fig. 2: Parallel Motion of UAV

$$\textit{rotary motion:}\ \begin{cases}\ddot{\emptyset} = l(F_1 - F_2 - F_3 - F_4)/\, I_x\\ \ddot{\theta} = l(F_1 + F_2 - F_3 - F_4)/\, I_y\\ \ddot{\psi} = (F_1 - F_2 + F_3 - F_4)/\, I_z'\end{cases}$$

Fig. 3: Rotary Motion of UAV

by each propeller, parallel motion and rotary motion of the UAV are derived using Newton's Second Law. The resulting equations can be seen in Figures 2 and 3, respectively. Through analysis of this journal, the method to be implemented in this paper will take a similar approach to autonomous navigation. The journal also discusses the use of computer vision for implementation of the autonomous navigation. In order to utilize the camera for computer vision, the journal discusses utilizing the concept of a pinhole camera and determine the edges of objects through Hessian's gradient matrix. A Kalman filter was also utilized with the computer vision to combat noise in the images collected by the camera. The found journal does not delve into ROS, so the ROS implementation will be further explored and developed in this paper.

Another literature journal focusing on autonomous drone racing, was the one mentioned in the eighth literature paper titled "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning" [8]. This paper focuses on the implementation of computer vision in order for a UAV to accurately detect the gates that it has to fly through when participating in a drone race. The method described in this paper may be too intense for the hardware that this paper plans to utilize its own method. The journal details that all of their vision processes are "performed in real time on the onboard NVIDIA Jetson TX2 embedded computer" [8]. This NVIDIA computer is utilized to implement a convolutional neural network for detecting the center of a gate as well as its color to determine which hoop the UAV must fly through. The method that this paper proposes a neural network known as ADRNet which was utilized for the UAV guidance. In order to train their neural network, the paper discusses the use of their own data set for drone racing which includes a variety of images of drone racing gates in various environments. An image depicting the gates utilized in this paper can be seen in Figure 4. The paper also details that the guidance of the UAV was implemented using a Line-of-Sight (LOS) algorithm.

To properly implement the control laws required for autonomous flight, the ninth literature paper can assist. The journal written by Yu, Chan, Fan, Song, and Zhu describes the dynamic modeling. It describes that an aircraft is generally defined as the equation seen in Figure 5. The paper details the "small angle approximate model based on Eulerian - Lagrange

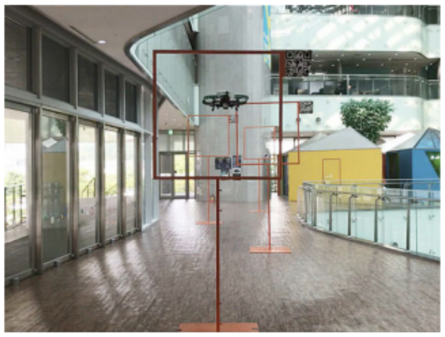

Fig. 4: Gates utilized for Training Neural Net in Literature 8

$$q = (\xi, \eta)^T = (x, y, z, \psi, \theta, \phi)^T \in R^6$$

Fig. 5: Definition of an Aircraft from Literature 9

equations of motion as seen in Figure 6 [9]. In order to implement these equations, the journal details that ROS was utilized to model and analyze Crazyflie with the proposed control laws which can be seen in Figures 7 and 8 [9].

Another paper that is relevant to the topic of object detection by a UAV is the journal known as ”A Deep-Learning-Based Sea Search and Rescue Algorithm by UAV Remote Sensing.” This literature paper proposes a method to locate where victims may be on a map using images acquired by a UAV. This implementation is similar to the method proposed by this paper, as both methods would implement the usage of cameras and algorithms that would analyze the images taken and then determine the targets based on the objects located in the images.

My implementation of the UAV implementation will be similar to a method discussed in the fifth literature article concerning the use of ROS and Gazebo to simulate a Quadrotor UAV. This article discusses the implementation of a UAV robot in ROS. It details the use of URDF’s to represent the

$$\tau_\phi = -\sigma_\phi \{\dot{\phi} + \sigma_{\phi_2}[\dot{\phi} + \phi + \sigma_{\phi_3}[\dot{\phi} + 2\phi + \frac{\dot{y}}{g} + \sigma_{\phi_4}(\dot{\phi} + 3\phi + 3\frac{\dot{y}}{g} + \frac{y}{g})]]\}$$

Fig. 7: Rolling Control Law defined in Literature 9

$$\tau_\theta = -\sigma_{\theta_1} \{\dot{\theta} + \sigma_{\theta_2}[\dot{\theta} + \theta + \sigma_{\theta_3}[\dot{\theta} + 2\theta - \frac{\dot{x}}{g} + \sigma_{\theta_4}(\dot{\theta} + 3\theta + 3\frac{\dot{x}}{g} + \frac{x}{g})]]\}$$

Fig. 8: Pitch Control Law defined in Literature 9

UAV within the software. If needed, the UAV detailed in this paper could be implemented using URDF’s, but by utilizing a pre-built robot such as hector_quadcopter or the AR Parrot UAV, these would be included within the ROS packages. An example of the environment model implemented in ROS in the fifth literature article can be seen in Figure 9. This literature journal also delves into the path planning algorithm that was implemented. The method to be implmented through this paper will probably take a similar approach to that seen in Figure 10.

The sixth literature article details research conducted in regards to autonomous obstacle avoidance and flight path planning. This journal details the basic design of a control system for the implementation of obstacle avoidance. The following graphic seen in Figure 11 depicts the schematic of the obstacle avoidance control system.

III. Method

A. Task Environment and Initial Methodology

The task environment chosen to implement autonomous navigation is the Gazebo environment built into ROS. Multiple literature papers discuss implementing this and this will permit analysis of the UAV without the need for expensive drones,

$$\begin{aligned}
m\ddot{x} &= -F_T \sin(\theta), \\
m\ddot{y} &= F_T \cos(\theta)\sin(\phi), \\
m\ddot{z} &= F_T \cos(\theta)\cos(\phi) - mg, \\
\ddot{\psi} &= \tau_\psi, \\
\ddot{\theta} &= \tau_\theta, \\
\ddot{\phi} &= \tau_\phi,
\end{aligned}$$

Fig. 6: Eulerian-Lagrange Equations of Motion from Literature 9

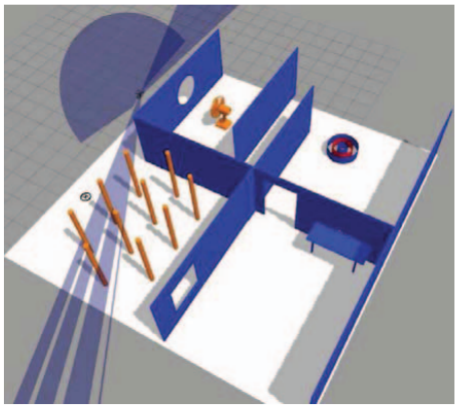

Fig. 9: Simulated ROS Environment from Fifth Literature Journal

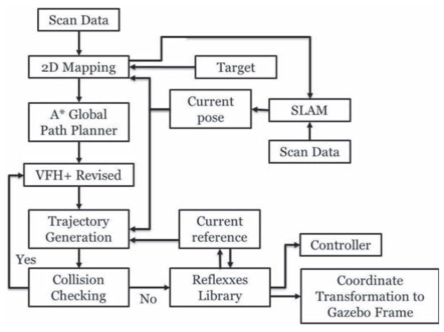

Fig. 10: Example Path Planning Algorithm from Fifth Literature Journal

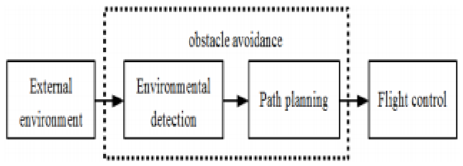

Fig. 11: Overview of Obstacle Avoidance from Sixth Literature Journal

sensors, or equipment. Additional softwares and simulation environments such as Unity and Coppellia may need to be explored in order to simulate the UAV with a proper physics engine and the effects of lift and gravity on the system.

The overall method that this paper will attempt is that of an obstacle avoidance to implement a simple autonomous navigation with target identification through the use of cameras. If time permits, a SLAM algorithm similar to that utilized by the Husky robot could be implemented.

The AR Parrot 2.0 Drone ROS packages will be utilized to model and simulate a UAV in ROS. This will be similarly implemented as described in literature paper 7. The AR Parrot 2.0 Drone will assist in testing as it has built-in sensors and cameras which will be utilized by the UAV to observe and respond to the virtual environment.

B. Methodology

Initially, the proposed method for implementing this project was to have the UAV identify hoops located in a ROS Gazebo environment and then have it move towards the identified hoop utilizing obstacle avoidance as the main method for implementing simple autonomous navigation. Instead of utilizing obstacle avoidance as the main method of autonomous navigation, a more sophisticated approach was taken.
The methodology taken involved utilizing the Parrot AR Drone 2.0 in Gazebo in ROS which can be seen in Figure 12. The ROS packages ardrone_autonomy and tum_simulator were utilized to implement the AR Drone within the ROS Kinetic distribution. Refernce 10 details a method to installing and setting up these packages within ROS Kinetic. No other simulation environments were required. Within this environment, a Drone Racecourse was designed and implemented utilizing multiple gray wall models as well as hoop models which were imported from a custom designed stl. A top-down view of the racecourse designed and implemented within Gazebo can be seen in Figure 13. Twelve hoops in total are utilized and are organized in a blue, green, red pattern in a clockwise orientation. The goal of the UAV will be to navigate around this course while passing through all of the hoops in the correct color order.

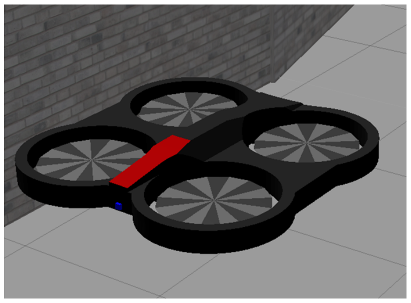

Fig. 12: The Parrot AR Drone 2.0 in Gazebo

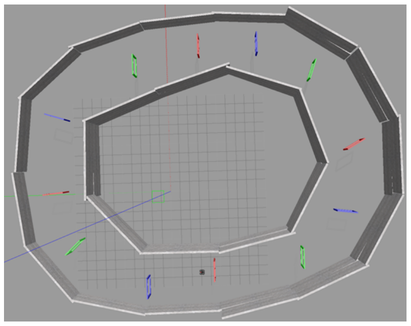

Fig. 13: Top-Down View of Racecourse implemented in Gazebo

The UAV was given primarily two algorithms to follow. These two algorithms include Autonomous Hoop Detection, and Autonomous Navigation. Both algorithms utilize the AR Drone 2.0's front and bottom facing cameras. The first algorithm Autonomous Hoop Detection is implemented utilizing OpenCV as well as the imutils *findContours* python package. Utilizing both of these packages, the UAV is able to read and interpret the images taken by the front and bottom built-in cameras. From the Opencv package, the images are analyzed for blue, green or red colors defined to match the same colors of the hoops located within the racecourse. An order to which the UAV should pass through the hoops is specified as blue, green and red, and then repeats in order to complete the racecoures. Once these colored hoops are located, the one matching the current hoop to find (i.e. blue initially), will be masked by through the opencv package. By masking the image, only the color of the hoop will be visible while the rest of the image will be blacked out. By

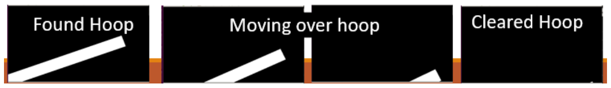

Fig. 14: Bottom Camera Hoop Presence Identification

doing this, the hoops area can be found utilzing the imutils findContours function which finds contours or edges within an image and applies a bounding box or area around the shape it finds. From this shape, the findContours function is able to determine the area of the shape as well as the corners of the identified shape. Knowing the area of the shape (the hoop in this case), the UAV can roughly determine distance to the hoop. As the UAV approaches the hoop, the area of the identified colored shape increases. As defined in the code, once a certain area threshold is reached, the UAV is notified that it has reached the front of the hoop. From this stage, the bottom camera comes into play. The bottom camera is utilized by the UAV to identify if it has passed through the hoop, or whether it needs to move forward to pass through it. The bottom camera utilizes the same implementation that was described for the front camera (the use of opencv as well as imutils). When the bottom camera sees the color of the hoop, it sets a over_hoop_flag which is used to let the UAV know that it is over the hoop and also keeps track for the next time it takes an image. Once this camera takes an image without the hoop present in it, the over_hoop_flag from the previous image is checked to see if the hoop was present. If it was present in the previous image, then the UAV is no longer over the hoop, and the UAV is then specified to look for the next hoop, having successfully passed through a hoop.

The previously mentioned algorithm focused on the hoop detection implemented by the UAV. In order for the UAV to autonomously navigate within its environment, the front camera images need to be analyzed further to inform the UAV where to move to. This is performed by calculating the midpoint of the shape found by the findContours function. A bounding box or region is specified at the center of the camera image to which the midpoint of the shape should be in order for the UAV to move towards the hoop. If the found hoop's midpoint is located to the left of this bounding box, the UAV is programmed to rotate clockwise slightly about its yaw axis. If the midpoint is found to be above the bounding box, the UAV is told to climb in elevation while, if the midpoint is below, the UAV will descend until the midpoint of the found hoop shape is located within the bounding box. A pictorial depiction of this can be seen in Figures 16 and 17. When the midpoint of the found hoop is located within the bounding box, the UAV is told to move forward and then check the next image for hoop shape area and midpoint location. Up until the hoop shape area threshold as mentioned above is reached, this navigation algorithm is followed. Once the area of the hoop surpasses the threshold, the UAV moves forward until the bottom camera sees the presence of a hoop and then its absence at which time the UAV begins to look for the next hoop by rotating clockwise or has already found the next hoop and is preparing to move towards it.

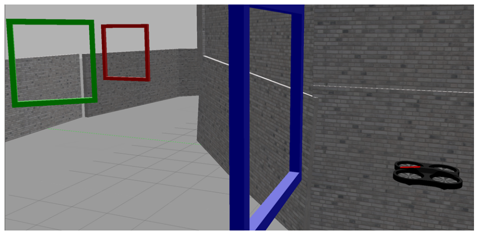

Fig. 15: The UAV as it is about to utilize the bottom camera to determine hoop presence and absence

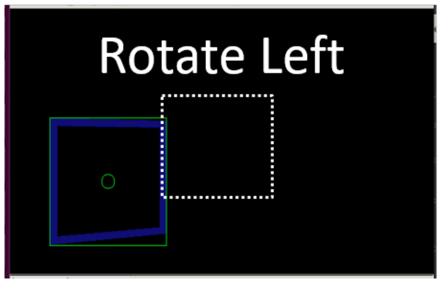

Fig. 16: Detected Blue Hoop with Contour and Midpoint Identified (white bounding box)

In order to try to prevent the UAV from getting lost, and to assist in hoop detection, the UAV begins its operation by rotating clockwise until it sees its first specified hoop color. This feature also occurs if the UAV does not see the specified hoop color within 30 front camera images.

C. Validation

The method of implementing this project is deemed successful if the UAV implemented in simulation can autonomously navigate itself through its environment (around a drone race course-like environment with a variety of colored hoops). After receiving a list of targets, the UAV should explore its environment and identify the targets or hoops and fly through them in the specified order based on hoop color.
Once the UAV has successfully navigated its environment and

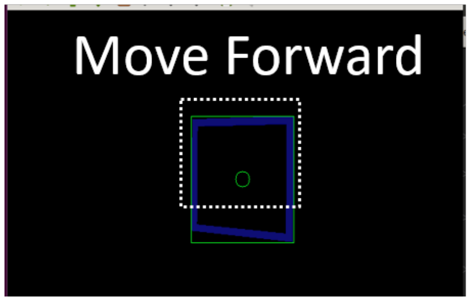

Fig. 17: Detected Blue Hoop within white bounding box; therefore UAV move forward

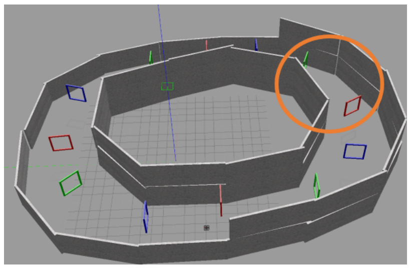

Fig. 18: The circled region is the longest stretch of track between hoops

autonomously identified and flown through all of the specified targets, the system could be timed and additional algorithms or changes to the algorithm could be made to decrease the time it takes the UAV to navigate the course.

The hypothesis is that the program may require a more Simultaneous Localization and Mapping (SLAM) oriented approach to autonomous navigation, although to initially implement the algorithm, a more simple autonomous navigation algorithm will be attempted.

IV. Results

A. *Expected Results*

As mentioned in the Validation section of this paper, the expected results will be that the UAV will be able to autonomously navigate through its virtual drone race-course environment based on ROS and Gazebo while also flying around identifying targets or hoop that it must fly through in a drone race course-like fashion. It is expected that autonomous navigation of the UAV will entail the most work in regards to algorithm creation.

V. Discussion

Looking back on the hypothesis made in the Validation section of this paper, it was not exactly correct as a SLAM approach was not required in order to implement autonomous navigation of a drone racecourse. Utilizing the method described in the Methodology involving Autonomous Hoop Detection as well as Autonomous Navigation, the UAV was able to navigate through most if not all the hoops. There was a section of the track which resulted in the UAV getting lost a few times. This stretch of track is located in Figure 17 as highlighted by the orange circle. Through analysis of the hoop shape area, it was determined that the distance between the green hoop and the red hoop were great enough that the findContours function could not find any red hoop components despite the openCV function being able to determine the color of the hoop to be red. To fix this an algorithm implementation utilizing SLAM would help determine the distance to the hoop as well as the hoop opening through which the UAV could fly.

When comparing the results of this implementation to various other drone racing implementations analyzed in the Literature Survey, it can be said that this paper's implementation probably yields slower drone racecourse lap times than another implementation which utilizes SLAM. Currently, the UAV would be able to complete the course in about 6 minutes. This slow lap time could be a result of the current unoptimized movement algorithm of the UAV, as the working implementation is somewhat jerky and lacks more precise movements. Delays between movements, velocities, as well as thresholds could all be optimized to yield a faster autonomous racecourse navigation. Using SLAM, the UAV would be able to better understand its environment and from this, be able to better navigate through it. If additional obstacles were to be added to the course, the UAV would definitely take more time to navigate through the course utilizing solely hoop detection and navigation. Despite this, since the method proposed in this paper produces autonomous navigation of a drone racecourse without the use of Neural Networks and Deep Learning like many other papers, the UAV is not required or limited to expensive equipment or a lot of computing power if this were to be implemented in hardware.

VI. Conclusion

In conclusion, autonomous UAV nagivation through a drone racecourse was able to be implemented in ROS utilizing the Gazebo simulation environment as well as the ROS packages, ardrone_autonomy and tum_simulator. In the end, the goal of the project was achieved, as the UAV was able to recognize hoops within its simulated environment and then from these identified hoops, maneuver its way around the race track.

If this work were to be continued in the future a few things could be modified or improved on in order to make this implementation more robust. First, the algorithm described in this paper could be optimized. A couple of the delay values, velocities, and threshold values throughout the program could be adjusted to result in the best performance of the movement of the UAV so that it could complete the race faster. Additionally, a monocular SLAM algorithm could be implemented using the front facing camera in order to enable the UAV to map its environment and to respond to obstacles and objects other than walls and hoops. One such example of SLAM which could be implemented is ORB-SLAM. Figure 19 depicts an image of a UAV utilizing ORB-SLAM to map a room. Additional sensors could also be added to the Parrot AR Drone 2.0 in order to make the system be able to implement more extensive autonomous navigation and mapping as well as better obstacle and object avoidance. This increased obstacle avoidance would be critical if racing with other UAV's on a drone racecourse.

Fig. 19: UAV utilizing ORBSLAM to map a room [11]

References

[1] "The Evolution of Drones: From Military to Hobby & Commercial." Percepto, 9 Feb. 2021, percepto.co/the-evolution-of-drones-from-military-to-hobby-commercial/.

[2] "Powering the World's Robots." ROS.org, www.ros.org/.

[3] "Unmanned Aerial Vehicle." Encyclopædia Britannica, Encyclopædia Britannica, Inc., www.britannica.com/technology/unmanned-aerial-vehicle.

[4] Wang, Shubo, et al. "A Deep-Learning-Based Sea Search and Rescue Algorithm by UAV Remote Sensing." 2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC), 2018, doi:10.1109/gncc42960.2018.9019134.

[5] Zhang, Mengmi, et al. "A High Fidelity Simulator for a Quadrotor UAV Using ROS and Gazebo." IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society, 2015, doi:10.1109/iecon.2015.7392534.

[6] Gao, Kai, and Kan Xu. "Research on Autonomous Obstacle Avoidance Flight Path Planning of Rotating Wing UAV." 2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC), 2017, doi:10.1109/icctec.2017.00261.

[7] J. Lu and B. Smith, "Autonomous navigation in drone racecourse," 2017 IEEE MIT Undergraduate Research Technology Conference (URTC), Cambridge, MA, USA, 2017, pp. 1-4, doi: 10.1109/URTC.2017.8284213.

[8] S. Jung, S. Hwang, H. Shin and D. H. Shim, "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning," in IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2539-2544, July 2018, doi: 10.1109/LRA.2018.2808368.

[9] Yu, Fengmin, et al. "Autonomous Flight Control Law for an Indoor UAV Quadrotor." 2017 29th Chinese Control And Decision Conference (CCDC), 2017, doi:10.1109/ccdc.2017.7978396.

[10] eborghi10."eborghi10/AR.Drone-ROS."GitHub,github.com/eborghi10/AR.Drone-ROS.

[11] ORB-SLAM Implemented Using Unmanned Aerial Vehicle .i.ytimg.com/vi/Cx94iW5SrGY/maxresdefault.jpg.