Autonomous Drone and Rover Navigation

Steve Gillet
Department of Electrical and Computer Engineering
Texas Tech University
Lubbock, United States of America
stephen.gillet@gmail.com

Rishikesh
Department of Electrical and Computer Engineering
Texas Tech University
Lubbock, United States of America
rishikesh.rishikesh@ttu.edu

Isaac Mondragon
Department of Electrical and Computer Engineering
Texas Tech University
Lubbock, United States of America
ismondra@ttu.edu

***Abstract*—Mechanics and methodologies for an autonomous drone and rover navigation project that is the Bachelor's capstone project for the authors and IEEE Region 5 Robotics Competition for 2023. The object is to have the rover and drone navigate around an area with obstacles while using camera data to inform one another where to go next autonomously.**

***Index Terms*—autonomous, computer vision, drone, object detection, rover**

I. Introduction

This document lays out the basic mechanics and methodologies used in a project for the IEEE Region 5 Robotics Competition for 2023. The rules of the competition state that a ground robot will go through cardboard boxes in a particular order that will be discovered by reading QR codes inside the boxes that have the designation of the next box to go to. The identity of each box is located on top and therefore a drone will be used to read the box ID's from the top while the ground robot reads the next designated box from the bottom. In this manner the ground robot will enter a box, read the QR code for the next box, the drone will find the next box, and then the ground robot will go to that box, and the process will repeat until all of the boxes have been entered at which point the drone will land on the robot and the round will end [1].

The drone is a Ryze Tech Tello Mini Drone Quadcopter that receives string commands and returns command acknowledgements and raw video data via wifi [2] [3]. The ground robot uses a Rover 5 Robot chassis, an L298N Motor Driver, 2 Tenergy 9.6V Flat NiMH batteries, an Orange Pi 5 computer, a Raspberry Pi USB camera mounted on a pan and tilt mount, and servo motors to aim the camera. The ground robot will connect to the drone via wifi and send the drone search pattern commands using a wifi socket in Python [7] [8]. The ground robot will use the camera to look for the entrance to the first cardboard box by singling out the shape and color of the box and its entrance using OpenCV [4].

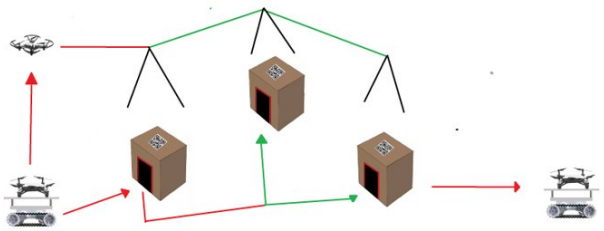
The ground robot will then enter the box and use its second, upward-facing camera to read the QR code printed inside the box with OpenCV's QRCodeDetector function. Then the ground robot will send a list of search pattern strings to move around until it finds the target box QR code, also using Pyzbar. The list of string commands that the drone takes is detailed in the Tello SDK [2] and is usually some simple word and measurement if necessary, 'forward 20' for example which tells the drone to move 20cm forward. The list is kept in a separate text file that the main Python program reads off of whenever the search pattern needs to be executed, this list is configured so that The drone moves in a lawnmower pattern. The ground robot then goes to the position determined by the drone's scanning and movement. So, the list looks something like: 'forward 90', 'right 40', 'back 90', 'right 40', etc.

Once the drone has found the QR code it saves the position and if the ground robot has the corresponding QR code as the next QR code to go to then it will move to that position. Once the ground robot is close enough to the box to navigate around it, it focuses on the box and uses the OpenCV color and shape navigation for the box again [5].

Once the ground robot goes through all of the boxes the drone then navigates to it using the ground robots camera to give the drones commands until it is directly over the camera using pixel coordinates [6].

This project combines the use of ground robots and drones to navigate through a set of cardboard boxes in a specific order using computer vision and wifi communication. The following sections will go into detail about the components used and the methods used to achieve this task.

II. Project Sketch



Funded by Texas Tech University Department of Electrical and Computer Engineering.

III. Groundbot Software

The Orange Pi 5 utilizes Armbian 11 OS and Python 3.10 for operating the ground robot. The main script, runCode.py, employs various libraries for timing, threading, GPIO, and PWM control, and includes custom functions to modularize the code.

A key function connects the system to the drone autonomously. The GroundBot class is crucial, as it houses controls and variables for the robot. Its methods enable the robot to move, controlling the motor driver input pins and speed via the PCA9685 I2C breakout board library.

The camera's pan and tilt are controlled by a function that takes input from the image processing code. A PID algorithm smooths out the movements, while the servokit library sets the servos to new values. The np.clip function prevents values from exceeding limits.

Additional methods employ the MPU 6050 to track the groundbot's yaw, ensuring consistent heading returns, and a wait function allowing the groundbot to pause until its location is updated by the drone search code.

The camera class, initialized within the groundbot's method, manages the pan and tilt values. In the main loop, the drone initiates its search pattern and enters the main loop. It reads the box, backs out, corrects its orientation, and waits for the drone to find the next QR code position.

If the next QR code value is "DONE," the groundbot moves to the last position and stops. Otherwise, it proceeds to the next position and continues the loop, ensuring it stops a meter before reaching the correct box.

IV. Drone Software

The drone code comprises two primary functions: 'tellopath' and 'QR tello'. The 'tellopath' function outlines the drone's search path, while 'QR tello' processes the frames from the Tello drone, searching for QR codes and updating the position attribute based on the QR code value.

The 'send command' method controls the drone using simple strings, like 'forward 20', which instructs the drone to move forward 20 cm. The drone moves in a lawnmower pattern, going forward, right, and then backward.

The 'QR tello' function updates the 'box position' attribute using barcode data from the 'decode' function of the 'pyzbar' library. Both functions run concurrently using the 'threading' library, allowing separate code pieces to execute simultaneously.

The lawnmower pattern covers a 20 by 30 feet area, ensuring the entire playing field is covered when the groundbot starts at a corner. Running the entire drone code on threads allows the groundbot's main control code to run continuously while the drone moves simultaneously.

V. Groundbot Box Navigation

The 'readBox' function handles groundbot movement, camera aiming, and QR code reading when the groundbot is near an individual box. This script includes a 'timer function' for determining the groundbot's movement duration and an 'equalizeHistograms' function to control for lighting changes. Color masks are fine-tuned for competition lighting, singling out specific colors, like the red tape around the box's entry.

The code identifies the red frame by applying masks, finding contours, and locating a square shape with a specific width in pixels. If the red shape covers 85% of the screen, the groundbot is considered close enough to begin searching for the QR code. The camera tilts up, and the groundbot moves slowly forward until the QR code is found. If detected, the 'nextQRcode' attribute is set, and the loop returns to the main code.

A secondary code handles cases where the box's entrance is not visible from the drone's perspective. It searches for the box's side, turns the groundbot, and moves it in a forward arc until the entrance is seen. The orange mask looks for the branding on the side of the box, and the groundbot is controlled to turn and move using the 'slowRight' and 'angleLeft' functions.

VI. Groundbot Movement

The final crucial code section defines how the groundbot moves to the next box's position as provided by the drone. The code is relatively simple, with the groundbot moving the designated distance in the x direction, then turning and moving in the y direction. The x and y directions are maintained using the yaw attribute. The groundbot turns until the yaw corresponds with a 90-degree right or left turn, based on whether x is positive or negative. Then, it moves forward with the 'move' function unless an obstacle is detected.

If an obstacle is identified as a large contour in the frame, the groundbot performs a simple square movement and resumes normal movement afterwards. Finally, the robot moves in the y direction.

VII. Hardware

The primary pieces of hardware for this project are the DJI Tello Drone, Rover 5 Chassis, Orange Pi 5, L298N H-Bridge Motor Driver, 2 Tenergy 9.6V Flat NiMH batteries, pan and tilt servos, camera, and PCA9685 breakout board.

The DJI Tello Drone is an educational drone that can be controlled using the Tello app or the DJI Tello SDK. It offers 13 minutes of flight time per battery and is valued for its safety, simplicity, and ease of use. The drone's camera was modified to face straight down for QR code reading. The Rover 5 Chassis is a simple rover body that includes tread, two motors, and a motor encoder, which is unused in this project.

The Orange Pi 5 is a single board computer with a Rockchip RK3588s octa-core 64-bit processor, ARM Mali-G619 GPU, and an NPU for 6Tops AI computing power. It was chosen for its affordability and powerful CPU. The L298N H-Bridge Motor Driver is a dual-channel full H-Bridge, allowing two DC motors to be controlled independently. This motor driver was selected for its affordability, simplicity, and compatibility with the Orange Pi.

Two Tenergy 9.6V Flat NiMH batteries were used on the groundbot, one for powering the motors and the other for the computer. Miuzei SG90 Servo Motors are used for the pan and tilt camera mount, chosen for their accessibility and 180-degree turning ability. The camera used was a Raspberry Pi camera, chosen for its USB connection as the Orange Pi 5 uses a different CSI connection.

Lastly, the PCA9685 16-Channel PWM Module is 5-volt compatible and can be run at 3.3 volts. It was used in this project because only three PWM pins were available on the Orange Pi 5, and four were needed for the four motors (2 DC, 2 Servo). All that is needed for the PCA is one available I2C channel.

VIII. Conclusion

The project was able to complete all of the objectives independently but faced challenges during the competition. The code for the groundbot to find the entrance of the box was inconsistent, primarily due to lighting. The reflectiveness of the red tape, which was intended to help, ended up causing issues as it increased the variability in lighting conditions. In hindsight, employing additional sensors or more image processing techniques simultaneously could have addressed this problem.

The drone search pattern also did not work out as expected, as the starting point was inconsistent. This inconsistency revealed the need for a more robust search pattern, such as one that allowed the drone to stop when it saw the boundary of the competition ground. Despite these challenges, the integration of all components and the implementation of an automatic starting mechanism were successful, although it was more difficult than expected and required additional time and effort.

Moving forward, it is important to address the limitations encountered during the competition. Developing a more adaptive and flexible solution for handling lighting variability and improving the drone's search pattern to accommodate inconsistent starting positions are essential steps to enhance the performance of the system.

IX. Acknowledgements

We would like to express our sincere gratitude to our partners and co-authors, Rishikesh and Isaac Mondragon, for their invaluable contributions to this project. Their hard work, dedication, and collaboration were instrumental in the success of this project. We would also like to thank our professor, Dr. Hemmert, for his guidance, support, and encouragement throughout the project.

Additionally, we are grateful to our advisor, Dr. Nutter, for providing us with valuable feedback, insights, and recommendations that helped us to improve our project. We would like to thank the IEEE and Texas Tech University for their support and resources that made this project possible. Finally, we would like to acknowledge the countless individuals who have contributed to the development of the technology and tools used in this project.

Thank you all for your support and guidance throughout this project.

References

[1] "2023 IEEE Region 5 Annual Conference Rules for a Student Robotics Competition".
[2] "Tello SDK".
[3] "Tello User Manual".
[4] H. Kato,F. Nagata, Y. Murakami, and K. Koya, "Partial Depth Estimation with Single Image Using YOLO and CNN for Robot Arm Control," 2022 IEEE International Conference on Mechatronics and Automation (ICMA), 1727-1731, 2022.
[5] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios," CoRR, abs/2108.11539, 2021.
[6] G. Badakis, M. Koutsoubelias and S. Lalis, "Robust Precision Landing for Autonomous Drones Combining Vision-based and Infrared Sensors," 2021 IEEE Sensors Applications Symposium (SAS), Sundsvall, Sweden, 2021, pp. 1-6, doi: 10.1109/SAS51076.2021.9530091.
[7] D. Vaish, "Python Robotics Projects: Build smart and collaborative robots using Python ," 2018
[8] L. Joseph, "Learning Robotics Using Python," 1st ed., PACKT Publishing, 2015.