Lab 6 : Computer Vision Controlled Drones

Abstract

This report discusses two parts of a computer vision lab, as well as the challenges that were encountered in the process and future work that can be explored. The first part explains the objectives, methods, questions, and challenges faced when working with a drone and computer vision in the same degrees of freedom as we would with a grounded vehicle. The second part goes into the detail on the objectives, methods, questions, and many challenges that were faced when adding an additional degree of freedom. After the discussion of what happened during this lab, potential future work is detailed about how this lab could be improved in the future, as well as a conclusion summarizing the overall findings, challenges, and lessons learned.

Introduction

This lab explored the use of computer vision to examine the environment surrounding a robot, and the use of robotics in drone flight. The primary tools that were used to complete the two parts of the lab were the TelloPy Python API, for communicating with the drone, as well as OpenCV's Python distribution, for handling the computer vision component. These tools were utilized to control the drone, as well as to process photos that were taken and examine the environment. Based on the computer vision component, the drone would make decisions on where to go next.

In part one of the lab, we familiarize ourselves with the basic parts of TelloPy and OpenCV, learning how to move the drone around with code and also how to use OpenCV to detect shapes. The second part of the lab focuses on expanding the first part to an additional degree of freedom, being able to move the robot up and down. Although there is not a substantial change in the tools being used, the logic changes significantly, as do the potential challenges that can arise. The main challenges encountered throughout the lab involved the inconsistency that computer vision has when used alongside a low resolution camera, as well as the difficulty of testing the drone in a safe way.

Part I

Ia. Introduction

This section of the lab was mostly about getting familiar with the tools. The Tello drone was to be programmed using TelloPy, and the contour detection algorithm using OpenCV also had to be written at this stage.

Ib. Objectives

For part one, the goal was to familiarize myself with the inner workings of the two modules, including OpenCV and the Tello drone, with the end goal being a drone that is able to fly around

the perimeter of the room, given that the appropriate markers (in this case, rectangles) were always present.

Ic. Methods

1. Implemented basic contour detection in OpenCV, testing on random pictures pulled from the internet
2. Ran the contour detection on pictures taken by the drone. After doing so, used the results to modify the contour detection algorithm until it was reliably detecting the correct shapes
3. Added functionality for the robot (drone) to take off, move forward, rotate counter clockwise (for turning left), and land
4. Added logic functionality to the contour detection, for now making it possible to search for rectangles with a sufficiently large area
5. Using the contour detection and movement, implemented the decision making process for the drone. At each step, it would move forward, search for contours, then decide whether or not to turn. (steps were limited to prevent the drone from crashing unnecessarily)
6. Ran many tests, tweaking values for rotation, movement, takeoff, and time.sleep calls, until the drone finally performed as expected

Id. Challenges

The largest challenge faced in this part of the lab was the inconsistencies that I encountered while using OpenCV, especially with pictures taken by the drone's camera. Using various methods and settings made available by OpenCV, it was possible to improve the consistency dramatically, but there were still times where the contours were not detected correctly. For this part specifically, I was able to get it to a point where it was relatively consistent, but the consistency became an issue again in the next part.

The other challenge was the difficulty in finding a suitable place to fly the drone. In fact, due to restrictions around flying drones on campus that were recently implemented, the lab could only be performed in my room, which only had a corridor-sized open area. Even without the drone policy, it would be difficult to find a location that was open enough, with nobody around, to make sure no person or property was hurt or damaged. As a result, only the decision making could be tested, and there were no tests conducted to see whether the drone could actually traverse the room's perimeter.

Part II

IIa. Introduction

In this part of the lab, I explored the fourth degree of freedom that the drone had. Unlike other robots we have used in the past, which included translational movement over the x and z axes, as

well as rotation along the y axis, the drone provides movement along the y-axis as well, introducing a fourth degree of freedom.

IIb. Objectives

The main objective of this part was to allow the drone to traverse the perimeter of the room, even with obstacles in the way. So, where other robots such as the roomba would fail, the drone would be able to overcome these obstacles and continue along its course. This was accomplished by adding more shapes to the environment, circles and triangles, which eventually would sort of give the drone this capability.

IIc. Methods

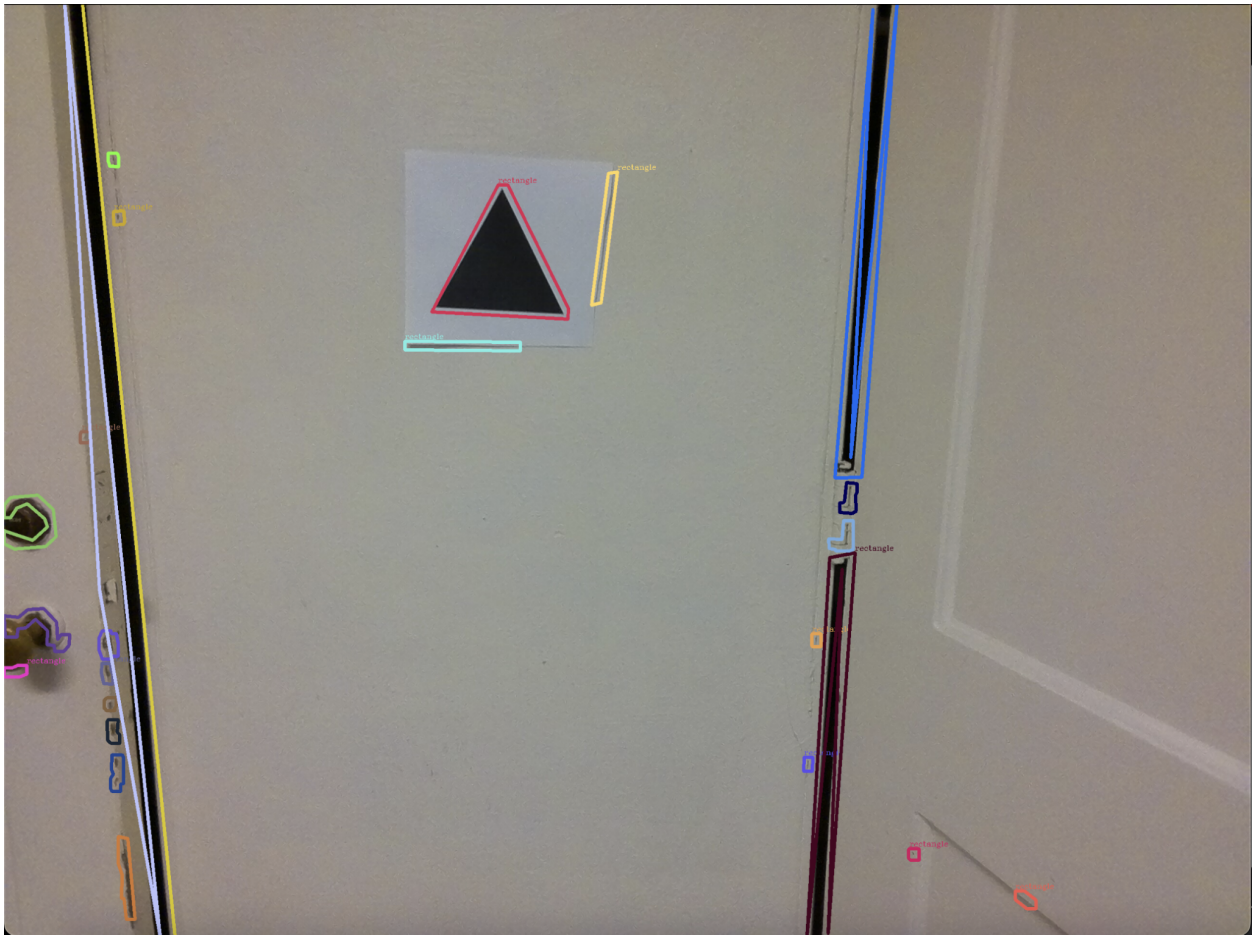
1. Added functionality for the robot (drone) to be able to move up and down, thereby taking advantage of its additional degree of freedom
2. Modified the handle_contours method to now search for circles or triangles of a certain area
3. Rewrote the decision making code, so that the drone can now decide to move forward, or upwards. While moving forwards, it will be searching for circles. Once found, it will start moving upwards, and search for triangles.
4. Modified the handle_contours method to dilate slightly more, to improve the reliability of finding circles.
5. Added functionality for stopping the drone in place, to prevent some of the drifting that would happen after the drone accrued multiple instructions
6. Tested the drone, and tweaked values until the drone performed as expected

IId. Challenges

The challenges for this lab the same as those encountered in Part 1, but were amplified by the added complexity that the fourth degree of freedom created. When it came to finding a suitable space, there weren't any other options, but adding verticality to the drone meant that it was now possible to really hurt bystanders, so the experiment could only be performed with nobody around. Additionally, the drone was affected significantly by the air current in the room caused by the air conditioning system, and would drift, so I had to implement a method to tell it to freeze in place.

As for the computer vision component, the final image used to find contours in Part 1 was not sufficiently accurate for both circles and triangles, so some adjustments were made to the dilation to ensure that the program was able to capture the shapes correctly. This led to some interesting results; for example, in the example video, the image below shows the contours that were captured by OpenCV. Note that there does not appear to be a triangle sufficiently large drawn,

yet OpenCV did find a triangle that met the criteria and told the drone to land anyway. This version of the image was processed after the experiment, although with no other changes made to it, so it is difficult to tell if this was how the program perceived it at the time or not. Regardless, it really highlights the inconsistencies with OpenCV.



Future Work

In this lab, I spent most of my time working on the OpenCV output, which in the end resulted in only marginally better results than the original, inconsistent ones. There are potentially a few different ways to try to improve this, with the most obvious one being further adjustments made to the input that OpenCV receives. A more elegant solution would be to process the video stream from the drone in OpenCV, likely checking every nth frame, with n depending on how quickly frames are received. At the moment, the drone processes photos taken in between movements, so running the computer vision on a separate thread would allow for continuous movement and decision making. The advantage to this with respect to consistency would be the ability to check results with those from the past few frames, and identify outliers

that would result in an incorrect decision. Of course, this all relies on OpenCV, and a more optimal solution would be to attach sensors on top of the drone, which would allow it to sense whenever it is sufficiently close to a wall. There are examples online of how to do this, but it requires some way of communicating the sensor output, which would be separate from the drone, to the robot program.

Another cool idea would be to have the drone solve a 3D maze, using either computer vision or added sensors, with a planning algorithm similar to how we did in Lab 3. This would require a planning algorithm, potentially pyhop, or a backtracking algorithm, since the simple idea of hugging the left wall would not work with a vertical component added. However, this should only be used if proper sensors are used, as the computer vision is simply too unreliable to solve a maze.

Conclusion

The major lessons learned from this lab were how to use computer vision in place of sensors, as well as how an additional degree of freedom increases the complexity of decision making for a robot.

I encountered many challenges with the lab due to difficulties testing the drone in a safe way, as well as in the inconsistencies with OpenCV. The majority of my time was spent on trying to make the computer vision more consistent on images taken from the internet, which provided for ideal conditions that were not necessarily met when the drone was taking pictures. Considering the many challenges that computer vision caused, it would have been more useful to use proper sensors instead. This would improve reliability and make it safer to try testing the drone in other environments. Nevertheless, it was still possible to implement a rudimentary functionality of decision making that is inherent to robotics.