Joshua Pulido, Pre-Lab 6

0. Lab 6 will build upon Lab 2, where we used the Create 2 robots to traverse the perimeter of the room, using sensors to detect when a turn needed to be made. However, instead of using a Create 2, we will use a drone, which will introduce an additional degree of freedom as it can now fly up and down, alongside forwards, backwards, left, and right. The interesting part of this lab is that the Tello has no forward facing sensors, it only has a forward facing camera, which will have to be used as a sensor somehow.

In order to accomplish this, we will be using computer vision technology to identify basic shapes (i.e. rectangles, triangles, circles), which will hopefully provide a very basic depth perception which will tell the drone how close it is to a wall and help it decide where to go next. These serve as a basic "sensor" for the drone, allowing it to interact with its environment even without official sensors like those found on the Create 2.

My goal with this lab is mostly to learn about computer vision. While it will be used in this case to make a simple robot that flies around the room, I am hoping to use this same drone for my comps next semester, and one of the things I might use is computer vision as the lack of sensors make it very difficult to interact with the environment.

1. Specifically, I will be using the Tello drone by Ryze and DJI, which has a full API, allowing it to be flown using instructions from a computer (similar to how we worked with the Create2). This provides a few advantages, first of which is that it is small enough to be safely flown in a room; it can not cause any significant damage to anything or anyone as long as proper precaution is taken. The other advantage is the popularity of this drone within the programmer community, which means that there are many APIs and sample projects that have been made available online and are the reason that a project like this is even possible. Alongside it's advantages, it has no real disadvantage to using a "normal" drone, as it still has access to all 4 degrees of freedom that any other drone would have (Moving in x, y, z, and rotation along y).

2. For this library, we will be programming in Python, as there are two existing libraries for the drone and computer vision, respectively: TelloPy and Python OpenCV. This also allows for the two libraries to communicate effectively.

a. Install TelloPy using the following command:
   pip3 install tellopy
   TelloPy will be used to send commands to the Tello drone, notably to tell it to move forward and rotate (much like we did with the Create2 and pycreate2).
b. Install Python Open CV 2 using the following command:
   pip3 install opencv-python
   OpenCV2 will be used for the computer vision part of this lab, and we are specifically installing the full version as it should help to have access to all features, particularly a GUI. There are headless versions, but it might be near impossible to figure out what's going on with those, so the standard installation is recommended.

c. Check this repository to see how to communicate between OpenCV and TelloPy. The repo is 4 years old, so there might be better ways of communicating between the two, but something similar should provide the functionality needed.

3. We will split the lab into two parts, first getting the robot to traverse using two translational degrees of freedom, and then adding the vertical aspect to it in a second part. I split it this way to keep the discovery of how things work as simple as possible, with the option of adding more complexity after in a second part depending on how things go.

Part 1

In part 1 of the lab, we first make the robot travel the perimeter of the room.

a. Using TelloPy, make the robot launch, go forward 1 meter, turn around, go forward 1 meter, and land. This should provide all the useful methods on how to make the robot move.
b. Using TelloPy, take a picture of the robot, and store the picture locally.
c. Open the locally stored image with opencv, and apply a grayscale affect to it.
d. Use OpenCV to find the contours of the image (look up contours if you are unfamiliar). Write code to recognize a rectangular contour; this will be the object that the drone will sense. (You should also tape a rectangular shape to the wall, roughly where the drone will hit, so that it is in view.)
e. Modify the code above to also check the area of the contour. Based on this area, the drone will decide if it is too close to the wall and should turn.
f. Put the movement, picture taking, and contour detection together to program the robot to move forward slightly, check for a rectangle, make a decision, then repeat.

Part 2

In part 2 of the lab, we will add to the code above, adding more decisions for the robot to make (allowing it to fly vertically).

a. Add more shapes to the contour detection, this could be a circle, triangle, pentagon, etc., whatever is reasonably easy to add in OpenCV.
b. Add code that makes the robot fly upwards upon seeing shape 1 (i.e. circle), until it sees shape 2 (i.e. triangle), then rotate the robot 90 degrees
c. Add code that makes the robot fly downwards upon seeing shape 2 (i.,e. triangle) until it sees shape 1 (i.e. circle), then rotate robot 90 degrees
    1. Note that the rotation is happening before the robot tries to make another decision. This is to prevent it from, for example, flying up, seeing a triangle, and deciding to fly back down, resulting in an infinite loop
d. Check that the robot is able to overcome obstacles
e. ALTERNATIVELY: Make the robot fly up a set distance after seeing a shape (i.e. circle). To make a robot go higher, place multiple circles on the wall. This is a simplified way, but would likely be less accurate.
f. ALTERNATIVELY: See if the drone is able to handle verticality in other ways (using the bottom sensors alongside the front camera, without using computer vision and contours).

Other Optional Possibilities:
1. It is possible to make the robot send a video stream, rather than intermittent pictures. This would provide for a much more seamless flight pattern, but would require decision making to be run asynchronously. Unfortunately, the original Tello drone, which is the one used for this lab, is difficult to work with asynchronously, so this is more of a stretch goal, but it is possible.

Included alongside this PDF is a sample of the current code I am running to test the computer vision. I have not yet started flying the drone, but it seems like the CV part for sensors is going to work (although potentially not reliably depending on the surrounding environment). Will continue working on this to make it more reliable, as OpenCV has a lot of options that can change the output contours.