Matthew Meloni & Craig Sampson

ENGN2045 - Computer Vision Project

Professor DiFilippo

**Alignment of circles using Arduino and a Webcam**

The goal of this project was to use the Open CV and Pyserial libraries in Python to sync with the Arduino code. In this project, we used a Webcam that sends video frames to Spyder which contains the Open CV library. From Open CV, the program was to interact with the webcam and create a grayscale image. After we got a grayscale image, we made the program create a binary image of the output, so the computer can analyze it. Also, we had to make the contours of outputted image, so we used the findContours function and the program drew the contours onto the outputted image.

The project parts called for an Arduino Uno R3, a breadboard, jumper wires, a USB 2.0 Power Supply, two micro servo motors, and a webcam.

In order to complete this project, three libraries in python were needed and one in Arduino. In python the first library I utilized was time, this was used for small delays in the code to make sure that spyder was connected to Arduino. Without this delay the python code would move on too fast and the connection could not be established. Time.sleep() was the function used from this library. Next a more important library in the code was cv2. It was used for several things including initializing the webcam, capturing contours on the live video, and also finding the center of the contour seen on the webcam. In order to initialize the webcam the cv2.VideoCapture() function was used and then stored as the variable cap, after that cap.read() was used to read the video coming in and was stored in the variable frame. Throughout the code, frame is used as the webcams output. Next, in order to capture contours on the webcam, cv2.cvtColor was used to grayscale the image, cv2.GaussianBlur then slightly blurred the grayscale image then finally, cv2.threshold turned the image into a binary image which allowed cv2.findContours to work properly. Next, cv2.drawContours was used to draw a blue line around the contour found. Cv2.moments is another cv2 function I used in order to find the center of the given contour. The final library I used in my python code was serial. This is a library that allows you to read and write data from your USB port on your computer. In this case it is reading/writing to/from an Arduino uno. I use this library to send individual letters from python to Arduino, and what those letters did was tell the servo motors which direction to move, depending on where the contour was located in the cameras view.

In my Arduino code I used one library, and that is the Servo library, which is used to control the two micro servo motors. More specifically, it is used to attach each motor to specific pins on the Arduino board, and then write the angle depending on which letter was sent over from python.

Next I will explain how each code works and what exactly the communication between them does. To start off in my python code, I begin by initializing my webcam, then creating a function called estConn, which establishes the connection between python and the USB port (Arduino). If this is successful then it will print ‘Arduino is all set’ and that tells the user that we are all set, and the code can continue. Next, the entire main part of the code is put inside a while True: statement, which allows it to constantly run and update. Inside this while statement, our actions on the frame come, grayscale, blur, then binary image. Next, I run a for loop, which also constantly runs and allows the drawContours function to constantly update even if the camera/contour moves. I then check to see how big the contour is and if it is not bigger than about 10,000 pixels the code will not run and print, ‘Contour bigger than 10000 not found’. Once a big enough contour is recognized the next part of code will run, and what this next part does is the most important, it first finds the center of the contour and then draws a single blue dot there. Next, it checks if the contour is centered in the camera, if it is, then the servo motors will stay where they are, and ‘the camera is centered’ will be printed out. On the other hand, if the contour is not centered, ‘you have to center the camera’ will be printed out. Once it is established that the camera isn’t centered a series of if, elif statements determine which direction the camera must move. And once the direction is determined (left, right, up, down, up and right, up and left, down and right, down and left) each direction has its own letter that is sent to the Arduino code using serial. For example, left is ‘l’, right is ‘r’, up and left is ‘b’. the way it is sent is using a function I wrote called sendData.

def sendData(byte):

serial\_arduino.write(byte.encode())

y = serial\_arduino.readline().decode('utf-8')

print(y)

The code for that function is written above, what this does is it takes one argument (byte) and what that will be is a single letter, determined by the if, elif statements and it sends that letter to the Arduino. So, in order to make the camera move to the left this is what you would write: sendData(‘l’). This function also reads the letter just sent to the Arduino and print it out just to make sure that it was sent.

My Arduino code was much simpler, because all it needed to do was read what was sent from python and then update the servo motors angle. So, I start off by running a for loop which says, if something is being read from python then run the code. In the estConn function in python I send an ‘i’ to the Arduino and this is what initializes the connection. In my Arduino code, I read the first thing written and if it is an ‘i’ then it will println ‘hello’, and center my cameras. After that first letter is sent, depending on where the contour is located relative ti the center of the camera different bytes will be sent at a very high speed and several per second. There are 8 else if’s that say, if the received byte equals ‘this letter’ then run this function and delay 15 microseconds. At the end of the if, elseif’s the value is written to the servo motor that needed to move.

A big problem that I ran into was getting errors if the camera did not recognize any contours and also if there was too much light the camera would draw too many contours. When it would recognize none, the program would crash and the window that shows the cameras output would not respond for 10-20 seconds. So, in order to fix that problem, I added an if else that says, if a contour bigger than 10000 pixels is found (the rectangle I used was about 12000 pixels) then run this code, find the center, and move the servos. But if no contour bigger than 10000 pixels is found it would continuously print out ‘Contour bigger than 10000 not found’ this got rid of the program crashing and not responding. This also fixed the error of drawing too many contours if light were added to the workspace, if a contour bigger than 10000 was not found it would not draw a contour at all.

Some things that could be added to this project in the future is changing the distance that the servo motors go depending on the distance that the center of the contour is. for example if the contour is a bigger distance away then the motor will move 5-7 microseconds instead of the normal 3 that I set it at.