Notre Dame University

Faculty of Engineering

Spring 2016

Instrumentation Lab

Final project

Wissam Antoun

Suleiman AL-Yaman

1. Introduction:

We were asked to design a small robot that is capable of gathering and collecting balls, with the ability of distinguishing between colors, then it will dispose all the collected balls in the enemy territory.

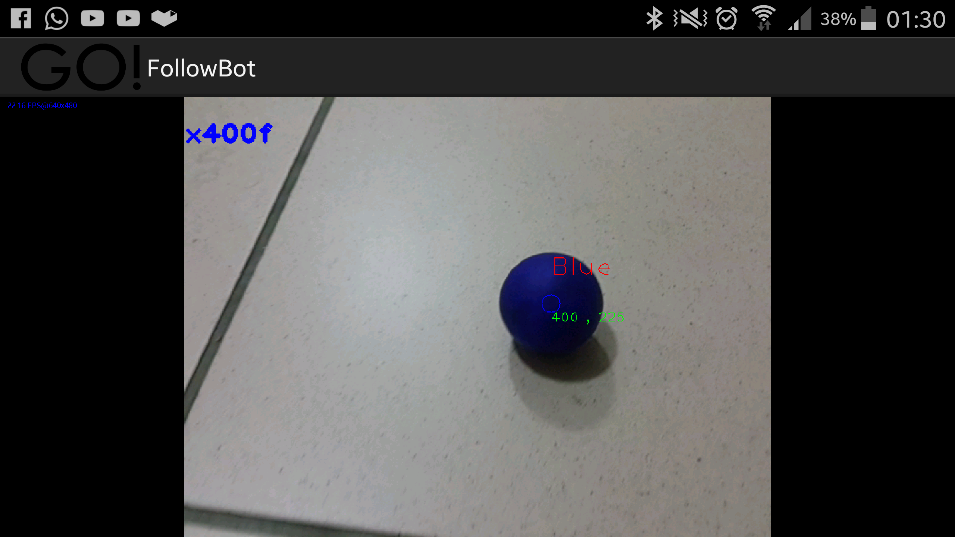
We realized that the only way two accomplish such task is to use computer vision. So we opted for OpenCV for android java because we can upload the program on a phone that is sitting on the robot without any wires.

1. Android Java Code:

The center of the java application is the onCameraFrame() function, this function gets called on each frame.

In each frame, the app starts by initializing the rgb hsv and threshold matrices, then it will convert the rgb to an hsv image that openCV can understand and work with better. Next the app checks the timer value (explained later). The timer initially is true, so the app will start by searching for any blue balls using the detectblueballs() function. If it finds any ball it will store it in the blueballs list. Next the app will search among all the detected balls to find the closest one to the robot. Then it will send the selected ball X position to the Arduino, and will mark its position on the screen. If no ball was found the app will tell the Arduino to turn left by sending it a fake ball with x pos less than 270.

In the meantime, a timer was counting down from 1 min, when it finishes it will set the timer Boolean to false then the program will switch to search for the wall to throw the collected ball and will turn the phone 180 degrees. The wall will have green stickers placed at equal distances from each other. The app will search for these stickers and will treat them like green balls and place all the green balls in a green list. But if we told the Arduino to go to the closest green ball it might not go in a straight way to the wall, so we came up with a way to fix the direction of the robot.

Figure - 400f means turn right since 400>320

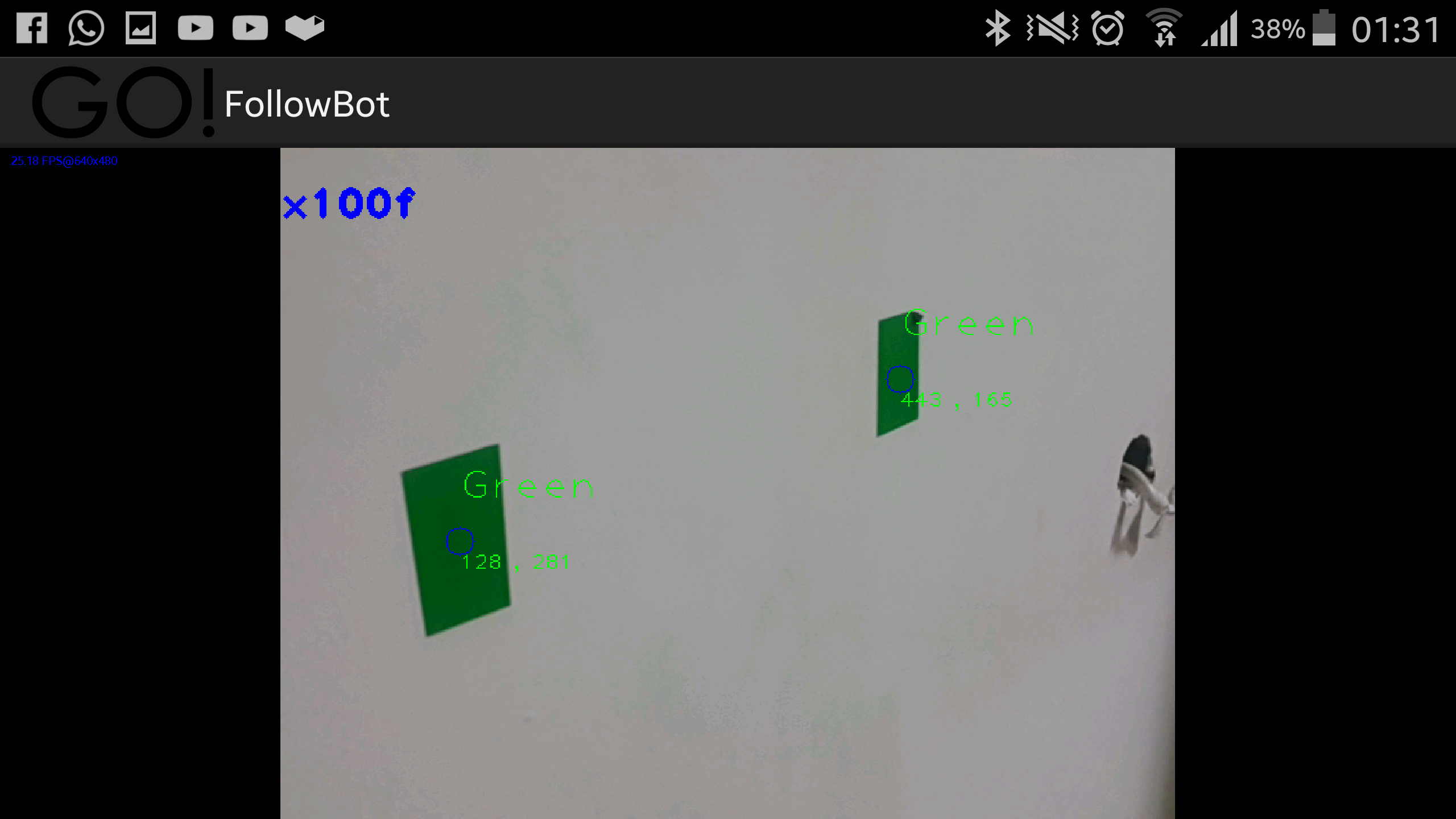


Figure - 100f means turn left

We know that the phone will be placed higher than the stickers on the wall, thus in a 2d picture the stickers will appear with different y-pos even though there are at the same elevation from the floor. This will always happen unless the camera is facing straight at the wall and not a tilt angle. Thus the program will first search for 2 balls that are closest to the center the image one from the left and the other from the right. Then it will compare their y-pos , if the left ball is higher this means that the robot should turn right, and if the right one is higher than the robot should turn left, it will keep doing that till the balls are more or less on the same y-pos then it will start going backwards till the x-pos separation is greater or equal than the separation distance seen by the camera when the back of the robot hits the wall. Then it will tell the Arduino to dump all the balls, turn the phone back, and restart the timer.

The detect blue and green balls function are the same except each one search for a different color. The function starts generating a threshold matrix that contains 1 in places where the color is in a given range. Then we apply morphological operation the threshold matrix (dilate and erode) which removes noise and Isolate individual elements and join disparate elements in an image). Then a temporary copy is created because the findcontours function will mess with the threshold matrix (used in testing the accepted ranges by replacing the return mRgba in the onCameraFrame by return threshold in order to see the result). The findcontours will find the balls and then if the app did detect more than 10 balls that’s probably noise and the filter range should be fixed. If there was less than 10 objects will loop around them creating image moments.

The m.00 is used for area, so if the area is greater than 20x20 object then we create a new ball and assign the x and y pos from the m.01 and m.10 and we add the ball to the appropriate ball list.

The Data is sent over blue tooth using the Bluetooth bee module for Arduino.

1. Arduino Code:

The Arduino code is much easier it only handles commands received over Bluetooth. First we start by getting the string sent from the phone then dissecting it into 3 parts: the First should be the letter x or else the program will skip, then comes the x-pos and finally the command (d for dispose, f for forward, r for rotate and s for stop).

The forward command will use the x-pos to tell if the robot needs to go left or right or even back if the forward bool is false. The reverse command will only turn the phone 180 degrees and set forward to false. The dispose command will turn the flip servo then stop then flip back (the flip servo is a continuous servo which means that we should specify his speed not his angle, thus the delays we obtained from experimentation) then it will turn back the phone to its initial position and set the forward to true.

1. Testing

A 12V battery was too powerful for the robot since it made it move so quickly that phone camera could not follow with the rapid movement. A 9V battery would have been better.

The dispose functionality was tested and proven to be quite good but unfortunately we weren’t able to get that on video.

1. Conclusion and future ideas

The end project was quite successful , but too many variables were involved thus proving that a project like this needs more calibration and processing power For Future ideas we might add collision prevention mechanism using proximity sensors mount on the robot.