Lab 4 Report

Task 1-3 revolve around getting acquainted with the e-puck's camera and object recognition in the world environment. Due to family circumstances, I will have fewer days to work on this lab than I'd like to, so I began working on it the moment the lab was released. The camera is going to be a compelling tool to use in the Webots software going forward. The world file lets you choose which colors you want the camera to recognize as an obstacle and once the camera sees that color it forms a red box around the object that shows all its details when you hover over it.

[Task 1 – Goal Facing] Task 1 just wants us to spin our robot in place until the camera picks up the goal object. For all three tasks the goal object is a yellow cylinder that's 2 feet tall and 4 inches in radius and because of the material that's being used is both non-reflective and bright in color we should be able to recognize it from up to 8 feet away! At first the prints from the camera functions made little to no sense to me but studying the documentation made it more clear and surprisingly simple. I decided to use the position_on_image() function's X value to center the object on camera before stopping. I had this task totally completed and was part way through task 2 when I accidentally deleted all my work from my PC hard drive during a routine cleanup…what a time to be alive. This will likely not be the last time I give myself more work because of my own recklessness so I'll just keep calm and carry on.

[Task 2 – Motion to Goal] Motion to goal sounds easy enough but using PID with the side sensors in the last lab was a lot of trouble for me -and many of the students in the course I was relieved to hear- so I stepped into this one a bit anxious. To my surprise the professor provided us with a python solution for how side PID works and how it's used to wall follow left and right walls. An important note is that the last lab said specifically to "*apply the same control values to the two side sensors at the same time*" which is where most of us got stuck whereas this solution chooses either a left or right wall. With the work I did on this unintentionally erased I felt completely deflated but with the given solution in hand, my old Lab3 code, and renewed momentum this task was accomplished on time. Time for the tough one.

[Task 3 –Bug Zero Algorithm] The Bug 0 Algorithm is a sort of logic for the robot that in principle will get it from a starting point to a goal without knowing what obstacles are in the environment between start and goal. It's a simple algorithm so it has been known to be faultier than some of the other algorithms we have studied in class and will sometimes entirely fail to reach the target goal. The algorithm simply tells the robot to [MOTION TO GOAL] drive directly toward the goal in the case that there is no obstacle blocking its path, or [WALL FOLLOW] drive along the wall in one direction (left or right) in the case that there is. I was not able to finish this task one week before due date which is the last chance I have to work on it.

I can't seem to get through a week without being stuck on a problem, something going wrong with a project, some typo or technical issue, or just some good old fashioned human error knocking me on my –but despite my magnetism to misfortune I couldn't help but have fun with this lab. The camera is such a powerful tool to add to our Webots toolkit, and I think the recognition of objects and the first-person view screen for a change in perspective adds a lot of 'cool' factor to programming and driving around these little robots. I've spent a lot of time these last couple years working on difficult projects that in the end only amount to a few LEDs flashing on an FPGA board or whatever and it feels lame having all that work result is something that feels so mediocre; Mobile Robots doesn't feel that way. If anything, it feels like I'm doing something both educational and kind of exciting.

Calculations

```
//test cases for p component

        Kp = [0.1, 0.5, 1.0, 2.0, 2.5, 5.0]

// proportional control

        distance error = -desired distance - -front

        new speed = proportional gain * distance error

// saturation

        if new speed > max:

                new speed = max

        if new speed < - max:

                new speed = -max

// IMU degrees
        degrees = imu.getRollPitchYaw()[2]*180/math.pi + 180

// rounded distance sensors in inches
        sensorArray = [round(leftDistanceSensor.getValue()*39.3701, 5),
round(frontDistanceSensor.getValue()*39.3701, 5), round(rightDistanceSensor.getValue()*39.3701, 5)]

// control goal facing orientation based on position in image
        if image position[0] > 41:
                setSpeedsIPS(1, -1) #left spin
        if image position[0] < 39:
                setSpeedsIPS(-1, 1) #right spin
        if image position[0] == 39 or image position[0] == 40 or image position[0] == 41:
                setSpeedsIPS(0,0)

// PID control for 5-inch distance from goal
          front = getDistanceSensors()[1]
          proportional gain = 1.5
          desired distance = 5
          distance error = -desired distance - -front
          new speed = proportional gain * distance error
          setSpeedsIPS(new speed, new speed)

// PID wall follow logic
      error = getDistanceSensors()[0] - target Distance
      if(error<0):
        setSpeedsIPS(v,v - abs(error)*Kp_side)    #turn away from left wall
      else:
        setSpeedsIPS(v - abs(error)*Kp_side, v)   #turn towards left wall
```