

Comp2403 Computer Vision: Line Following

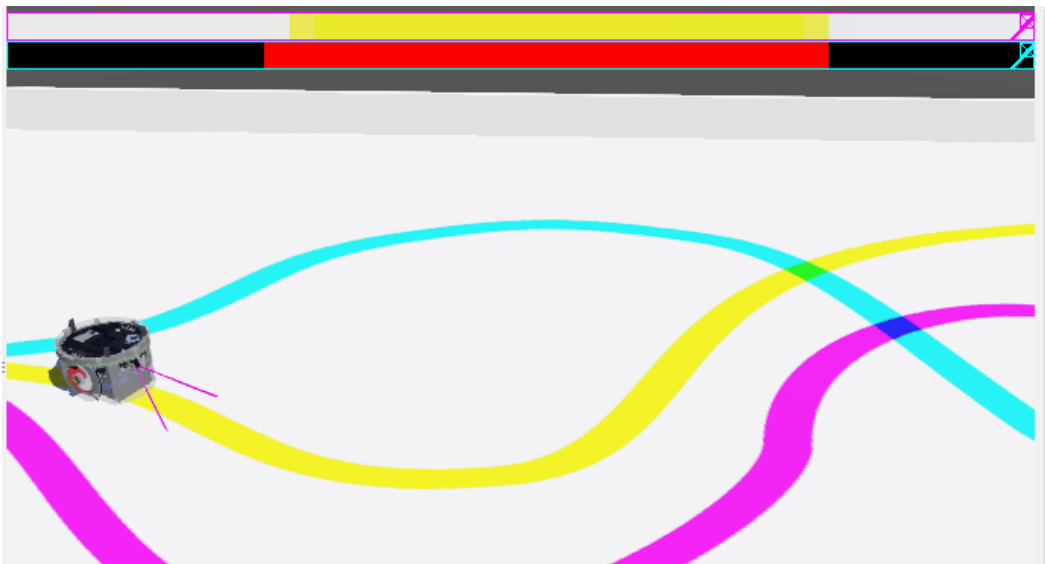
C.B.Price October 2020

| | |
|-------------------------|---|
| Purpose | To use color information from a linear camera to follow a line. |
| Files Required | Webots project folders on website. Use the world CBP_ePuck_LinearCamera_1.wbt Controller is CBP_ePuck_LinearCamera_1.c |
| ILO Contribution | 1,2 |
| Send to Me | If you are working online, please send movie-clip of your solution. |
| Homework | Look at CBP_Notes Chapter 4, sections 4.3 to 4.5 |

Activities

1 Coding the Controller

Let's start by getting the robot to follow the yellow line. So it's a good idea for it to start on the line like this



The camera has width 40 pixels and height 1 pixel, so is a “linear camera” and is pointing down as the camera frustrum shows above. The image seen by the camera is shown at the top of the screen, and the image segmented into yellow pixels is shown underneath, represented as red pixels.

The code gets the RGB **image** from the camera, then segments it into **lineImage** when the pixel color has yellow hue. It then gets the centre location **loc** of the yellow pixels.

(a) Decide how to compute a value for the variable **error**, how far the camera image is off the line centre. This error value should have a meaningful range e.g., 0.0 – 1.0 or something thereabouts.

(b) Decide how to use the **error** signal to produce **omegaL** and **omegaR**. Remember the robot should move forward when the error is zero. I've given you a variable **gain** which you can use to multiply the error.

2 Possible Investigations – just for fun!

- (a) Try to get the robot following the other lines
- (b) Modify the controller so the robot can jump across intersecting colored lines
- (c) Perhaps create a new function to return the color (in degrees) of the majority of pixels seen by the camera. You could create a floor with different colored blobs to make the robot follow a trajectory. The function will have the form

```
double getColor(const unsigned char *image,unsigned char *imageThresh, int width, int height) {  
    // This would start off like imageSegment and you would need to store an array of numbers  
    // of hue values (index 0 – 359) then find the maximum in this number array and return the  
    // associated hue modeHue  
  
    return modeHue;  
}
```

- (d) What would the robot do if it started on a floor containing a color wheel? You would need to chose its starting location carefully.
-