Applied project: Worksheet Vision

These worksheets are designed to provide you with some help
in designing your controller robotic controller to develop a Vision system for your AI

Prerequisite

1) Complete all worksheets today
2) Ensure you have a camera added to the robot.
3) Enable the camera with the following lines (you need to make sure you put them in the correct place)

```
static WbDeviceTag camera;

camera = wb_robot_get_device("camera");
```

Aim for Today
Build a simple colour detection algorithm, and use this algorithm to inform motor control.

Recall the two resources in the environment are of a certain colour and size.

As our environment is relatively sterile, we could find these resources by their RGB ratio which would indicate the presence and distance of a resources. We however cannot just rely on a single RGB value i.e., just the value of red.

Why?

The RGB value of red is 255, 0, 0
The RGB value of purple is 255, 51, 204

So, if we just used R we would find both red and purple objects. Thus, we need to set a ration and range for the values we need

For example

Upper range 255,20,20

Lower range 240,0,0

You can use tools such as
https://www.w3schools.com/colors/colors_picker.asp
to look at the RGB makeup of various colours

Task 1) Make the robot go forward when a red object is seen

We can get the RGB values with the following lines of code

Firstly, we make a variable to store the current frame of the img

const unsigned char *image = wb_camera_get_image(camera);

We can then use the following functions to get the RGB values for XY coordinates in the image

int r = wb_camera_image_get_red(image, image_width, x, y);

int g = wb_camera_image_get_green(image, image_width, x, y);

int b = wb_camera_image_get_blue(image, image_width, x, y);

For the time being set the XY coordinates to be the centre point of your camera. You can find with Width and height of your camera by expanding the child node.

pseudo code example
If Red saturation > lower && Green and Blue <Upper

Move forward

We could also consider using a bit of sensor fusion to use multiple sensors to aid the function for example. We could set our IT sensors to stop the robot if it gets to close the red object

If Red saturation > lower && Green and Blue <Upper

If IR< x

Move forward.

How can we determine the correct min value for red, and max value for blue and green? Simple, trial and error, print out the RGB values and try moving the robot around or moving objects in front of the robot.

Task 2) Update the Vision algorithm to detect the colour RGB values for the entire image

Using two for loops you can go through every pixel in images as such

```
for (int x = 0; x < image_width; x++){
for (int y = 0; y < image_height; y++) {
int r = wb_camera_image_get_red(image, image_width, x, y);
int g = wb_camera_image_get_green(image, image_width, x, y);
int b = wb_camera_image_get_blue(image, image_width, x, y);}}
```

the above will start at the top left pixel, search all pixel in that column before moving to the next row i.e.,

x		

x		
x		

x		
x		
x		

x	x	
x		
x		

This will give us the output for every single pixel (a lot of prints).

You now need to change this to give the sum of Red Green Blue

Next how could we normalise this value, changing RBG saturation to a scale say between 0-1 or within the normal 0-255?

Task 3) Advanced: Divide the camera into three windows

Currently our algorithm will move forward the robot forward when red reaches a certain threshold.

However what if the object is to the left, or right side of the camera?

We could divide the camera into segments, and calculate the saturation for each segment

Window 1 window 2 window 3

We can define 3 segments easily with the following

w = width of the camera in pixels
Window1 = w/3
Window2 =(w/3)*2 (or w*.66)
Window3 = w

We can then add an if statement to our for loop, to allow us to assign the RBG values to different variables depending upon if we are currently searching a column in window 1, 2 or 3.

Using this logic, it is now possible to determine where abouts a red object is in the robot's field of view i.e., if its in window 1, its to the left of the robot, so we would ideal want the robot to turn to the left.

Note a better way of doing this would be to use contours and moments to determine the precises size, shape and centre point of a object. Feel free to explore this in your own time!