Assignment Brief	
Title:	Computer Vision Assignment 1
Submission Deadline:	15th February 2024, 16:00
Submission:	Online (DLE)
Contribution to Module Grade:	40%
Individual/Group Assignment:	Individual
Module:	ROCO321
Module Leader:	Dr Dena Bazazian
Lab Technician:	James Rogers

Overview

These labs involve using the owl robot that can be booked out from stores. Please **work individually** with the robot and **use the same owl each week**. **Submitted reports are individual pieces of work.**

Marking

Assignment 1 is worth 40 marks (40% of the module).

Tasks 1 and 2 are worth 15 marks each and are broken down as such:

- **Solution (4 marks):**
  - **(0-1)**: Code barely fulfils the task specification.
  - **(2-3)**: Code fulfils the task specification.
  - **(4)**: Code fulfils task specification and extra features have been added that improves the program.

- **Method (4 marks):**
  - **(0-1)**: Little to no effort explaining the code or background on theory.
  - **(2-3)**: Some theory, and an explanation of the code.
  - **(4)**: A comprehensive background in relevant theory, and a detailed explanation of the code with appropriate justifications.

- **Evaluation (4 marks):**
  - **(0-1)**: Little to no effort assessing the performance of your solution.
  - **(2-3)**: A measure of performance is identified (such as: error rate, false positives / negatives, etc.) and data is collected to assess your solution.
  - **(4)**: A measure of performance is well explained and justified, and a significant amount of data is collected to assess your solution. This may involve making your own data sets to evaluate your solution in a broader scope.

- **Conclusion (3 marks):** Based on your evaluation (with data and numbers to back it up), what conclusions can you make from your solution, and what would you improve on?

By James Rogers (james.rogers@plymouth.ac.uk)

An **extra 10** marks are awarded for including the following in the report:

- **Videos (2 marks):** Must contain links to YouTube videos of your code running on the owl. The videos don't need to be complicated, just a short clip demonstrating your code functioning as the task describes.
- **Flow Diagrams and Code Snippets (3 marks):** Flow diagrams and snippets will be needed to explain your code, marked on how clear and informative they are.
- **References (3 marks):** All quotes, information, and diagrams used in the report should be clearly cited.
- **Acceptable English and Grammar (2 marks):** Not marked strictly but do proofread before submitting.

Task 1: Colour Classifier (15 marks)

Images are broken down into pixels, which each have red, green, and blue values. Your task is to identify 5 different colours using these RGB values.

Tasks:

- Open the Task1.pro file in the AINT308Part1 folder for the task template.
- Modify the code to identify one of five colours based on the RGB values.
- Display the detected colour to the screen, with text written in that colour.

Notes

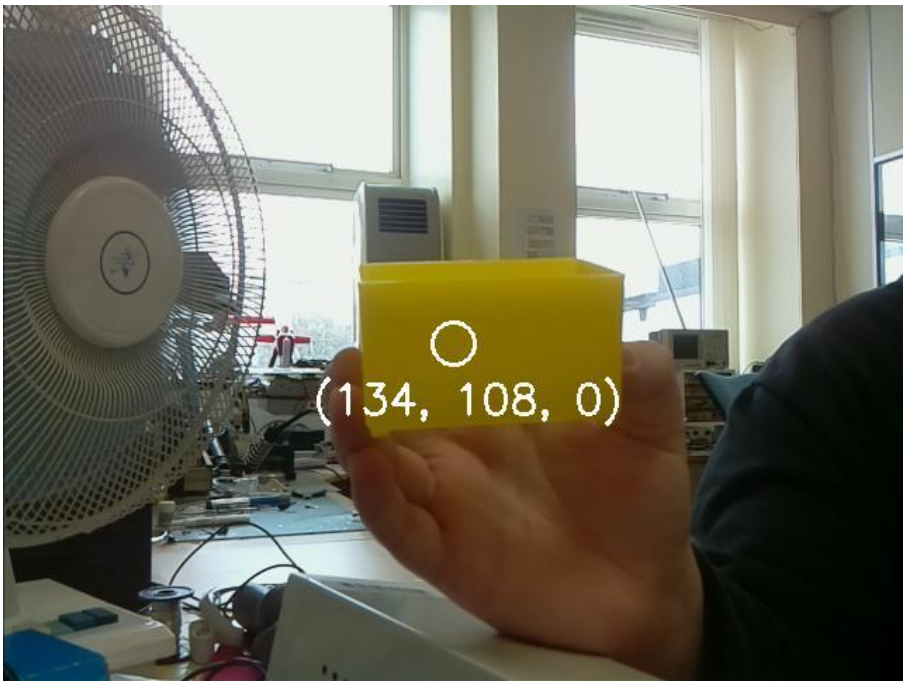
When opening the project file, you'll be prompted to select a compiler. **Make sure to select MinGW.**

In the code given to you, there is a class called "robotOwl". This handles communication between the robot and the PC, allowing you to more easily control the motors and read images from the cameras. A function in the owl class called "getCameraFrames()" is used to capture the left and right eye images (though in this case only the left eye is used). Images are stored in a datatype called "Mat" in openCV.

After capturing the images, the centre most pixel's RGB values are extracted from the Mat and stored in three unsigned int's. Using these values, drawing functions display them on the original image for you to see. Then "imshow()" and "waitKey()" are used to display the image. **More information on these functions and datatypes can be found in the OpenCV Basics document.**

A pixel consists of red, green, and blue values, which each range from 0-255. With these three values, a wide variety of colours can be mixed. For example, (0,0,0) will be black, and (255,255,0) will be yellow. **The core of this task is how do you check if a pixel looks red, white, etc based on these three values.**

By James Rogers (james.rogers@plymouth.ac.uk)



Above is the RGB values for a yellow box, as you can see the red and green values are high, whereas the blue value is low. These will change a little based on the lighting and shade of colour, so how would you design ranges around this?

Task 2: Colour Tracker (15 marks)

In this task you will need to identify a target colour in the frame (using HSV), locate its centre of mass, and use the owl's servos to track it.

Tasks:

- Open the Task2.pro file in the AINT308Part1 folder for the task template.
- Use the cvtColor() function to convert the frame to the HSV colour space.
- Use the inRange() function to identify all the pixels that are the target colour.
- Use OpenCV Moments to find the centre of mass of the colour target.
- Using OpenCV drawing functions, draw some marker on the image to show the position of the target's centre of mass.
- Move the servo's so that at least one eye tracks the target.

Notes

The code given in the task template will set up the owl class same as before, but this time it will enable the motors. **Just make sure to run "./OWLsocket" each time you run your code.**

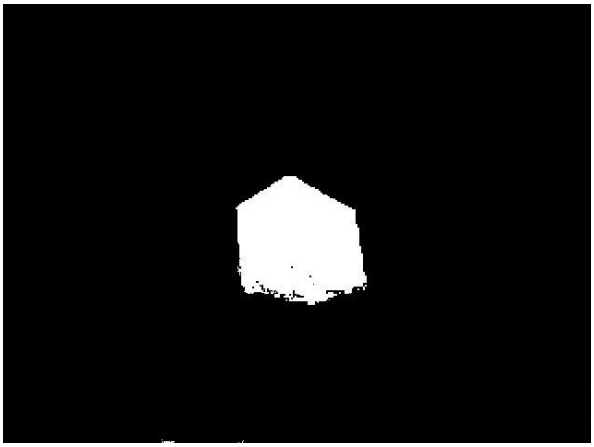
Colour tracking is done in the HSV colour space, as a targets hue changes very little under different lighting conditions. **Convert the frame from BGR to HSV with the cvtColor() function** (a new Mat "FrameHSV" is needed to store the output image).

Now apply a threshold to FrameHSV so that only the pixels on the colour target will be highlighted. This is done with the inRange() function. You can pick any target you like, **but make sure it is brightly coloured.**

By James Rogers (james.rogers@plymouth.ac.uk)

```
Mat FrameFiltered;
Vec3b LowerBound(HueLower, SatLower, ValLower);
Vec3b UpperBound(HueUpper, SatUpper, ValUpper);
inRange(FrameHSV, LowerBound, UpperBound, FrameFiltered);
```

Choose your upper and lower bounds carefully to filter out all hues that are not part of your chosen target. Hint, the upper and lower bounds around Hue will be a lot stricter than saturation or value. You can check how well your thresholding is working by displaying FrameFiltered with imshow():



Now find the centre of the white pixels with OpenCV Moments. The code below converts the moments output into a point representing the centre of mass. Note, if m.m00 is equal to zero, your program will crash. It's up to you how you handle this edge case.

```
Moments m = moments(FrameFiltered, true);
Point p(m.m10/m.m00, m.m01/m.m00);
```

Use drawing functions such as line() or circle() to **draw a marker on the Frame** to show the tracking in action.

By James Rogers (james.rogers@plymouth.ac.uk)

With the position of your target known, use the owl servo commands to try and move the target to the centre of the screen. The functions available to you are:

setServoAbsolutePositions(Right Eye X, Right Eye Left, Left Eye X, Left Eye Y, Neck): This sets the servos to given positions. (0,0,0,0,0) will being looking straight forward, but try different values to see how it works.

setServoRelativePositions(Right Eye X, Right Eye Left, Left Eye X, Left Eye Y, Neck): This moves servos a given amount from where its already looking. (0,0,0,0,0) will result in no movement, (100,0,0,0,-20) will move the right eye 100 units right and the neck 20 units left.

setServoRawPositions(Right Eye X, Right Eye Left, Left Eye X, Left Eye Y, Neck): This is less useful to you but included for completeness. This sends raw PWM values to the servos, similar to the absolute movement function, but (0,0,0,0,0) will be the extreme extension to the left for all servos.

Its up to you how you use these functions to track the target. But at least one eye needs to reliably follow the colour target.

Deadline and Submission

Deadline on **15th February 2024, 16:00.** The deadline information for submission is on the ROCO321 DLE page via the submission link.

Plagiarism

This is an individual assignment and must reflect the work of that individual. Thus, while you may discuss this assignment in general with your colleagues and give each other technical help, your scientific literature review must be entirely your own work.

The University treats plagiarism very seriously. If you cannot satisfy me that your work is your own, formal plagiarism procedures will be started.

The penalty for submitting work which is wholly or partially the work of someone else is usually, at least, a mark of zero for the assignment. Do not be tempted to help a colleague by giving them your work, as both parties will be guilty of an assessment offense, and both face the risk of a zero mark. Please refer to your student handbook for guidance as to what constitutes original / individual work.

Module Learning Outcomes Assessed

- ALO-1: Demonstrate a systematic understanding of the underpinning vision theories and techniques applicable in object recognition.
- ALO-2: Critically evaluate current research and methods in artificial vision.
- ALO-3: Design and implement an original practical vision system.