

University of Plymouth

Owl Servo Calibration Guide

James Rogers

james.rogers@plymouth.ac.uk

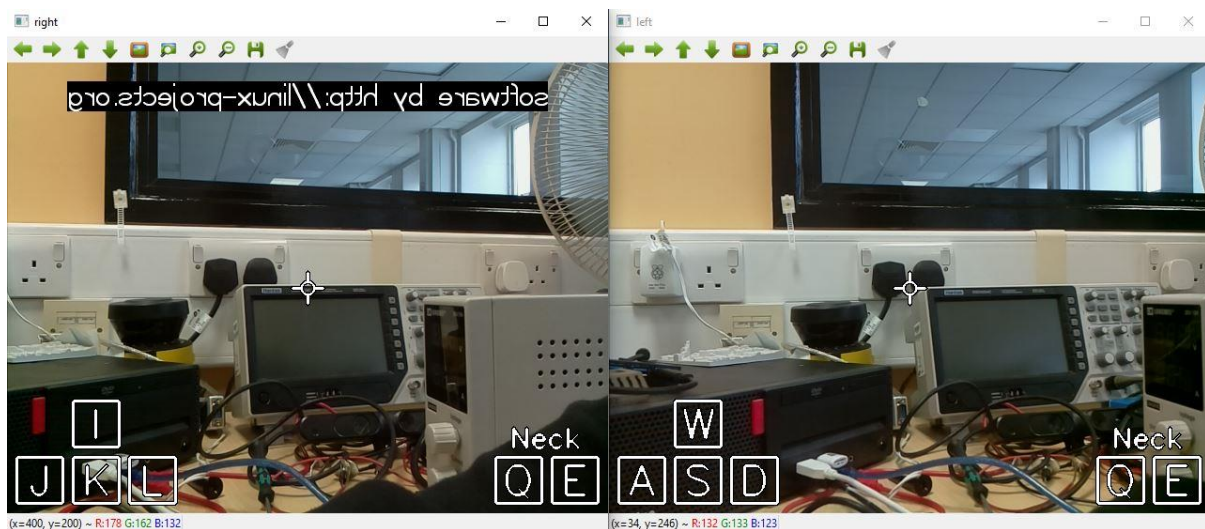


Introduction

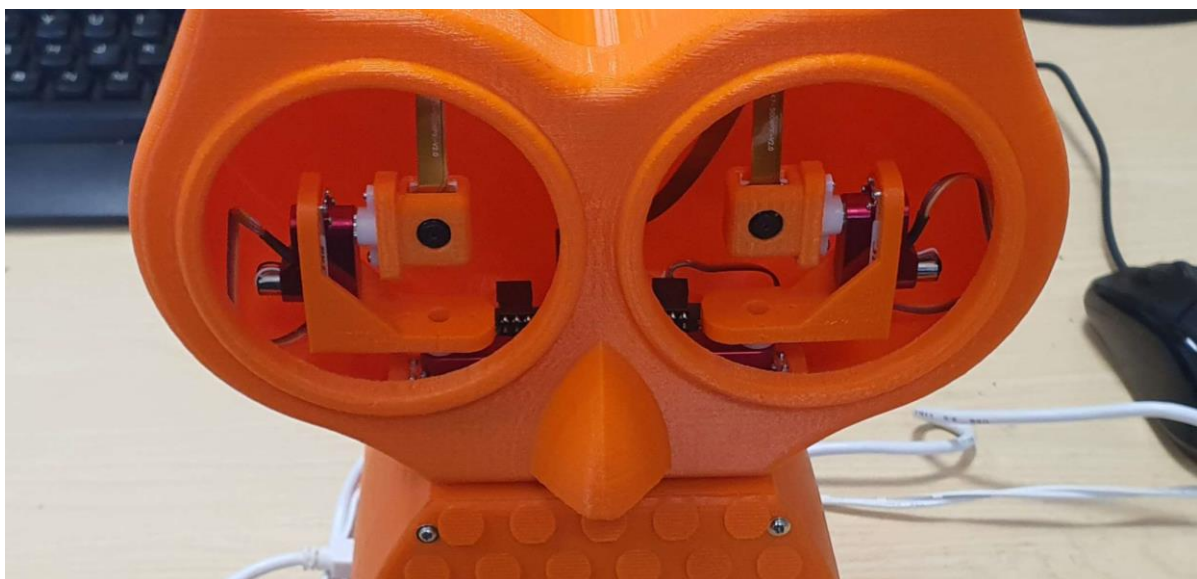
Why is this needed? For doing stereo computer vision algorithms like disparity mapping, the two cameras are assumed to be hard mounted with respect to each other. However, the owl robot's cameras are individually actuated. So the servos need to be adjusted until they are perfectly aligned. This will change from robot to robot so **make sure to book out the same owl every week**. In the example projects given to you, one is called "Owl Servo Calibration". This document outlines how to use this program to align the servos.

The Calibration Process

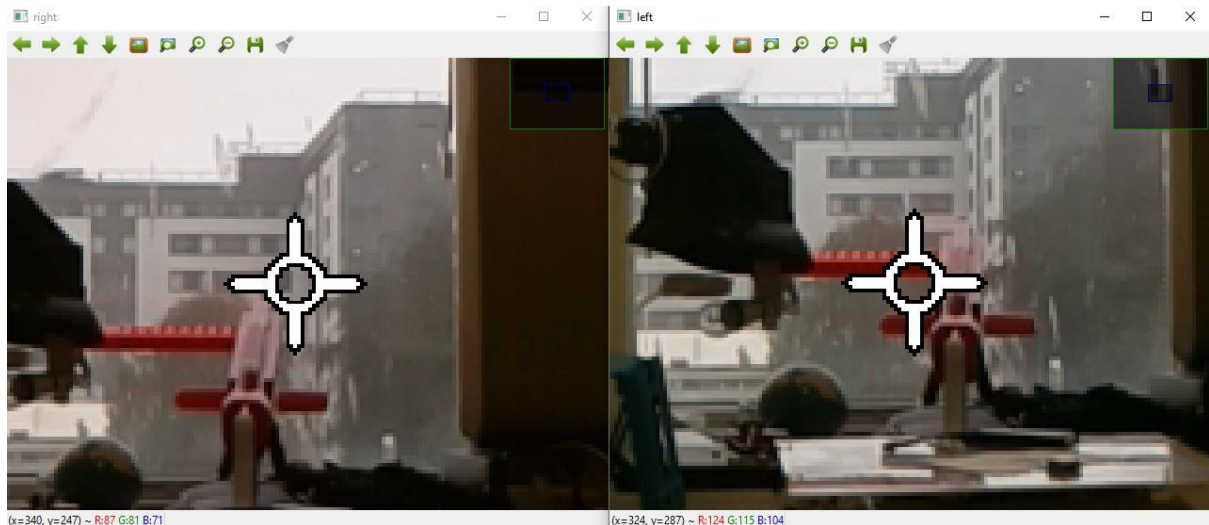
Open the calibration program and use putty to start the owl socket (refer to the owl set up guide for how to do this). You'll be met with the UI below. Each camera has controls for moving them independently, as well as neck controls. Press or hold each key at a time to see how it controls the motors.



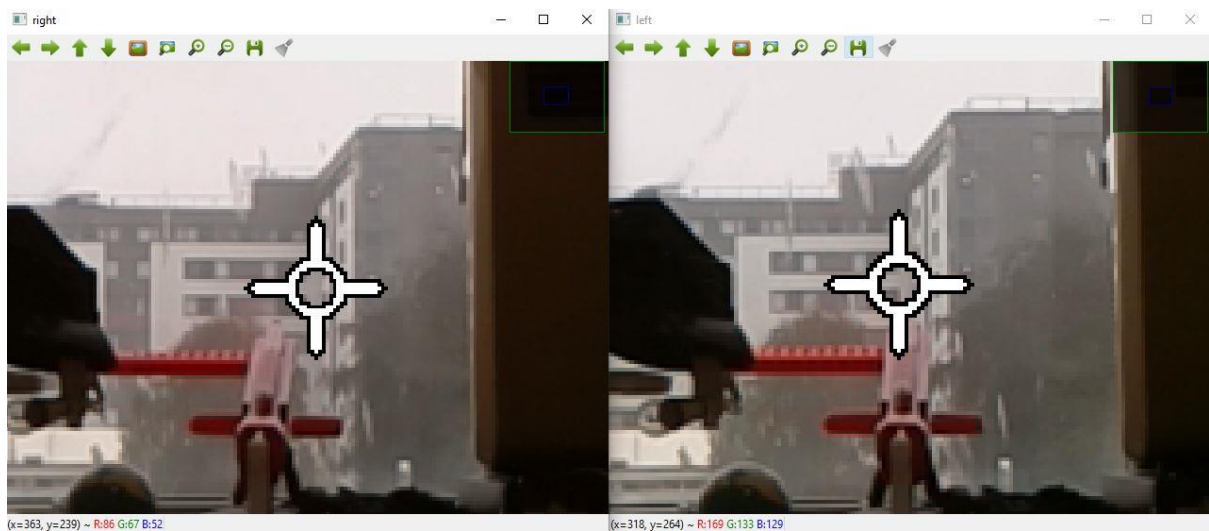
Move each motor to make the eyes look straight forward, and have the head in line with the body. Do this as best you can, so it resembles the image below.



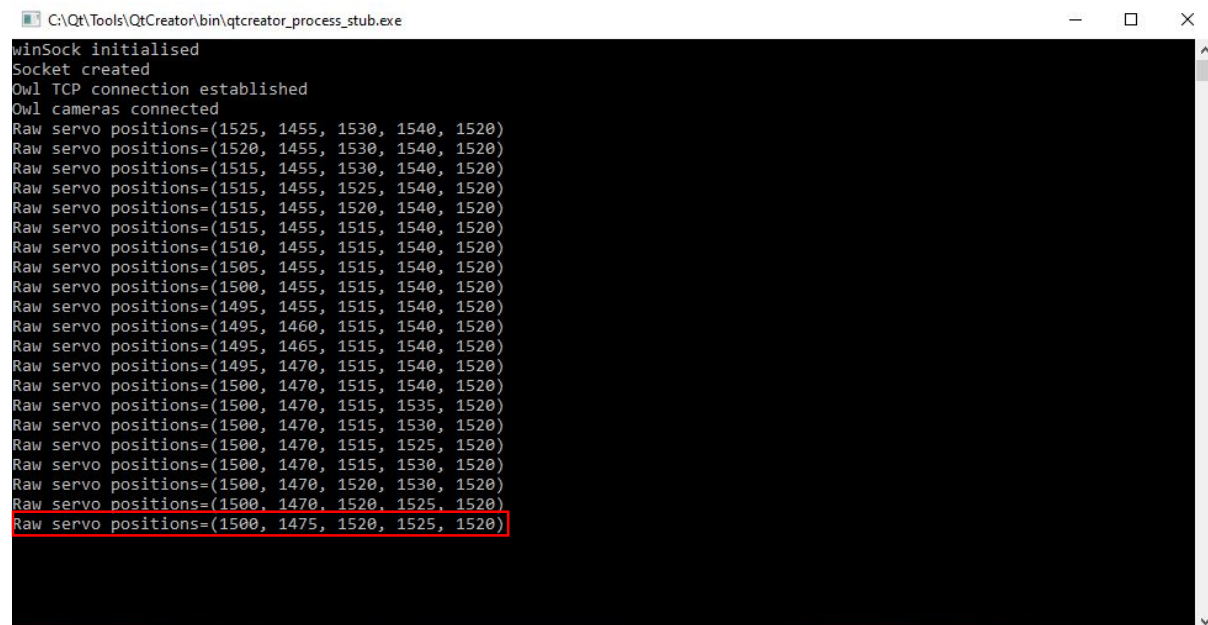
Next, move the owl robot to aim at some distant target. The best way to do this is to aim it out the window. Calibrating with a distant target will put the eyes into a parallel configuration, instead of converging. As can be seen below, the cameras are looking roughly in the same direction, but not exactly.



Now pick one of the cameras, and try to align it to match with the other. You can scroll to zoom in, but be careful to only move one of the cameras in this step. Due to the accuracy of the servos, you won't get it exact. But get as close as you can. The image below shows that both cameras are aimed at the same spot.

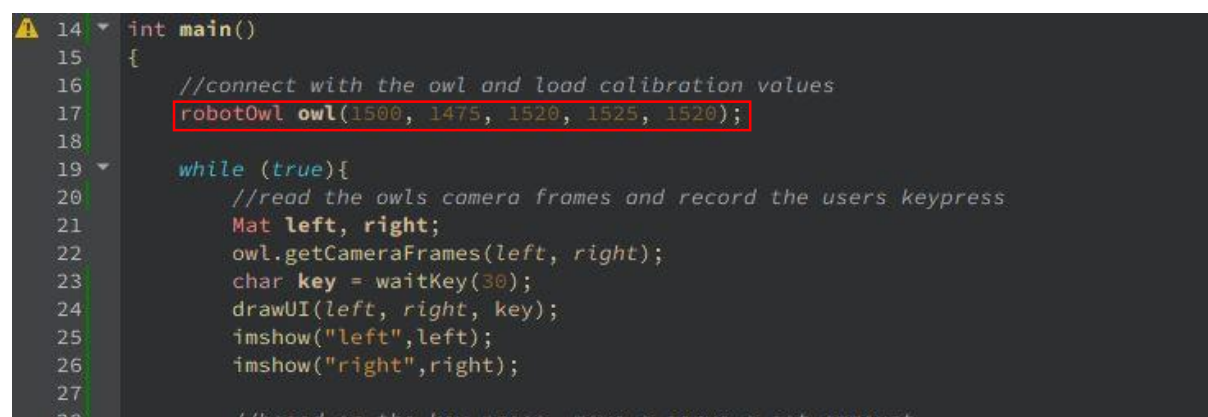


Throughout this process, the console has been printing the servo positions to the terminal. After the last step, the last set of values printed to the terminal will be your calibration values. These will act as the origin point for your servos.

A screenshot of a Qt Creator console window. The title bar shows the path 'C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe'. The console output shows a series of 'Raw servo positions' followed by five numbers in parentheses. The last line, 'Raw servo positions=(1500, 1475, 1520, 1525, 1520)', is highlighted with a red rectangle. The console window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
winSock initialised
Socket created
Owl TCP connection established
Owl cameras connected
Raw servo positions=(1525, 1455, 1530, 1540, 1520)
Raw servo positions=(1520, 1455, 1530, 1540, 1520)
Raw servo positions=(1515, 1455, 1530, 1540, 1520)
Raw servo positions=(1515, 1455, 1525, 1540, 1520)
Raw servo positions=(1515, 1455, 1520, 1540, 1520)
Raw servo positions=(1515, 1455, 1515, 1540, 1520)
Raw servo positions=(1510, 1455, 1515, 1540, 1520)
Raw servo positions=(1505, 1455, 1515, 1540, 1520)
Raw servo positions=(1500, 1455, 1515, 1540, 1520)
Raw servo positions=(1495, 1455, 1515, 1540, 1520)
Raw servo positions=(1495, 1460, 1515, 1540, 1520)
Raw servo positions=(1495, 1465, 1515, 1540, 1520)
Raw servo positions=(1495, 1470, 1515, 1540, 1520)
Raw servo positions=(1500, 1470, 1515, 1540, 1520)
Raw servo positions=(1500, 1470, 1515, 1535, 1520)
Raw servo positions=(1500, 1470, 1515, 1530, 1520)
Raw servo positions=(1500, 1470, 1515, 1525, 1520)
Raw servo positions=(1500, 1470, 1515, 1530, 1520)
Raw servo positions=(1500, 1470, 1520, 1530, 1520)
Raw servo positions=(1500, 1470, 1520, 1525, 1520)
Raw servo positions=(1500, 1475, 1520, 1525, 1520)
```

Make a note of these values, as all future tasks with the owls will need them. Whenever you initialise an instance of the “robotOwl” class, copy these calibration values in. This will keep the cameras aligned when using the servo move commands.

A screenshot of a C++ code editor. The code is inside a 'main' function. Line 17 contains the line 'robotOwl owl(1500, 1475, 1520, 1525, 1520);', which is highlighted with a red rectangle. The code includes comments and a while loop for camera frame processing.

```
14 int main()
15 {
16     //connect with the owl and load calibration values
17     robotOwl owl(1500, 1475, 1520, 1525, 1520);
18
19     while (true){
20         //read the owls camera frames and record the users keypress
21         Mat left, right;
22         owl.getCameraFrames(left, right);
23         char key = waitKey(30);
24         drawUI(left, right, key);
25         imshow("left",left);
26         imshow("right",right);
27
28         //based on the key press, move a servo a set amount
```