Comp2403 Object Localization using Vision

C.B.Price October 2021

Purpose	(i) To rotate to an object using vision (ii) To locate an object using Vision.
Files Required	**Pixymon** exe, available on the webpages.
ILO Contribution	LOs 1,2,3
Send to Me	nix
Homework	Read chapters 4 & 5

Activities

Protocol:
(1) Treat the kit like eggs (not hard-boiled)
(2) When you upload a sketch to the Arduino, make sure the power switch is off (position 0).
(3) When starting a sketch, hold down reset then turn the power on (switch position 2)
(4) Never drive the turret with the Pixymon cable attached.
(5) Stay awake and focussed.

---

1 Rotating to an object in the Field of View

The goal is to rotate the robot to face a red object at the centre of the cameras field of view (FOV). Below is a diagram which shows the situation.

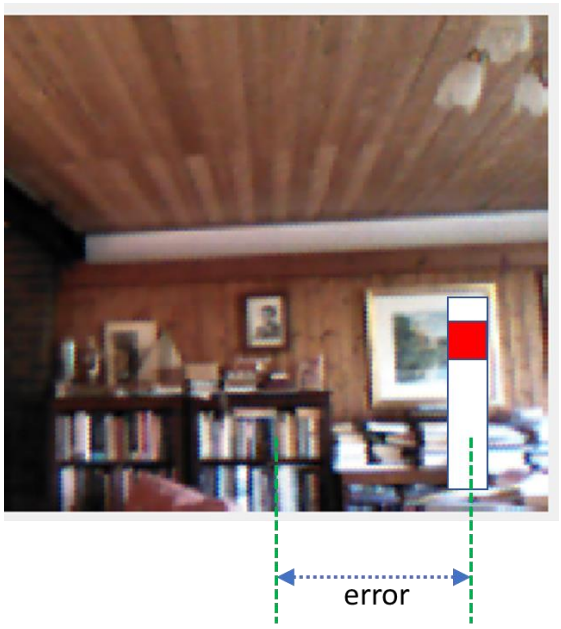
The object should be in the centre of the frame; the error from this position is shown.

Open up the sketch **CBP_2403_R_MoveToBlock** and look over the code. Mist will be familiar.

You have two declared variables **x** which is assigned to the x-location of the centre of the segmented block, and **frameWidth** which is the width of the image.

(a) Use these to code an expression for the **error**

(b) Make sure you *normalize* the error value into a range -1.0 to 1.0 (or thereabouts)



(b) Decide how to use the **error** to provide **driveL** and **driveR** to the motors. Please consider using the proportional gain **Kp**, and don't forget to give this a value.

(c) Investigate! Move the object around by hand, and make sure the robot tracks the object as it moves.

(d) Perhaps increase **Kp** until the system becomes unstable, and then add some **Kd** derivative control

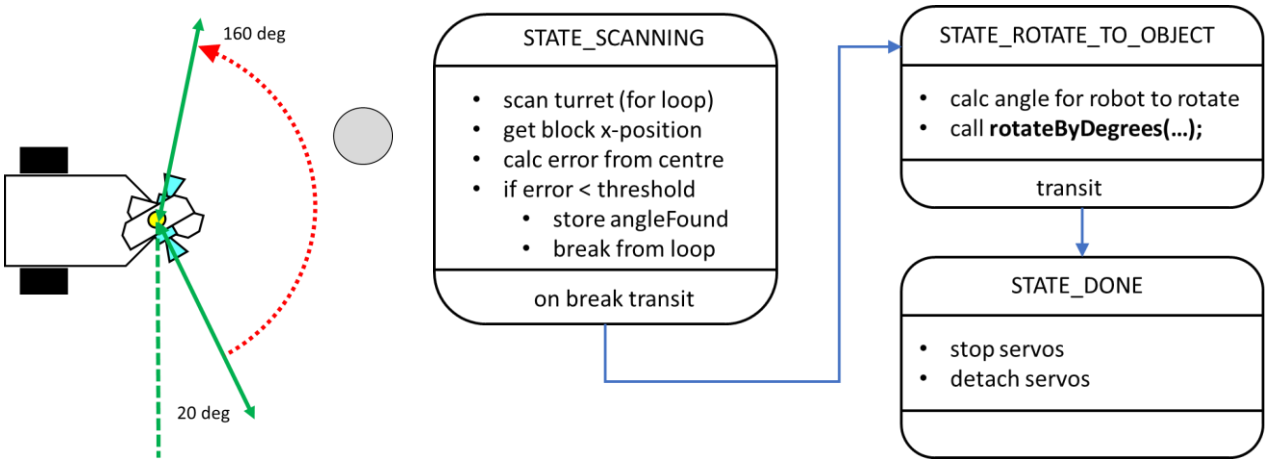
---

(e) Add some **fwd** drive to get the robot to move towards the block, and to track it.

(f) You might consider putting a small red object on the back of a second robot and try to follow it using vision.

2 Locating the Object

Here we shall use the turret servo to scan the camera over a range of angles from 20 degrees to 160 degrees, greater than the camera FOV. You've seen this before when we used the Ultrasonic distance sensor, the arrangement is shown below on the left. We shall code using a good old FSM, shown below right. The FSM template is provided. The sketch is **CBP_2403_R_LocateBlock.**



You will need to draw on your previous sketch(es). First work on **STATE_SCANNING**

(a) Add a line of code to get the x-coordinate of the object in the image
(b) Add a line of code to find the error (to the image centre)
(c) Add a line of code to normalize the error (to a range -1.0 to +1.0)
(d) Add an if-statement to test if the error is less than the threshold **errorThresh**. Take care since the error can be both positive and negative. (DO NOT use another if-statement to resolve this). The body of the if-statement should record the **angleFound** and also **break** out of the for-loop.

Now complete the **STATE_ROTATE_TO_OBJECT** (you have been here before)

(e) Add code to compute **rotAngle** which you will then pass as a parameter to the function **rotateByDegrees(rotAngle);**

All done. Now see what happens.

(f) You may like to add some code (in one or other state) to get the width of the object detected, and to use your function **getDistanceFromObject(uint16_t _width)** to get the distance to the object and then to move exactly to the object.