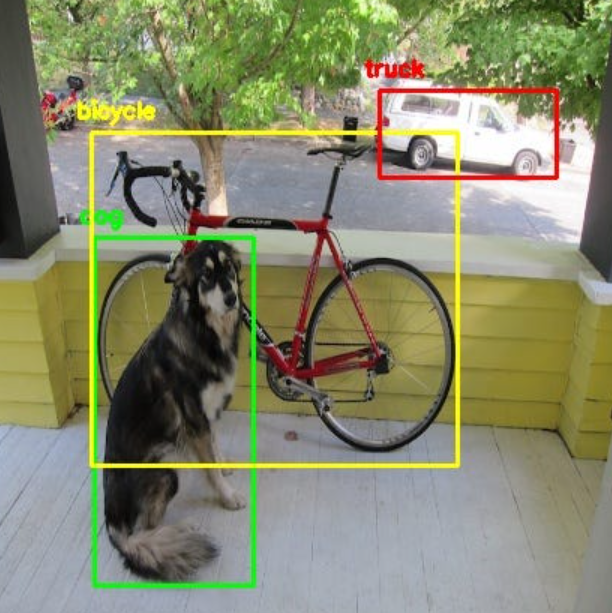
**Object Detection in OpenCV**

As the name suggests, we're concerned with detecting different elements in an image. Object detection involves image classification and enclosing it in a bounding box. Image classification deals with identifying the object as a class the computer was trained to detect, and that object is enclosed within a boundary, known as the bounding box.



There are plenty of Machine Learning/Deep Learning models for this, such as YOLO, SSD, HOG etc. However, we are going to discuss about Contour detection here.

Let us describe it as a step-by-step process.

First, we need to do some pre-processing on the image, such as blurring it. We have used Gaussian blur for this.

Then, two paths arise- one being thresholding, and the other, Canny edge detection. Thresholding is done for images with much less details than the ones we use Canny edge detection in.

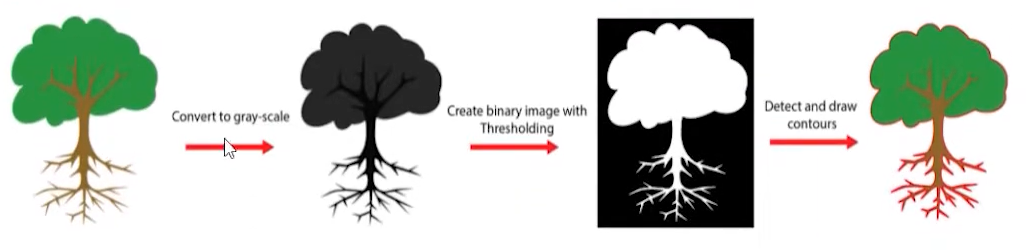
If we are going for the thresholding approach, then we need to make sure that our object is white, and the background, black.

If we are going for the Canny edge approach, then we don’t need to worry much about it.

Of course, we need to select suitable thresholds for each of the two methods, such that we get an accurate image.

Lastly, we detect the contours based on the results of the previous step. If they were accurate enough, we’d be able to see the contours by drawing them on the original image (or a copy of it).

Here’s a summary of the entire process via the thresholding method.



This is what Canny edge detection gives, for more complex images.





For our coin detection task, we can think of a much simpler approach to it, than contour detection. It involves Hough transforms.

We can use the in-built functions in OpenCV to detect circles in our image. This is done by tweaking a few parameters, of course.

The entire information of a circle in our image plane can be characterised by three variables. The x and y coordinates of the center, and the radius. Once we know these three numbers, we can point out the circle in our image plane.

Therefore, we can find these three numbers for each “potential” circle on the image plane (again, “actual” circles can be distinguished from the “fake” circles, once we have done our pre-processing part correctly). Then, we can simply draw these circles on our original image (or a copy of it).



The cross-section of a can is a circle, after all.