# Caltech image bounding boxes

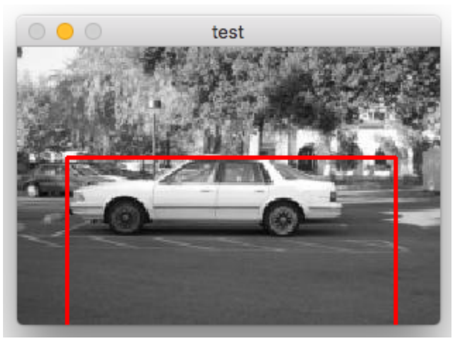
**mmilano** **Jul 31**

I wanted to visualize the bounding boxes on the .mat files, but I was expecting the car to be more closely bound.

I'm using this to grab the box data:

```
(y, h, x, w) = io.loadmat(p)["box_coord"][0]
```

Here is what it looks like when I overlay annotation_001.mat on top of image_0001.jpg from the car_side dataset:



Is that bounding box correct? or am I retrieving the data wrong?

One thing to note is that the box was way off when I was using the raw output from io.loadmat in cv2.rectangle(), so my final result in the picture casts y, h, x, w to int before using them in cv2.rectangle().

The code looks something like this:

```
(y, h, x, w) = io.loadmat(p)["box_coord"][0]
img = cv2.imread(imagePath)
y = int(y)
h = int(h)
x = int(x)
w = int(w)
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 1)
cv2.imshow("test", img)
```

Jul 31

**1 / 5**
Aug 1

Aug 1

**ThamNgapWei** **Jul 31**

Have you tried another images?If many or all of them have the same issues, that means your codes have some flaws, if only a few of images(one~ten) have this kind of issues, that means part of the bounding boxes are inaccurate.

**mcduffee** **Jul 31**

I could be wrong*, but I think the bounding box coordinates are stored as pairs of (x,y) coordinates, not an initial (x,y) coordinate and a (w,h) tuple. Have you tried displaying your rectangle as (x, y), (w, h) instead of (x, y), (x+w, y+h)?

- I don't think I'm wrong. Inspired by your example, I managed to cobble together a bounding box display program from bits and pieces of Adrian's example code. The bounding boxes I display over the car images look pretty tight.

I apologize if this code is doing something inelegant, I'm still learning...

```
for p in glob.glob(conf["image_dataset"] + "/*.jpg"):
        image = cv2.imread(p)
        # extract the image ID from the image path and load the annotations file
        imageID = p[p.rfind("/") + 1:].split("_")[1]
        imageID = imageID.replace(".jpg", "")
        p = "{}/annotation_{}.mat".format(conf["image_annotations"], imageID)
        annotations = io.loadmat(p)["box_coord"]

        # loop over annotations (as if there might be more than one) and display
        [cv2.rectangle(image, (x, y), (w, h), (0, 255, 0), 1)
                        for (y, h, x, w) in annotations]
        cv2.imshow("test", image)
        cv2.waitKey(0)
```

**mmilano** **Aug 1**

**@mcduffee** ... that was it. Thinking of the values as width and height instead of just another set of coords threw me off.

Thanks!

**Adrian** Chief PyImageSearcher **Aug 1**

Normally bounding boxes are represented as a list of: starting x-coordinate, starting y-coordinate, width, and height.

However, for whatever reason, the CALTECH dataset stores coordinates as y, h, x, w. I'm not sure why. I personally find it pretty confusing.