

Object detection: own data set

nicocarosio

Jan 14

1 - Yes

2 - python

3 - I work with Ubuntu 1404 LTS , but I have a virtual pc with windows so .exe **it will be ok for me!**

4 I work wit Ubuntu 14.04 LTS

5- If it is easier, yes! the thing is that I am learning to code also and to adapt some code to my example take me a lot of effort! (I am ok with that, just to let you know)

nicocarosio

Jan 14

Know I am reading this example...

http://dlib.net/train_object_detector.py.html

ThamNgapWei

Jan 14

Since you prefer to work with python, why do you need .mat file? I though it is a format design for matlab?

If you want to train your own object detector by python, I think you do not need to transfer your interesting regions into .mat file at all. In that way, I would prefer solution 5(by python) to

- 1 : parse the information of the xml
- 2 : Crop the regions from the images
- 3 : Save those regions into a list


After that it is up to you to process the images, transfer it to gray scale, normalize and so on.

If you want to train your object detector by dlib, you do not need to write anything to parse the xml, just copy the codes of the example

Please correct me if I am wrong, not too familiar with python or matlab.

nicocarosio

Jan 14

Nothing to correct!, I just wanted to use the .mat file following the example of the lesson.
Now I am trying to use this example: http://dlib.net/train_object_detector.py.html to train it.-
I will try with this and if I have a question, let you know!
thanks!!! 

nicocarosio

Jan 14

I am having an error:

RuntimeError: Unable to open file: tmp/images/01.jpg 😞

but I think it is related to the SO or something like that... I still working on it!

ThamNgapWei

Jan 15

I guess it is because the program cannot find the location of the file, or your dlib cannot read jpg file, in c++, I have to add DLIB_JPEG_SUPPORT to tell the dlib I want it to support jpg, maybe python also need to do something like that.

nicocarosio

Jan 15

It's work!!!    !!

Now cool down, read it again and improve it!! thanks!!! 😊

ThamNgapWei

Jan 15

Congratulation, may I know what was the problem?

The **python script** which crop the interesting regions(I assumed the xml file is generated by imglab). If there are any single line of codes are unclear to you, feel free to ask 😊, have a nice week

ps : This script do not consider performance issues and doing any error handle

nicocarosio

Jan 15

Thanks

Yes, It is simple root mystake need to solve by code, meanwhile hard solution:

When I run `./imglab mydataset.xml`

generate the file with the root **tmp/images**

Then I label the objects i want to be detected and so on...

Here everything is ok...

Finally the example ask me to send the root where are the image (in this folder it suppose to be also the **mydataset.xml file**) so I copy and paste the file there.

But when I run this:

```
python dlib_train_detector.py tmp/images/
```

the program use tmp/images to read the image and xml file.

But Now the xml file because it is in the same place of image the root inside is wrong.

So a copy again the file tree inside the file.

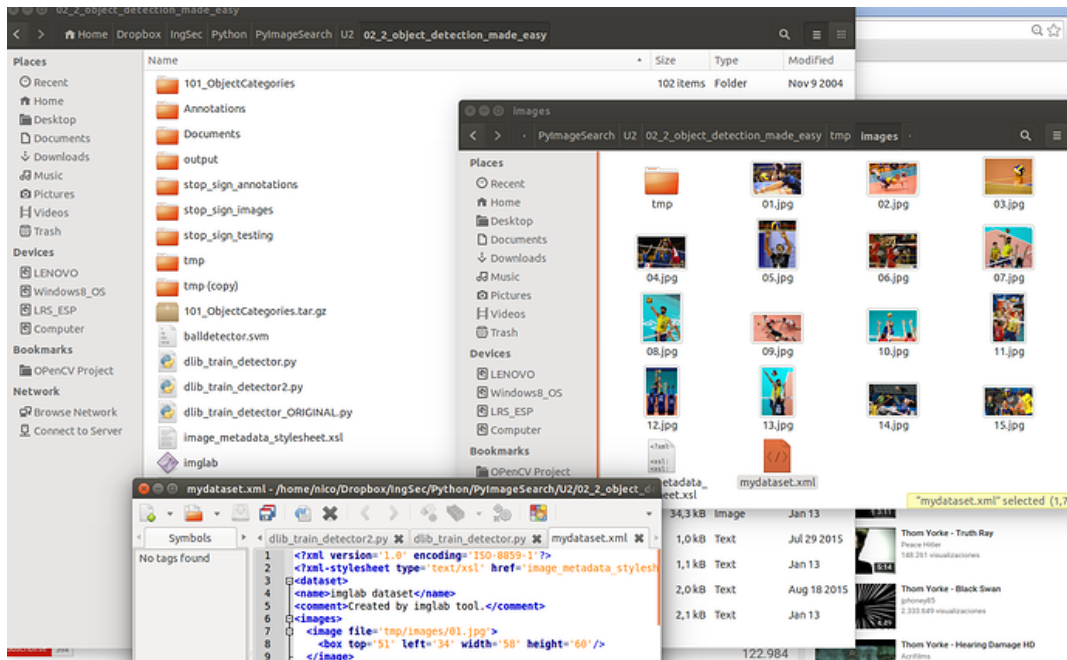
I made a picture from my file tree so it help to understand my explanation..

Hope it make sense!

Thank for the code i will read it and use it, if i have a question i will ask..

One more question: What type of food do you prefer to eat when you come to Berlin 😊 ?

Nico



ThamNgapWei

Jan 16

Thanks for your explanations and congratulation for your success 😊

nicocarasio:

What type of food do you prefer to eat when you come to Berlin

If I have a chance to Berlin(I live at Malaysia), I would like to give Wiener Schnitzel a try.Thanks for you invitation, but I do not think I would go to Berlin any time soon.

Adrian Chief PyImageSearcher

Jan 16

@nicocarasio : The SciPy library actually has a **built-in method** to handle saving arrays as **.mat** format.

Basically, I would suggest using the **BeautifulSoup** Python package (or your favorite HTML/XML parser than can navigate the DOM of the XML document), parse out the bounding box coordinates and image path, and then save as a **.mat** file.

It's actually not as hard as it sounds once you start working with it 

bogdan.craciun

Aug 7

any news on this Adrian? that would be really helpful

Adrian Chief PyImageSearcher

Aug 7

bogdan.craciun:

any news on this Adrian? that would be really helpful

Any news regarding what? I'm not sure what your exact question is.

bogdan.craciun

20d

Adrian:

I have a simple browser interface that I wrote using Django. Basically it accepts a set of images, loads them into a browser interface, and allows you to do the cropping there. Is there any interest in me getting this project online?

sorry for not being precise. I was refering to your first statement above "I have a simple browser interface that I wrote using Django. Basically it accepts a set of images, loads them into a browser interface, and allows you to do the cropping there. Is there any interest in me getting this project online?"

bogdan.craciun

20d

ThamNgapWei:

I wrote a small program to change the ratio of the bounding boxes of the xml file generated by imglab.

why does dlib need to have the same aspect ratio for the bounding boxes? I noticed once, that dlib was giving me an error because the aspect ratio of the bounding boxes was not the same. I don't know what is its tolerance

Adrian Chief PyImageSearcher

20d

To be totally honest, I haven't touched the script in awhile -- I've been too swamped with other projects. I actually highly recommend using Sloth for image annotations:

http://sloth.readthedocs.io/en/latest/first_steps.html

bogdan.craciun:

why does dlib need to have the same aspect ratio for the bounding boxes? I noticed once, that dlib was giving me an error because the aspect ratio of the bounding boxes was not the same. I don't know what is its tolerance

That's not specific to dlib, that's true for *all* HOG-based detectors. Take a look at the "Preparing your experiment and training data" lesson for the reasoning behind this:

<https://gurus.pyimagesearch.com/lessons/preparing-your-training-data/>

The gist is that HOG is a structural descriptor that expects a rectangular image input. We use sliding windows (a rectangle) to determine object location. And that sliding window needs to have an aspect ratio similar to our training data. Therefore, if the aspect ratios for the training data aren't similar then we can't define a rectangular window size for the data.

bogdan.craciun

12d

Ok, I got that. The thing is that chapter 2.5 (preparing your training data) explains that we compute the average width and height, along with the aspect ratio of the bounding boxes. So if we compute an average means that the input bounding boxes have different aspect ratio, comparing one bounding box to another, right? We

compute the average to determine the size of the sliding window.

I also understand that each of the bounding boxes have to have an approximate aspect ratio as the sliding window.

My question is how approximate? what might be the tolerance. Should all of the training data (pictures) that contain the object that I want to detect, be captured from the same distance, angle of view, so that the aspect ratio of the bounding boxes is the "same"?

Adrian Chief PyImageSearcher

11d

bogdan.craciun:

My question is how approximate? what might be the tolerance. Should all of the training data (pictures) that contain the object that I want to detect, be captured from the same distance, angle of view, so that the aspect ratio of the bounding boxes is the "same"?

It's hard to give a value to say what the exact tolerance is since it can vary slightly, but in general, keep in mind that the HOG descriptor is a *structural descriptor*. It does assume that the objects you are trying to detect have approximately the same viewing angle. The distance itself doesn't matter (due to the image pyramid). But the aspect ratios *absolutely* have to be similar.

In fact, if you were to use the dlib library to train a HOG detector on images that had dramatically different aspect ratios it would throw an error saying that the bounding boxes have "impossible" values. You can actually see this error message in the dlib source code:

http://dlib.net/train_object_detector.cpp.html

The dlib library performs this check by computing the average aspect ratio of the bounding boxes and then dividing it by the target size which is derived from the HOG descriptor. If the ratio of the target size and the scale are out of whack it throws the above error message.

Again, if you're working with images with wildly different viewing angles then I would suggest using CNNs instead of the HOG detector