imglocate: locate objects in images and write annotations of detected objects as TSV

Leonardo Taccari
s1069964@studenti.univpm.it

Abstract

imglocate is a Python 3 [11] module/script that uses OpenCV libraries [7] to detect objects in images and write annotations in tab-separated values (TSV) text files.

1 Introduction

In the last decades more and more photographies are digitalized and images are used in several different contexts.

It is often problematic to categorize them. Some cameras, cellphones and other devices can store meta-data like the time-stamp when it was taken. If they have a GPS some of them also store the coordinates where the photograph was taken. However - also given that - it is still difficult to select images based on the content.

Unix tools like `cut(1)`, `grep(1)`, `sed(1)` or programming languages like AWK [3] permits to easily process text efficiently and effectively.

imglocate tries to make the Unix philosophy [6] usable on images as well.

imglocate is a Python 3 [11] module/script that uses OpenCV libraries [7] to detect objects in images and write the corresponding annotations in tab-separated values (TSV) text files. Each detected object has an entry in the annotation with the following fields, in order:

label class label of detected object

confidence confidence

x x coordinates of the bounding box (top-left point)

y y coordinates of the bounding box (top-left point)

height height of the bounding box

width width of the bounding box

imglocate supports two subcommand: *annotate* and *search*.

Given a list of images as argument, `imglocate annotate` create annotates in a TSV text file in the same path of each image appending to them the `.txt` suffix. When running `imglocate annotate` multiple times against the same images, the last modification time (*mtime*) of annotation and image are

checked. The object detection is performed only if the last modification time of the image is newer than the last modification time of the annotation. The `-f` option force to always performs object detection and regenerate the annotation.

Given a label and a list of images - previously annotated via `imglocate annotate` - as argument, `imglocate search` search if the label is present in image and print all the resulting images containing the label to the standard output.

2 Related works

To the knowledge of the author no similar tool exists to detect object in image and then generate a simple text file that can be easily processed in the Unix environment.

The famous optical character recognition (OCR) engine Tesseract OCR [13] (and other similar programs) given an image as argument generate a text file as an output - similarly to imglocate - but with a different purpose, that is the purpose of OCR: recognizing characters in the images, not detecting objects in them.

Luminoth [15] (now unmaintained) provided something similar to `imglocate annotate -s` via `lumi predict` command. However, the output generated was not a simple TSV but a JSON that could be not processed via Unix tools as trivially (despite that the information stored in the JSON are actually the same of the one stored by imglocate).

AWS Command Line Interface [4] has an `aws rekognition detect-labels` [5] command that despite using Amazon Web Services has an interface somewhat similar to imglocate. However, also in this case the output generated is a more complex JSON (containing similar information of imglocate plus possible parents of the objects detected) and the usage/passing of arguments is more complicated than imglocate, making it less "Unix-y".

3 Installation, configuration and usage

3.1 Installation

imglocate is freely available under a 2-Clause BSD license.

The source code can be fetch with Git via:

```
% git clone https://github.com/iamleot/imglocate.git
```

Apart Python the only dependency needed is OpenCV. To install OpenCV via pkgsrc [14], e.g. on NetBSD:

```
% cd pkgsrc/graphics/opencv
% make install
```

Alternatively, on PyPI [10], unofficial pre-built OpenCV packages for Python are available for a couple of platforms and, assuming `pip` is present, they can be installed via (for the latest 3.4.x release available at the moment of writing):

```
% pip install opencv-python==3.4.10.35
```

After cloning it and installing all dependencies, imglocate is just a standalone Python module/script and can be invoked directly via:

```
% cd imglocate
% ./imglocate.py
```

Alternatively it can be installed in possible `/usr/local/bin`, user's `/bin` or similar via just, e.g.:

```
# install -m 0755 imglocate.py /usr/local/bin/imglocate
```

3.2 Configuration

imglocate needs to be configured before it can be used. A configuration file can be provided via the `-c` option. By default the configuration file `~/.imglocaterc` is used.

The configuration field should have an `[imglocate]` section and should contains all the following entries:

`weights` path to the deep learning network weights

`config` path to the deep learning network config

`labels` path to the labels of the classes returned by the deep learning network. There should be one label per line.

`confidence_threshold` confidence threshold

`nms_threshold` Non-Maximum Suppression (NMS) threshold

The frameworks supported by imglocate (and configuration entries `weights`/`config`) are the ones supported by the OpenCV Deep Neural Network module (`dnn`) [8], at the time of writing:

- Caffe
- TensorFlow
- Torch
- Darknet
- DLDT
- ONNX

For example, using YOLOv3 [12] and given `weights` [1], `config` [2], `labels` [3] in a `~/.imglocate` directory, a `confidence_threshold` of 0.2 and an `nms_threshold` of 0.3 the corresponding `imglocaterc` file will be:

[1] YOLOv3 weights can be downloaded at https://pjreddie.com/media/files/yolov3.weights

[2] YOLOv3 config can be downloaded at https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg

[3] YOLOv3 labels can be downloaded at https://github.com/iamleot/imglocate/blob/master/examples/yolov3.labels

```
[imglocate]
weights = ~/.imglocate/yolov3.weights
config = ~/.imglocate/yolov3.cfg
labels = ~/.imglocate/yolov3.labels
confidence_threshold = 0.2
nms_threshold = 0.3
```

Please note that this configuration file is used for all further examples in this paper.

3.3 Usage

imglocate supports two subcommand: *annotate* and *search*.

In the following sections common usages and possible more advanced usages are documented. Please note that possible further usage examples and, in particular, a complete syntax of commands can be found in the imglocate's homepage [1] at https://github.com/iamleot/imglocate.

3.3.1 Annotating images

Given a list of images as argument, `imglocate annotate` create annotations in a TSV text file in the same path of each image appending to it the `.txt` suffix. When running `imglocate annotate` multiple times against the same images, the last modification time (*mtime*) of annotation and image are checked. The object detection is performed only if the last modification time of the image is newer than the last modification time of the annotation. The `-f` option force to always performs object detection and regenerate the annotation.

For example, to annotate a single image `man-woman-walking-opposite-directions.jpg`:

```
% imglocate annotate man-woman-walking-opposite-directions.jpg
```

Once annotated the annotations are available as `man-woman-walking-opposite-directions.jpg.txt`:

```
person  0.99984 539     68      412     743
person  0.99753 138     71      328     705
handbag 0.97554 299     414     218     233
car     0.97361 390     87      84      55
car     0.91132 532     80      51      33
car     0.87065 1045    71      157     87
car     0.74787 581     70      66      59
```

The corresponding image with bounded box drawn [4] can be found in Figure 1.

3.3.2 Searching detected object in annotated images

Given a label and a list of images - previously annotated via `imglocate annotate` - as argument, `imglocate search` search if the label is present in images and print all the resulting images containing the label to the standard output.

For example, after annotating the previous image:

[4] Once annotated via `imglocate annotate` the image with the bounded box in Figure 1 was produced with `examples/draw_bounded_box.py` helper script, part of imglocate.

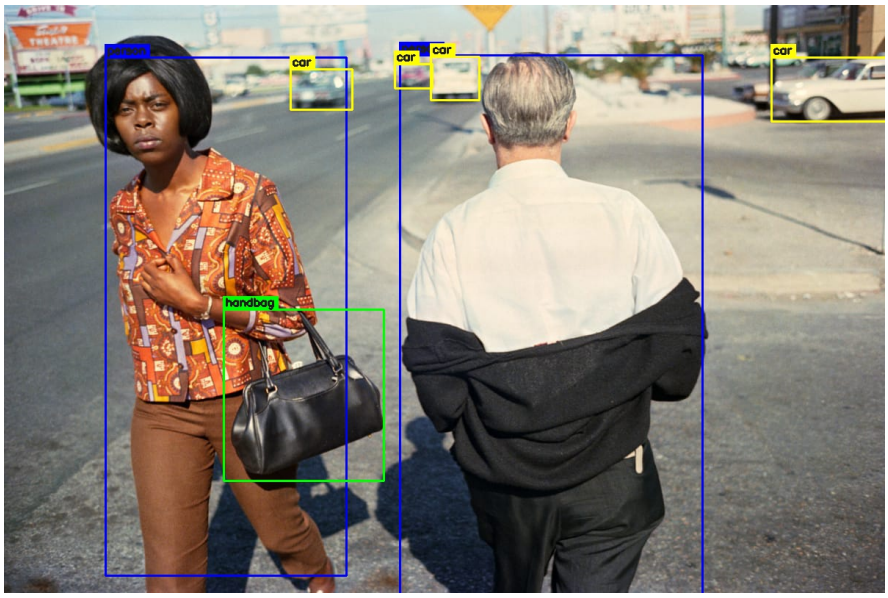

Figure 1: William Eggleston, Untitled, c. 1968, annotated via imglocate using YOLOv3

```
% imglocate search person *.jpg
man-woman-walking-opposite-directions.jpg
```

3.3.3 More advanced usages

Annotation can be easily parallelized via `find(1)` and `xargs(1)`. For example, to recursively annotate all `*.jpg` and `*.png` images in the current directory and parallelize the annotation to always have 6 instance of imglocate running at the same time against a set of 4 images per instance:

```
% find . \( -iname '*.jpg' -or -iname '*.png' \) -print0 |
    xargs -0 -n 4 -P 6 imglocate annotate
```

Given the simplicity of annotations, simple one-liners can be written to search annotated images with a certain criteria. For example, to search all images with at least 5 person in them:

```
% cat > at-least-5-person.sh <<EOF
#!/bin/sh

awk -F '\t' '
$1 == "person" {
        n++
}

END {
        if (n >= 5)
                print substr(FILENAME, 1, length(FILENAME) - 4)
```

```
}' "$1"
EOF
% chmod +x ./at-least-5-person.sh
% find . -name '*.txt' -print0 | xargs -0 -n 1 -P 8 ./at-least-5-person.sh
```

To find all images with exactly one person and one sofa:

```
% cat > one-person-one-sofa.sh <<EOF
#!/bin/sh

awk -F '\t' '
{
        label[$1]++
}

END {
        if (label["person"] == 1 && label["sofa"] == 1)
                print substr(FILENAME, 1, length(FILENAME) - 4)
}' "$1"
EOF
% chmod +x ./one-person-one-sofa.sh
% find . -name '*.txt' -print0 | xargs -0 -n 1 -P 8 ./one-person-one-sofa.sh
```

3.4 Development notes

imglocate was mainly developed on NetBSD/amd64 following the -current branch and pkgsrc-current and OpenCV 3.4.x branch and Python 3.8.5. However, it should work on any other platform supported by Python 3 and OpenCV.

The code is hosted in a Git repository [1] and each commit is subjected to a continuous integration (CI) project via GitHub Actions. The corresponding action can be found in the Git repository as `.github/workflows/python-app.yml`. All the workflows logs can be found at: https://github.com/iamleot/imglocate/actions.

As part of the CI, the YOLOv3-tiny weights and config are downloaded, all dependencies are installed and the code is checked and linted via flake8 [9] and then tested by invoking `imglocate` and comparing the output generated by `imglocate annotate` and `imglocate search` with expected output.

LGTM [2] is periodically run to find possible issues in the code. Current status can be found at: https://lgtm.com/projects/g/iamleot/imglocate/.

4 Future works

imglocate should be considered mostly feature complete and apart maintenance, bug fixing and possible future improvements no further development is planned.

In the Git repository `TODO.md` files document all known TODO/bugs/improvements that should be done.

4.1 Irreproducibility of `imglocate annotate` results

A particularly interesting problem that was pointed out by the test was the irreproducibility of `imglocate annotate` results.

Initially `imglocate annotate` printed the float number of confidence field with a precision of 17 numbers after the dot.

For the same commit, imglocate configuration, YOLOv3-tiny weight/config and all software version used by the testbed the action pointed out the following differences in the `imglocate annotate -fs examples/office_at_night.jpg` (omitting the bounding box fields that are the same for both runs):

```
 person 0.8456319570541382
-chair  0.7067238688468933
-person 0.6016818881034851
-diningtable    0.26269155740737915
+chair  0.7067239880561829
+person 0.6016820073127747
+diningtable    0.26269131898880005
```

We can see that chair confidence differs of $1.1920928955078125e-07$, person confidence differs of $1.1920928955078125e-07$ and diningtable confidence differs of $2.384185791015625e-07$ (that is $2 * 1.1920928955078125e-07$).

Given that such details of precision is not needed the problem was workarounded by limiting the precision to 5 numbers after the dot.

A possible hypothesis is that some component used by imglocate can do a CPU runtime detection of extension instruction set and on a certain testbed this instruction set is available while on the other it is not available and the underlying implementation has such possible multiple-of-$1.1920928955078125e-07$-bug.

5 Conclusion

In this paper we have discussed the design, implementation and usage of imglocate, a Python 3 module/script to locate objects in images and write annotations of detected objects as tab-separated values (TSV).

In section 3.3.3 we have seen how having a simple annotation format easily enable to reuse powerful Unix tools and possibly query for images with certain detected object only by writing simple one-liners.

The imglocate homepage [1] has further information about it and the API is extensively documented and can be accessed via `pydoc imglocate`.

References

[1] imglocate: Locate objects in images and write annotations of detected objects as TSV. URL https://github.com/iamleot/imglocate.

[2] LGTM: Continuous security analysis. URL https://lgtm.com/.

[3] Alfred V Aho, Brian W Kernighan, and Peter J Weinberger. *The AWK Programming Language*. Addison-Wesley Longman Publishing Co., Inc., 1988.

[4] Amazon Web Services. AWS Command Line Interface, . URL https://aws.amazon.com/cli/.

[5] Amazon Web Services. detect-labels - AWS CLI Command Reference, . URL https://awscli.amazonaws.com/v2/documentation/api/latest/reference/rekognition/detect-labels.html.

[6] Malcolm D McIlroy, Elliot N Pinson, and Berkley A Tague. Unix time-sharing system: Foreword. *Bell System Technical Journal*, 57(6):1899–1904, 1978.

[7] OpenCV team. OpenCV, . URL https://opencv.org/.

[8] OpenCV team. OpenCV: Deep Neural Network module, . URL https://docs.opencv.org/master/d6/d0f/group__dnn.html.

[9] PyCQA. flake8. URL https://gitlab.com/pycqa/flake8.

[10] Python Software Foundation. PyPI: The Python Package Index, . URL https://pypi.org/.

[11] Python Software Foundation. Python, . URL https://www.python.org/.

[12] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[13] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[14] The NetBSD Foundation. pkgsrc. URL https://www.pkgsrc.org/.

[15] Tryolabs. Luminoth: Open source toolkit for Computer Vision. URL https://luminoth.ai/.