imglocate

Locate objects in images and write annotations of detected objects as TSV

Leonardo Taccari

s1069964@studenti.univpm.it

Università Politecnica delle Marche

Informatica Multimediale

imglocate

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

Problem

How to search for possible objects contained in images in a à la Unix?

Related works

Tesseract	from images to text, but OCR
Luminoth	(now unmaintained) several dependencies, JSON as output
AWS CLI	(via `aws rekognition detect-labels`) images can be passed as local file (prefixing them via `fileb://`, base64-encoded bytes or S3 objects, JSON as output, processed via Amazon Web Services

Contributions

Provide a simple tool that:

- can be easily scripted to annotate an arbitrary number of images
- write annotations as simple text files that can be easily processed via Unix tools
- search detected objects via annotations

Installation

- Checkout the source code via Git:

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

- Install OpenCV from a package system or (for supported platforms) the unofficial pre-built OpenCV package `opencv-python` from PyPI:

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

- The script is completely standalone and can be installed to e.g. `/usr/local/bin` via:

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

Configuration

Before imglocate can be used a configuration file should be written as `~/.imglocaterc`.

weights	path to the deep learning networks weights [1]
config	path to the deep learning networks configuration
labels	path to the labels of the classes returned by the deep learning network (one label per line)
confidence_threshold	confidence threshold
nms_threshold	Non-Maximum Suppression (NMS) threshold

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

[1] Frameworks supported are the ones supported by OpenCV Deep Neural Network module (dnn), at the time of writing: Caffe, TensorFlow, Torch, Darknet, DLDT, ONNX.

Usage

imglocate supports two subcommands: `annotate` and `search`. By default it uses `~/.imglocaterc` as its configuration file but this can be override via the `-c` option. The `-v` option can be passed several times to increase verbose level.

```
% imglocate -h
usage: imglocate [-h] [-c config_file] [-v] {annotate,search} ...

Locate objects in images

positional arguments:
  {annotate,search}  action
    annotate         annotate images
    search           search annotated images

optional arguments:
  -h, --help         show this help message and exit
  -c config_file     configuration file
  -v                 logging level
```

Usage: annotating images

Given one or more images, detect objects in them, write annotations of detected objects as text with the same file name of the image but appending a `.txt` suffix (if it was not previously annotated or `-f` option was used).

```
% imglocate annotate -h
usage: imglocate annotate [-h] [-f] [-s] image [image ...]

positional arguments:
  image       image to annotate

optional arguments:
  -h, --help  show this help message and exit
  -f          force regen of already existent annotations
  -s          only print annotations (do not write them)
```

Annotation format

The annotation is a text file (TSV: tab-separated values) with 0 (if no object was detected) or more lines, each of them corresponding to an object detected and each of them with six fields separated by a tab character:

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)
width	width of the bounding box
height	height of the bounding box

Usage: searching detected objects in annotated images

Given one or more previously annotated images and a label, search for the label in the corresponding annotations files and print all images matching the label.

```
% imglocate search -h
usage: imglocate search [-h] label image [image ...]

positional arguments:
  label       force regen of already existent annotations
  image       image to search

optional arguments:
  -h, --help  show this help message and exit
```

Examples

- Download all the images from the ‘Eggleston Art Foundation’ homepage:

```
% wget -np -nd -r -A "*.jpg" \
    "http://egglestonartfoundation.org/"
```

- Annotate all downloaded images in parallel via `find(1)` and `xargs(1)`:

```
% find . -name '*.jpg' -print0 |
    xargs -0 -n 6 -P 4 imglocate annotate
```

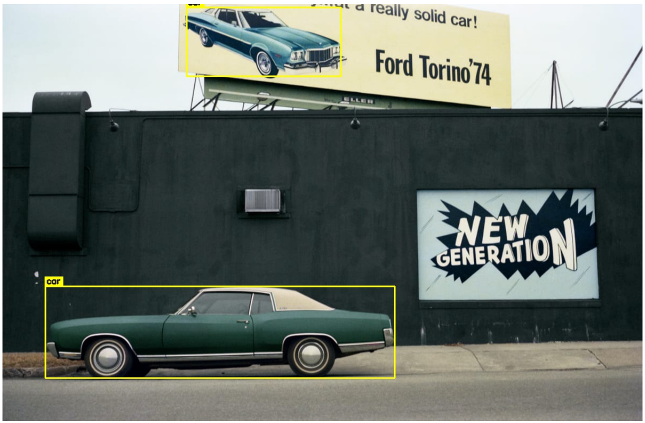
- Search for all annotated images where a car was detected:

```
% imglocate search car *.jpg
1296-04R EGL P_P9A NEW GENERATION.jpg
karco.jpg
man-woman-walking-opposite-directions.jpg
yellow-building.jpg
```

More advanced example

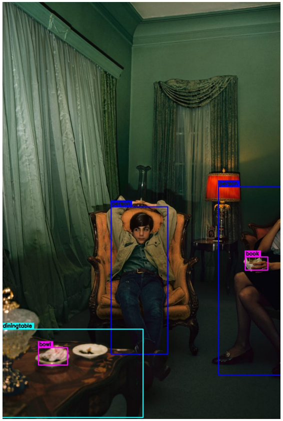
- Search for all annotated images where 2 or more people are present:

  ```
  % awk -F '\t' ' \
  $1 == "person" { a[FILENAME]++ } \
  END { \
      for (i in a) { \
          if (a[i] >= 2) { \
              print substr(i, 1, length(i) - 4) \
          } \
      } \
  }' *.txt
  boy-in-chair.jpg
  dolls.jpg
  man-woman-walking-opposite-directions.jpg
  ```



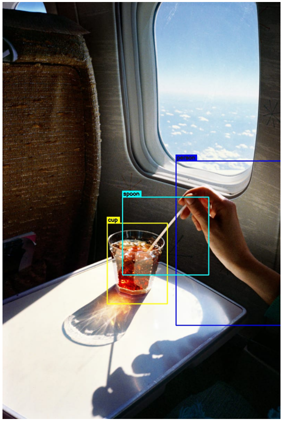
1296-04R EGL P_P9A NEW GENERATION.jpg

```
car 0.99571 80 529 655 172
car 0.94944 345 6 289 129
```



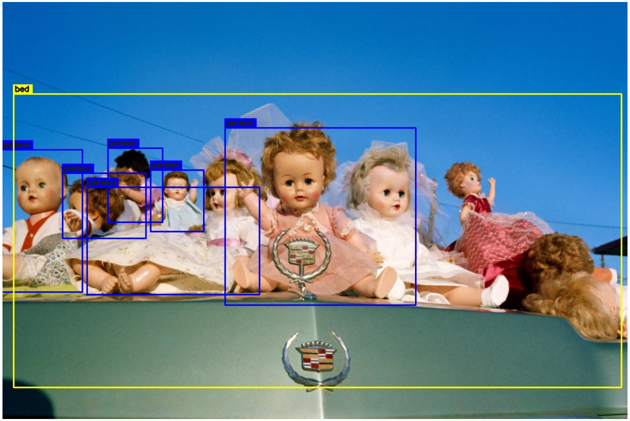
boy-in-chair.jpg

```
person 0.98806 312 589 163 424
diningtable 0.88059 -2 941 406 254
person 0.79687 621 531 180 542
bowl 0.21815 103 994 84 50
book 0.20355 698 733 65 38
```



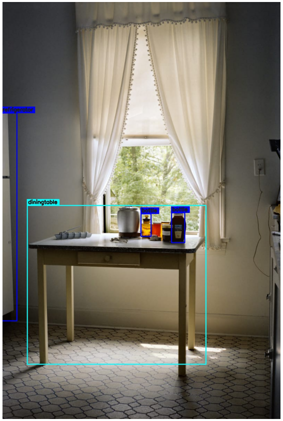
cocktail.jpg

```
person 0.77619 499 457 305 473
cup 0.56096 301 636 173 232
spoon 0.26322 345 561 248 224
```



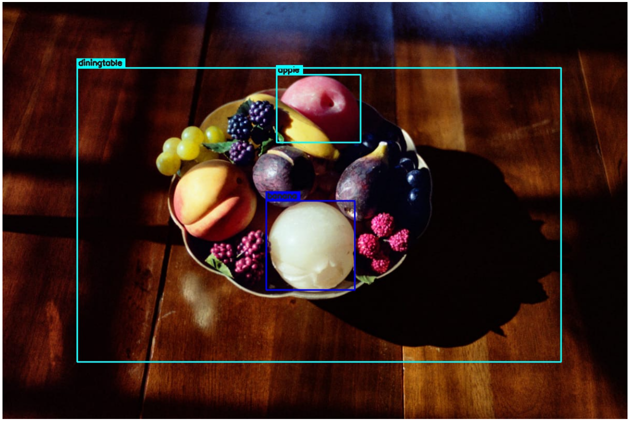
dolls.jpg

```
person 0.86000 427 241 366 340
person 0.84039 -2 283 155 273
person 0.71829 202 280 105 142
person 0.49095 162 355 331 206
person 0.46419 284 322 102 118
person 0.25174 115 328 160 125
bed 0.23555 21 176 1167 563
```



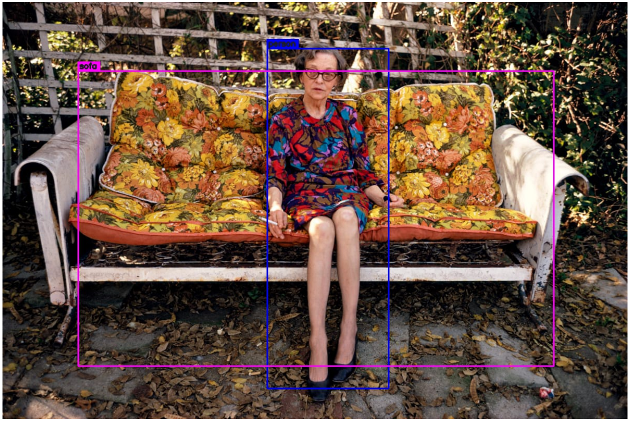
eudoras-kitchen.jpg

```
refrigerator 0.91400 -3 319 44 598
diningtable 0.73617 71 585 514 457
bottle 0.61059 486 606 38 86
bottle 0.33848 398 609 29 65
```



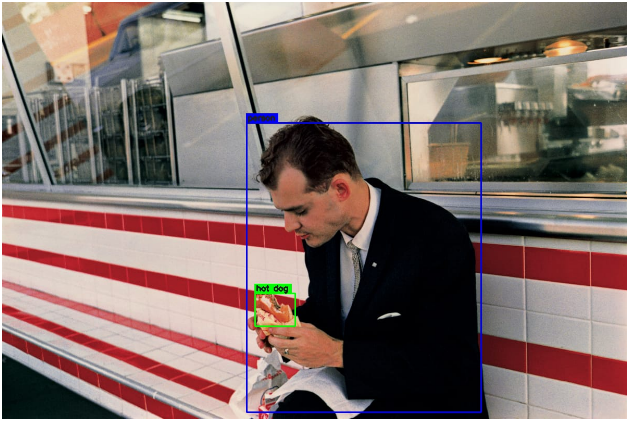
fruit.jpg

```
apple 0.36425 526 139 161 130
diningtable 0.24941 143 126 929 564
banana 0.23542 506 381 170 171
```



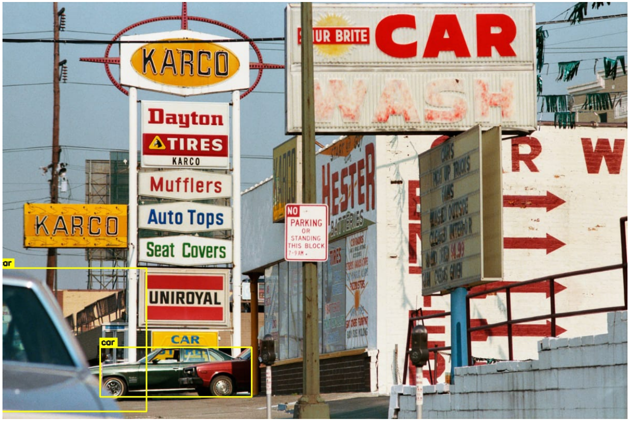
glider.jpg

```
person 0.99920 508 89 233 652
sofa 0.97783 145 132 913 566
```



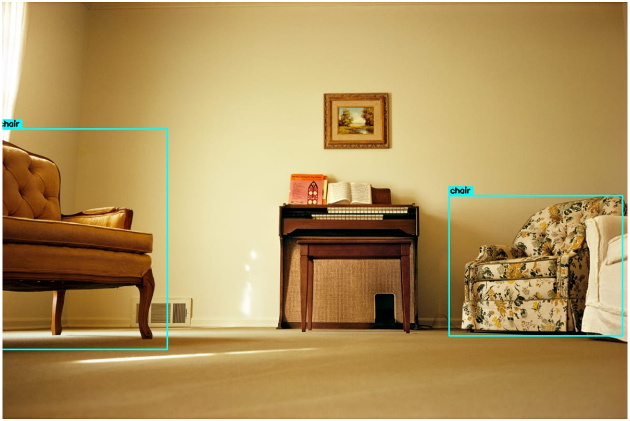
hamburger.jpg

```
person 0.99930 469 232 450 555
hot dog 0.22363 485 559 77 63
```



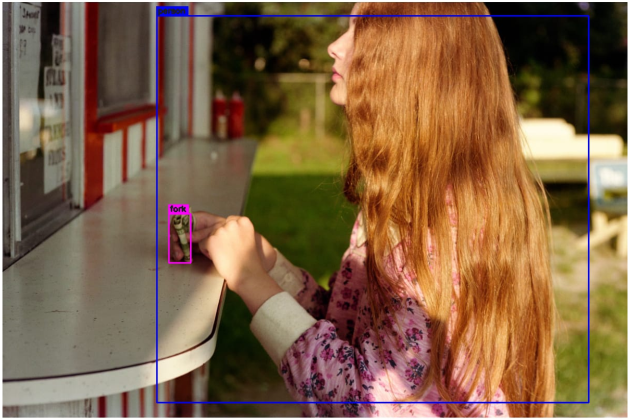
karco.jpg

```
car 0.96579 187 662 291 95
car 0.86057 -11 510 287 275
```



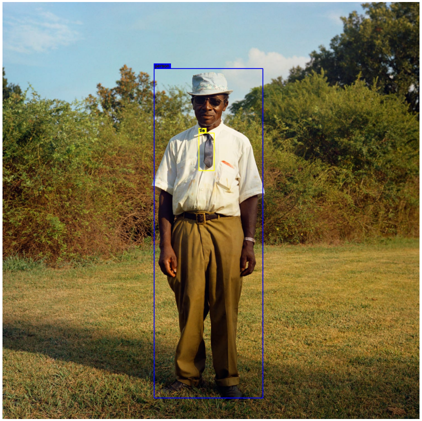
keyboard.jpg

```
chair 0.98876 -9 243 325 423
chair 0.83688 857 371 333 269
```



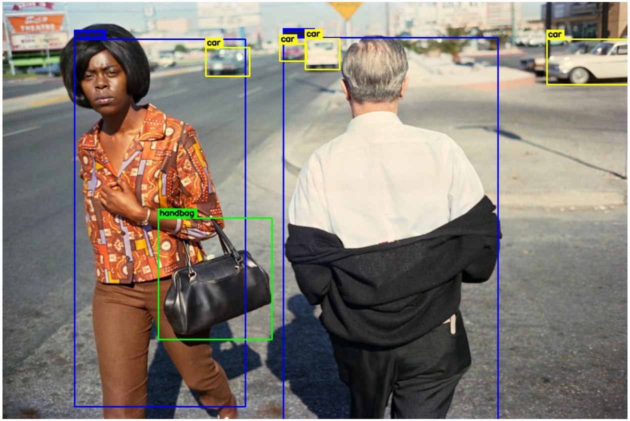
long-hair.jpg

```
person 0.99979 296 27 829 742
fork 0.36005 319 407 42 94
```



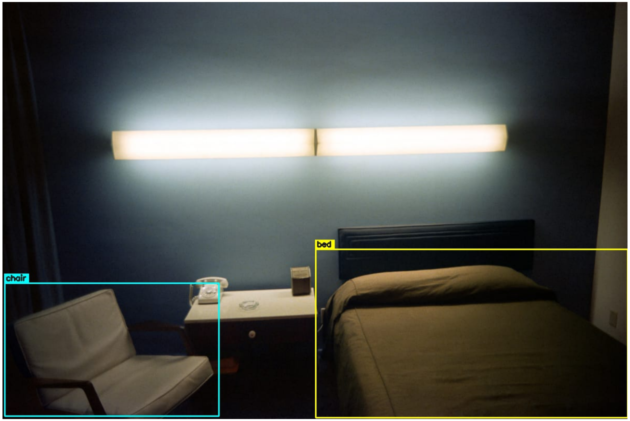
man-standing-in-field.jpg

```
person 0.99967 545 239 391 1185
tie 0.91748 706 471 55 135
```



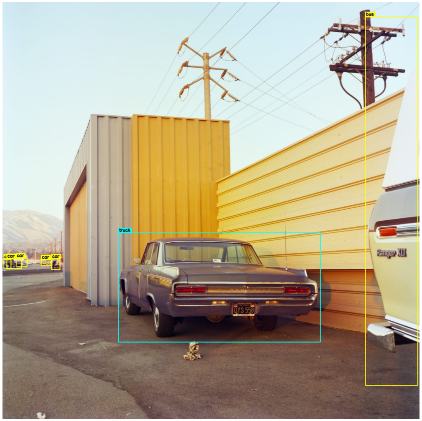
man-woman-walking-opposite-directions.jpg

```
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
```



motel-bed.jpg

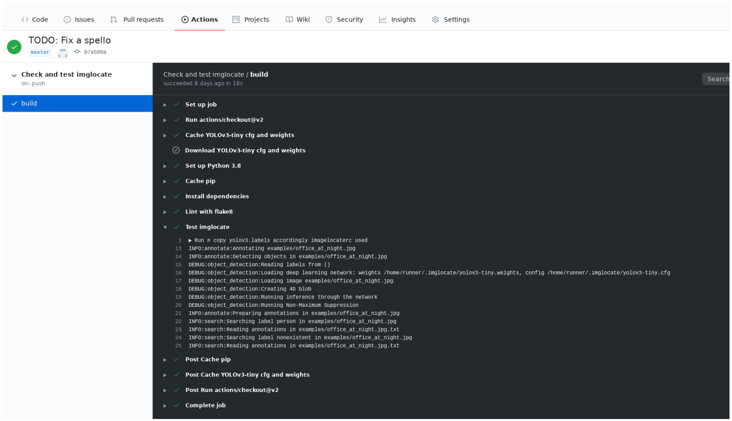
```
bed 0.99206 601 474 599 323
chair 0.92882 4 539 411 255
```



yellow-building.jpg

```
truck 0.96411 417 830 726 393
car 0.53191 4 925 26 32
car 0.51531 15 923 53 35
bus 0.36618 1305 53 188 1326
car 0.29218 176 924 30 38
car 0.20940 139 927 38 21
car 0.20504 49 921 39 29
```

Continuous Integration (CI)



TODO: Fix a spello

master b7a580a

Check and test imglocate
on: push

build

Check and test imglocate / **build**
succeeded 8 days ago in 18s

- ✓ **Set up job**
- ✓ **Run actions/checkout@v2**
- ✓ **Cache YOLOv3-tiny cfg and weights**
- **Download YOLOv3-tiny cfg and weights**
- ✓ **Set up Python 3.8**
- ✓ **Cache pip**
- ✓ **Install dependencies**
- ✓ **Lint with flake8**
- ✓ **Test imglocate**

```
▶ Run # copy yolov3.labels accordingly imagelocaterc used
INFO:annotate:Annotating examples/office_at_night.jpg
INFO:annotate:Detecting objects in examples/office_at_night.jpg
DEBUG:object_detection:Reading labels from ()
DEBUG:object_detection:Loading deep learning network: weights /home/runner/.imglocate/yolov3-tiny.weights, config /home/runner/.imglocate/yolov3-tiny.cfg
DEBUG:object_detection:Loading image examples/office_at_night.jpg
DEBUG:object_detection:Creating 4D blob
DEBUG:object_detection:Running inference through the network
DEBUG:object_detection:Running Non-Maximum Suppression
INFO:annotate:Preparing annotations in examples/office_at_night.jpg
INFO:search:Searching label person in examples/office_at_night.jpg
INFO:search:Reading annotations in examples/office_at_night.jpg.txt
INFO:search:Searching label nonexistent in examples/office_at_night.jpg
INFO:search:Reading annotations in examples/office_at_night.jpg.txt
```

- ✓ **Post Cache pip**
- ✓ **Post Cache YOLOv3-tiny cfg and weights**
- ✓ **Post Run actions/checkout@v2**
- ✓ **Complete job**

Screenshot of logs of imglocate action for CI

Benchmarks and accuracy

- On a machine running NetBSD/amd64 -current (9.99.64) and pkgsrc-current (`opencv-3.4.9nb4` built with `python38-3.8.5`), with an Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz, 8GB of RAM, having all the images on a `tmpfs` filesystem...
- The RAM used (`RES` in `top(1)`) by each imglocate annotate invocation was always around 600M-700M.
- Each imglocate annotate invocation on average needs 53.00 seconds with 2.31 seconds of variance.
- Both memory usage, time needed and accuracy are highly dependent on the framework, weights and config used.

Conclusion

- By just depending on OpenCV and using its Python bindings we have seen that with imglocate is possible to easily annotate arbitrary number of images.
- By using a simple text file and format (TSV) for annotations we have seen how easy is to write possible powerful one-liners for non-trivial queries.