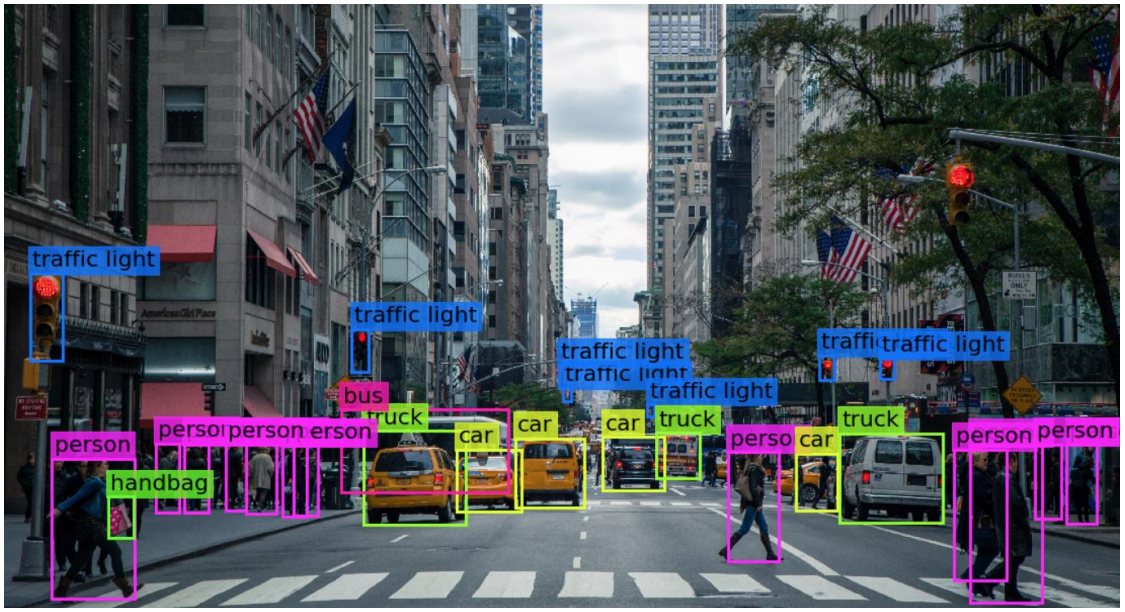


Computer vision in the new era of Artificial Intelligence and Deep Learning

Visión por computador en la nueva era de la Inteligencia Artificial y el Deep Learning

Rubén Usamentiaga*, Alberto Fernández°
*** University of Oviedo**
° TSK

Gijón (Spain)
5 – 16 April 2021

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

Google Colab

Collection of features, utilities and tricks on Colab and/or Python

collection_of_some_features_utilities_and_tricks.ipynb

Open in Colab collection_of_some_features_utilities_and_tricks.ipynb

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

Markdown

```
Markdown | Preview
--- | ---
`**bold text**` | **bold text**
`*italicized text*` or `_italicized text_` | *italicized text*
`` `Monospace` `` | `Monospace`
`~~strikethrough~~` | ~~strikethrough~~
`[A link](https://www.google.com)` | [A link](https://www.google.com)
`![An image](https://www.google.com/images/rss.png)` | ![An image](https://www.google.com/images/rss.png)
```

Markdown	Preview
`**bold text**`	**bold text**
`*italicized text*` or `_italicized text_`	*italicized text*
`` `Monospace` ``	`Monospace`
`~~strikethrough~~`	~~strikethrough~~
`[A link](https://www.google.com)`	A link
`![An image](https://www.google.com/images/rss.png)`	

Markdown

```
* One
* Two
* Three
```

- One
- Two
- Three

```
First column name   | Second column name
--------------------|-------------------
Row 1, Col 1        | Row 1, Col 2
Row 2, Col 1        | Row 2, Col 2
```

First column name	Second column name
Row 1, Col 1	Row 1, Col 2
Row 2, Col 1	Row 2, Col 2

```
# Run Commands

To run commands in cells, we
 can simply prefix an exclam
ation mark before the comman
d.
```

Run Commands

To run commands in cells, we can simply prefix an

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

```
<img align="left" style="padding-
right:10px;" src ="https://raw.githubusercontent.com/albertofernandezvill
an/computer-vision-and-deep-learning-
course/main/assets/university_oviedo_logo.png" width=300 px>
```

Run commands

To run commands in cells, we can simply prefix an exclamation mark before the command. For example, we can use pip command to install/uninstall a package

```
!pip install python-dotenv==0.15.0
```

One common issue associated with running commands is how we interact with prompts, such as installation confirmation (yes or no). The trick is to append the yes flag (-y) to the command.

```
!pip uninstall python-dotenv==0.15.0 -y
```

Active variables and API Lookups

To see the current variables, we can use the **magic method** %who or %whos

```
[1]  a = 5
     b = "hello"
     c = [1,2,3]
```

```
[2]  %who

     a       b       c
```

```
[3]  %whos

     Variable   Type    Data/Info
     ----------------------------
     a          int     5
     b          str     hello
     c          list    n=3
```

We can make use of ? to explore the API we are working with.

```
import cv2

?cv2.COLOR_BGR2*
```

```
cv2.COLOR_BGR2BGR555
cv2.COLOR_BGR2BGR565
cv2.COLOR_BGR2BGRA
cv2.COLOR_BGR2GRAY
cv2.COLOR_BGR2HLS
cv2.COLOR_BGR2HLS_FULL
cv2.COLOR_BGR2HSV
cv2.COLOR_BGR2HSV_FULL
cv2.COLOR_BGR2LAB
cv2.COLOR_BGR2LUV
```

```
import cv2

?cv2.cvtColor
```

```
Docstring:
cvtColor(src, code[, dst[, dstCn]]) -> dst
.   @brief Converts an image from one color space to
another.
.
.   The function converts an input image from one color
space to another. In case of a transformation
.   to-from RGB color space, the order of the channels
should be specified explicitly (RGB or BGR). Note
.   that the default color format in OpenCV is often
```

Magic commands

IPython Magic Commands are added on top of the normal Python syntax and are prefixed by the % character:

- ❑ line magics (%) operate on a single line of input
- ❑ cell magics (%%) operate on multiple lines of input

```
# Use %run command to execute the downloaded Python script
%run opencv_python_version.py

OpenCV version: '4.1.2'
Major version: ''4'
Minor version: '1'
Subminor version: '2'
```

Timing Code Execution: %time and %timeit

%time: Time the execution of a single statement

%timeit: Time repeated execution of a single statement for more accuracy

```
[8]  %timeit sum(range(10000000))

     1 loop, best of 5: 211 ms per loop
```

```
[9]  %%timeit

     a = 10
     b = 50
     sum(range(10000000))
     a = a+b

     1 loop, best of 5: 209 ms per loop
```

```
[ ]  %%time

     total = 0
     for i in range(10000):
         for j in range(1000):
             total += i * (-1) ** j

     CPU times: user 3.87 s, sys: 489 µs, total: 3.87 s
     Wall time: 3.88 s
```

Progress bar

We can make use of tqdm to show the progress bar in a loop. With tqdm, we can instantly make our loops show a smart progress meter: just wrap any iterable with tqdm(iterable)

```
[10] import tqdm

     print("tqdm version: '{}'".format(tqdm.__version__))

     tqdm version: '4.41.1'
```

```
from tqdm import tqdm
import time

for i in tqdm(range(10)):
  time.sleep(1)
```

```
 40%|████      | 4/10 [00:04<00:06,  1.00s/it]
```

Environment Variables

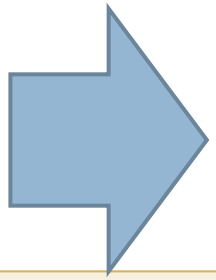
It is not recommended to upload private information to Github or other platforms (even if your repository is private). We can set and get the environment variables in Colab like this:

```
import os

# Append more values to an existing variable:
os.environ['PATH'] += ":/example/path/to/be/added"
# Create a new one:
os.environ['SECRET_ACCESS_KEY'] = "myKey"
```

Wen can access the value of these environment variables using os.getenv()

```
[17] print(os.getenv('PATH'))
     print(os.getenv('SECRET_ACCESS_KEY'))
```



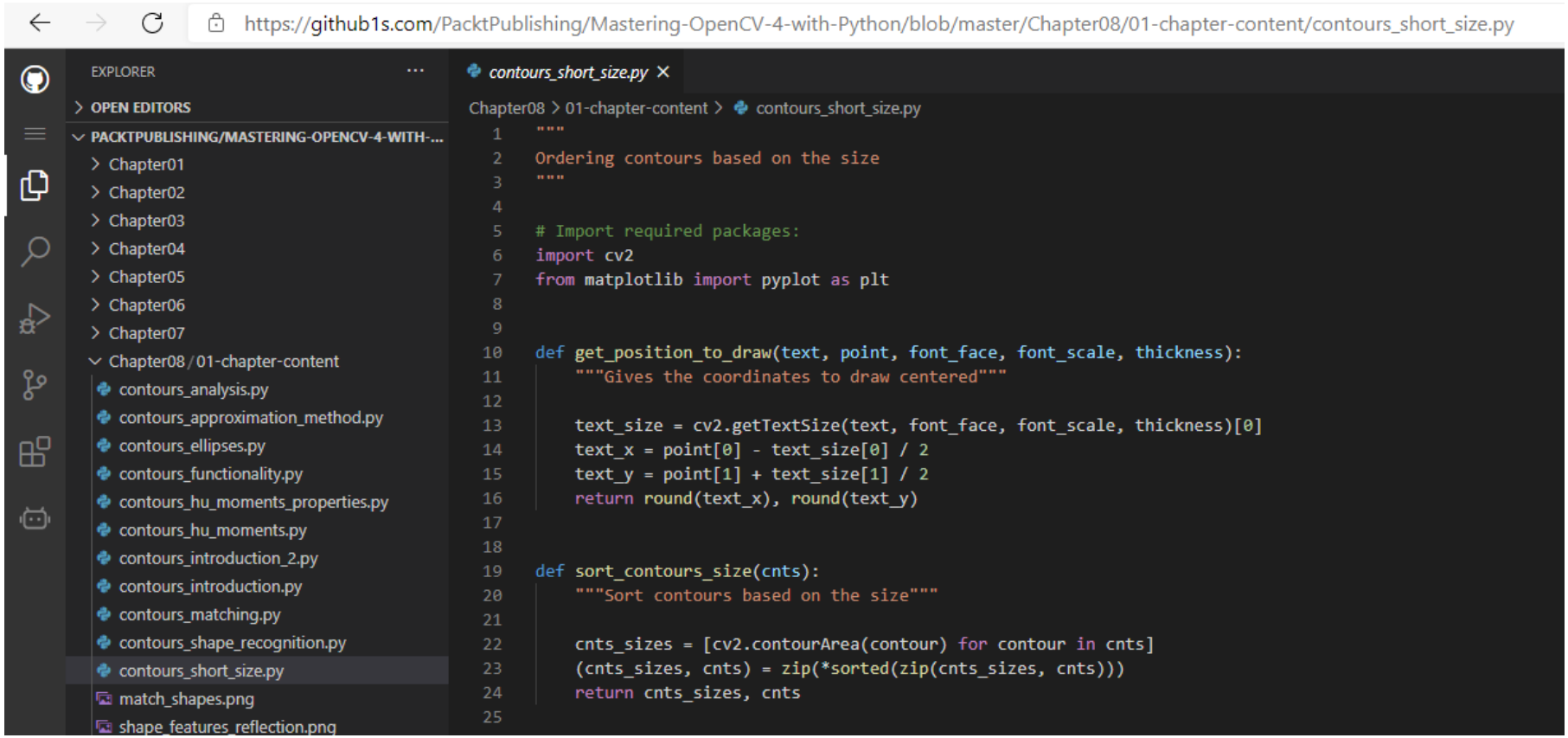
```
/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/tools/node/bin:/tools/google-cloud-sdk/bin:/opt/bin:/example/path/to/be/added
```

```
myKey
```

Github1s - Browse Projects on VSCode in Your Browser

https://**github.com**/PacktPublishing/Mastering-OpenCV-4-with-Python/blob/master/Chapter08/01-chapter-content/contours_short_size.py

https://**github1s.com**/PacktPublishing/Mastering-OpenCV-4-with-Python/blob/master/Chapter08/01-chapter-content/contours_short_size.py



Create and share beautiful images of your source code

We are going to make use of Carbon. There are a few different ways to import code into Carbon (see accompanying notebook)

https://gist.github.com/albertofernandezvillan/050687ee7f3b01aacd2277a5e7b2f125

https://carbon.now.sh/050687ee7f3b01aacd2277a5e7b2f125

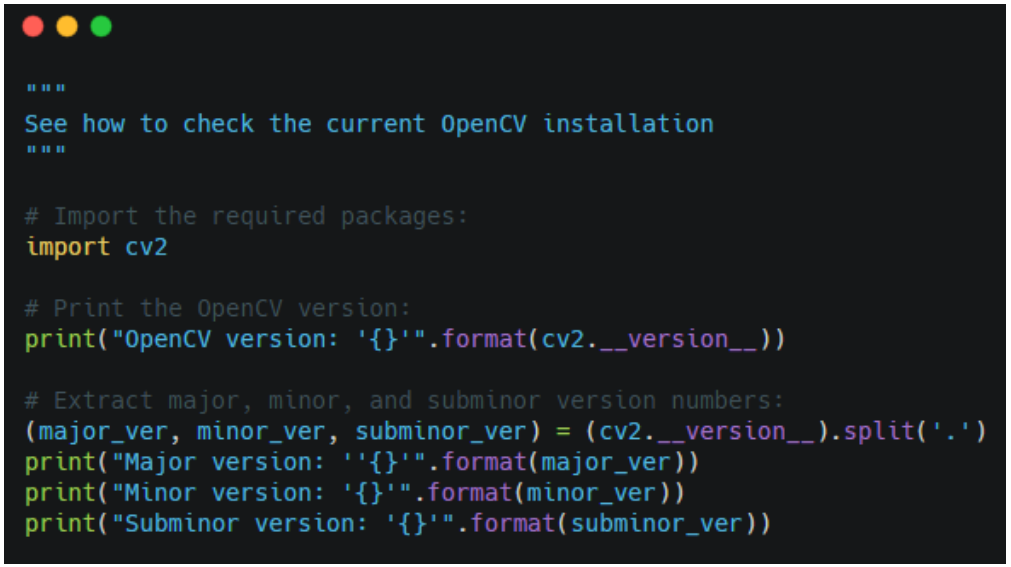
opencv_python_version.py

```
"""
See how to check the current OpenCV installation
"""

# Import the required packages:
import cv2

# Print the OpenCV version:
print("OpenCV version: '{}'".format(cv2.__version__))

# Extract major, minor, and subminor version numbers:
(major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')
print("Major version: ''{}'".format(major_ver))
print("Minor version: '{}'".format(minor_ver))
print("Subminor version: '{}'".format(subminor_ver))
```



Google Colab

Collection of features, utilities and tricks on Colab and/or Python

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course