NUS Fintech Society

Machine Learning Department
Training Wing

Session 9: Computer Vision (07/11/2020)

Content

- Computer Vision
- Photo Time
- Survey Time
    - Training Wing Feedback
    - Training Head Application

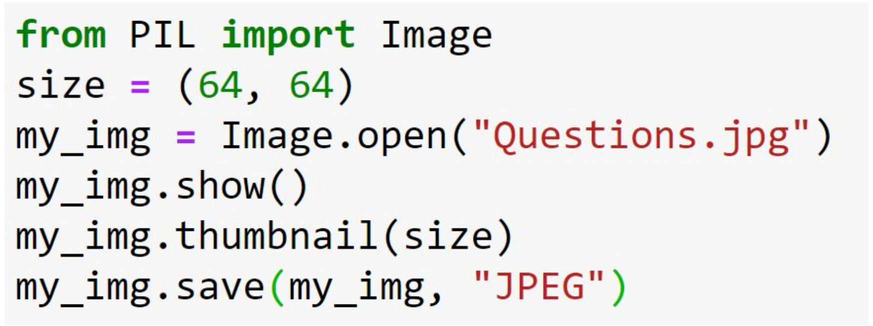
1. Reading the image

PIL

- Python Imaging Library
- Adds image processing capabilities to Python interpreter
- This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

PIL

- PIL python image library Pillow, it is for load\process\create images.
- To load, display, save an image looks like below:

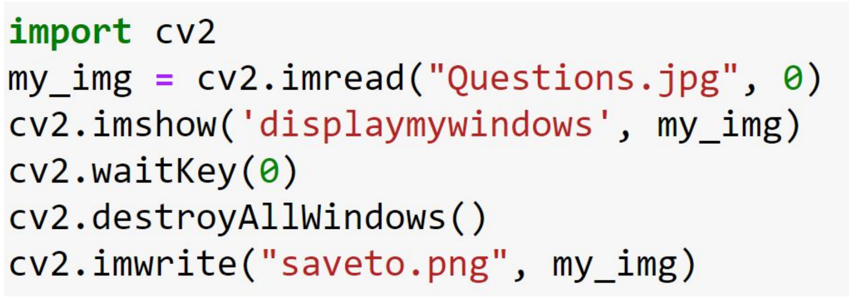


```python
from PIL import Image
size = (64, 64)
my_img = Image.open("Questions.jpg")
my_img.show()
my_img.thumbnail(size)
my_img.save(my_img, "JPEG")
```

- The Image module provides a class with the same name which is used to represent a PIL image. The module also provides a number of factory functions, including functions to load images from files, and to create new images.
- .show(): Displays this image. This method is mainly intended for debugging purposes.

OpenCV2

- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
- To load, display, save an image looks like below:



```
import cv2
my_img = cv2.imread("Questions.jpg", 0)
cv2.imshow('displaymywindows', my_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite("saveto.png", my_img)
```

- .imshow(): Displays an image in the specified window.
- waitKey(0): will display the window indefinitely until any key press (it is suitable for image display).
- waitKey(1): will display a frame for 1 ms, after which display will be automatically closed
- .destroyAllWindows(): Close all open windows

Convolution

- Traditional Image Processing Convolution Filters

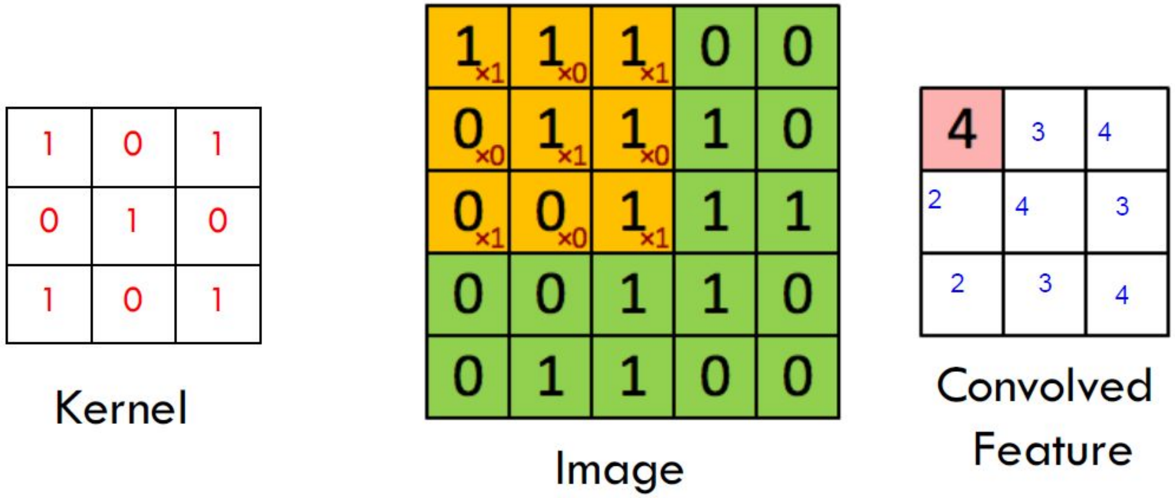
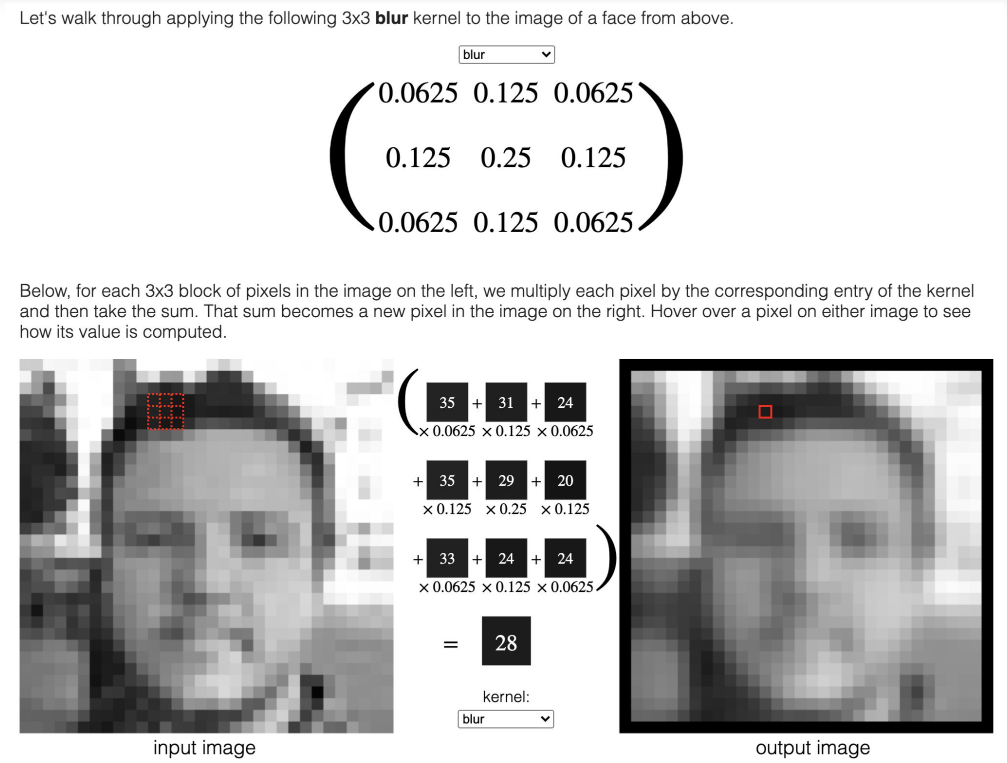


Image Kernels Explained



Let's walk through applying the following 3x3 **blur** kernel to the image of a face from above.

$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.

$$\begin{pmatrix} \boxed{35} \times 0.0625 + \boxed{31} \times 0.125 + \boxed{24} \times 0.0625 \\ + \boxed{35} \times 0.125 + \boxed{29} \times 0.25 + \boxed{20} \times 0.125 \\ + \boxed{33} \times 0.0625 + \boxed{24} \times 0.125 + \boxed{24} \times 0.0625 \end{pmatrix} = \boxed{28}$$

kernel: blur

https://setosa.io/ev/image-kernels/

Gaussian Kernel

- The ‘kernel’ for smoothing, defines the shape of the function that is used to take the average of the neighboring points.
- A Gaussian kernel is a kernel with the shape of a Gaussian (normal distribution) curve and used to remove Gaussian noise

One Dimension

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Two Dimension

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Back to the code

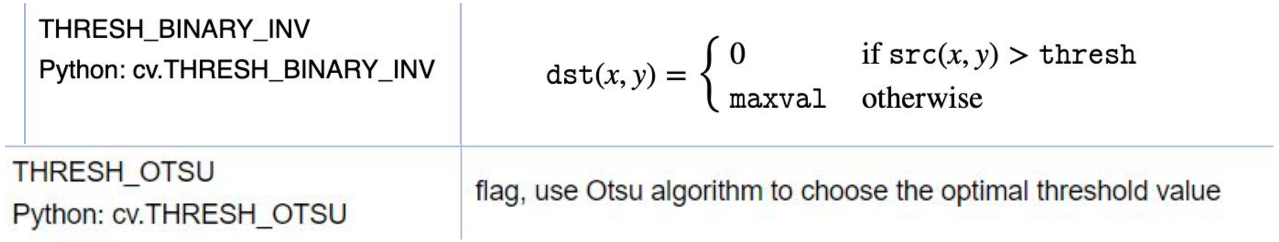
- Let’s look at the Jupyter Notebook “1. Reading the image”

2. Threshold the grayscale image

Threshold the gray image

- From the documentation of OpenCV2,

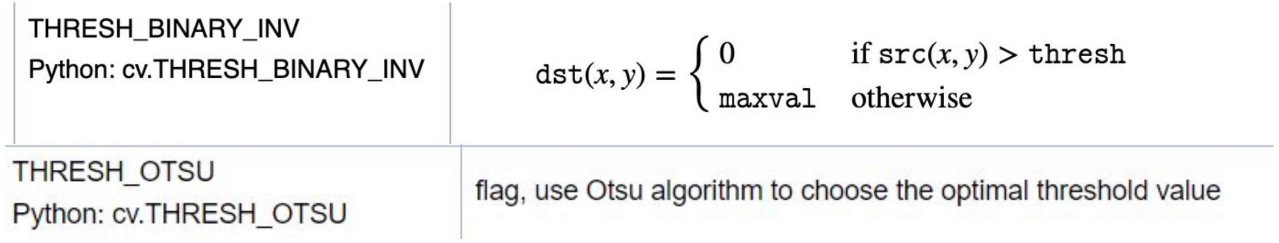
https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a19120b1a11d8067576cc24f4d2f03754



THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_OTSU Python: cv.THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value

- From a grayscale image, thresholding can be used to create binary images, which are images that consists of pixels that can have one of exactly two colors, usually black and white.
- Image thresholding reduces the noise in the image. As mentioned above, each pixel can only have two possible values, such as 0 and 255 instead of a range of values from 0-255.

Threshold the gray image



THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_OTSU Python: cv.THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value

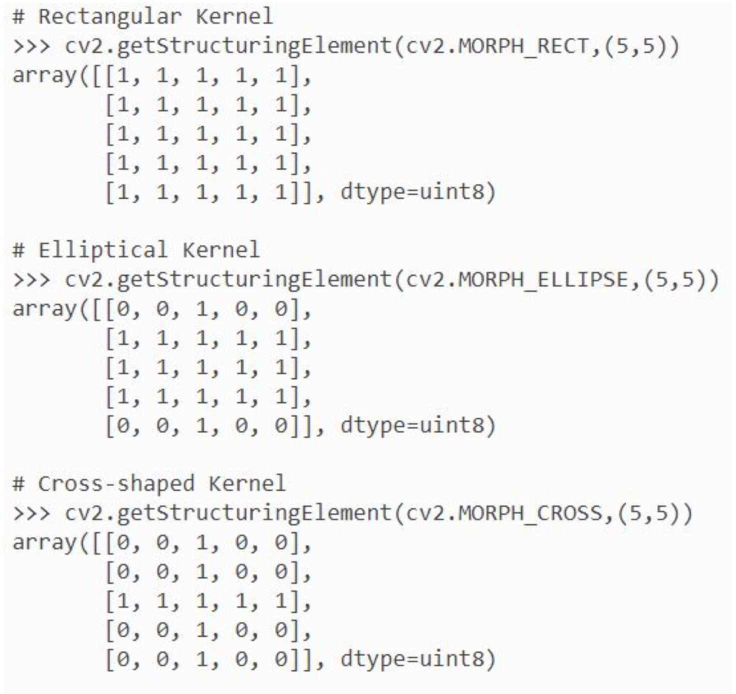
- Here, the matter is straight forward. If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is cv2.threshold.
- 0 is black, 255 is white
- Setting thresh as 0, any values greater than 0 will be set as 0. Hence, white parts are set to black
- Setting maxval as 255, Any values equal to 0 will be set as 255. Hence, black parts are set to white

Back to the code

- Let’s look at the Jupyter Notebook “2. Threshold the grayscale image”

3. Morphological transformation

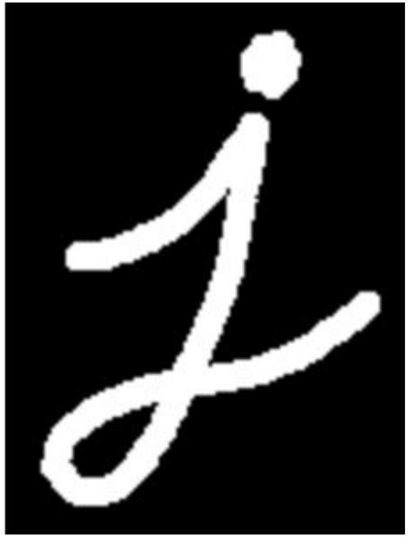
Structuring Elements



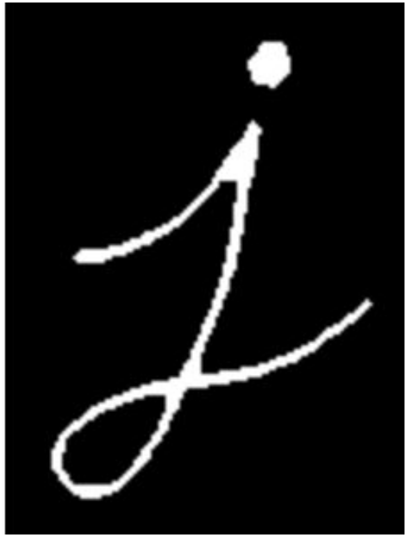
```
# Rectangular Kernel
>>> cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]], dtype=uint8)

# Elliptical Kernel
>>> cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
array([[0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0]], dtype=uint8)

# Cross-shaped Kernel
>>> cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))
array([[0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0]], dtype=uint8)
```

- Remember the image kernels explained earlier?
- OpenCV has a function, cv2.getStructuringElement(). You just pass the shape and size of the kernel, you get the desired kernel.
- The first kernel is a rectangular kernel, second kernel is an ellipse kernel, while the third kernel is a cross kernel (Notice how the '1's represent a cross!)

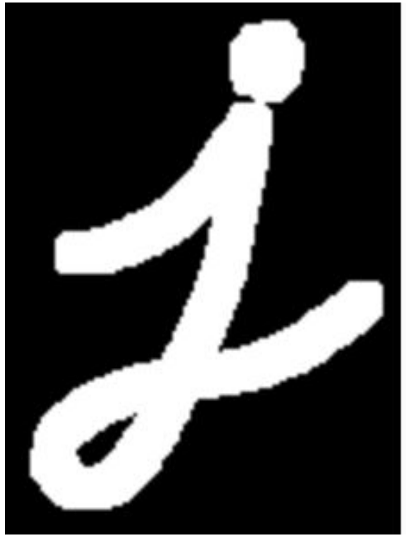
Erosion and Dilation



Original



Erosion



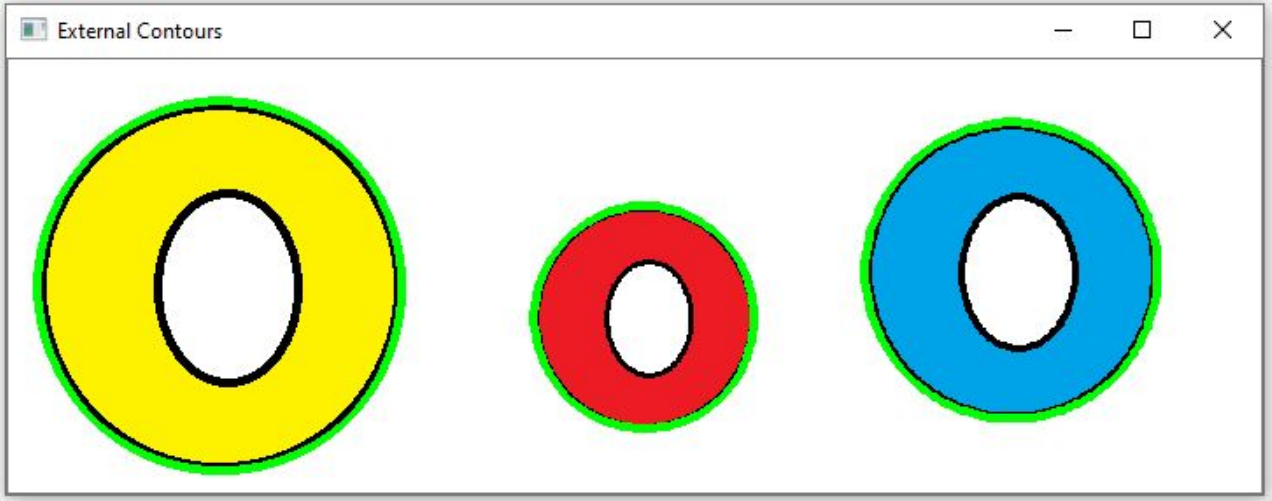
Dilation

Back to the code

- Let’s look at the Jupyter Notebook “3. Morphological transformation”

4. Contours generation

Contours



- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
- For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.

Back to the code

- Let’s look at the Jupyter Notebook “4. Contours generation”

5. Using Pytesseract on the contours

Pytesseract

- For Pytesseract, you need to run pip install pytesseract,
- For Tesseract on Mac:

  Step1: Install using MacPorts or Homebrew

  https://github.com/tesseract-ocr/tesseract/wiki

  Step2: Might need to add path to install tesseract by running export PATH=$PATH:[the required path]

- For Tesseract on Windows:

  Step1: Install using pacman

  https://github.com/tesseract-ocr/tesseract/wiki

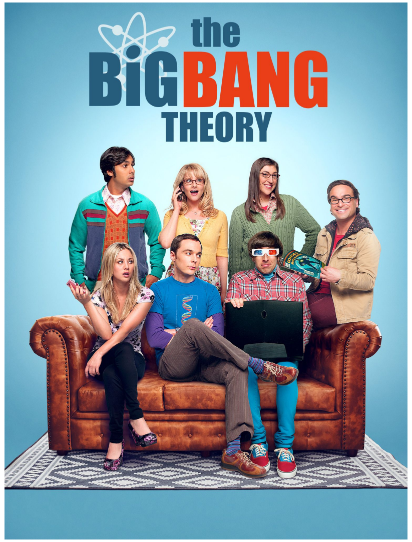
  Step2: Make sure to add the directory where the tesseract-OCR binaries are located to the Path variables!

Pytesseract

- Tesseract is an open source text recognition (OCR) Engine to extract printed text from images
- Allows you to extract words from an image!
- Pytesseract is a wrapper for Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

Pytesseract

Expectations

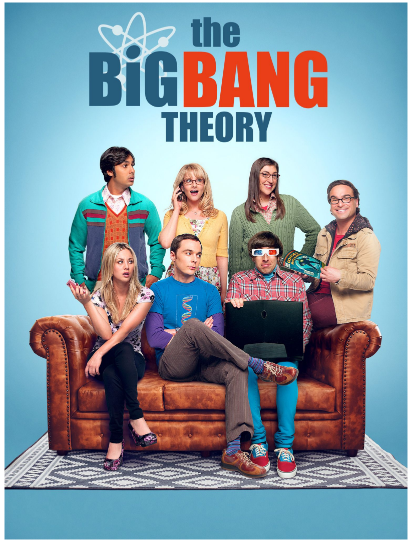


$\rightarrow$

the BiG BANG THEORY

Pytesseract

Reality

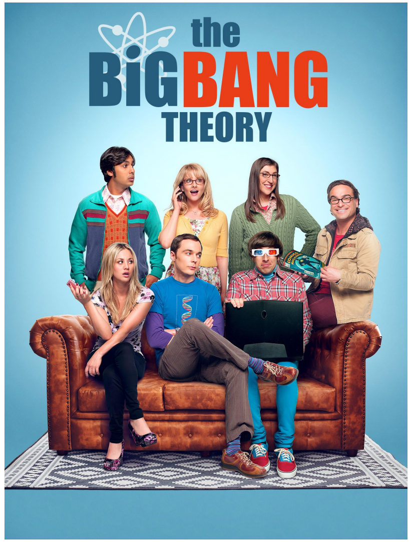


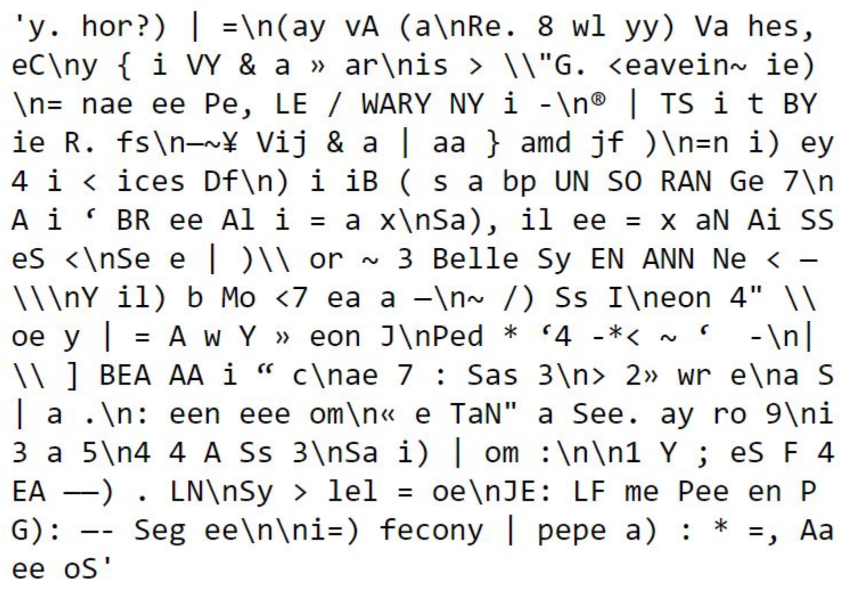
```
'y. hor?) | =\n(ay vA (a\nRe. 8 wl yy) Va hes,
eC\ny { i VY & a » ar\nis > \\"G. <eavein~ ie)
\n= nae ee Pe, LE / WARY NY i -\n® | TS i t BY
ie R. fs\n—~¥ Vij & a | aa } amd jf )\n=n i) ey
4 i < ices Df\n) i iB ( s a bp UN SO RAN Ge 7\n
A i ‘ BR ee Al i = a x\nSa), il ee = x aN Ai SS
eS <\nSe e | )\\ or ~ 3 Belle Sy EN ANN Ne < —
\\\nY il) b Mo <7 ea a —\n~ /) Ss I\neon 4" \\
oe y | = A w Y » eon J\nPed * ‘4 -*< ~ ‘  -\n|
\\ ] BEA AA i “ c\nae 7 : Sas 3\n> 2» wr e\na S
| a .\n: een eee om\n« e TaN" a See. ay ro 9\ni
3 a 5\n4 4 A Ss 3\nSa i) | om :\n\n1 Y ; eS F 4
EA ——) . LN\nSy > lel = oe\nJE: LF me Pee en P
G): —- Seg ee\n\ni=) fecony | pepe a) : * =, Aa
ee oS'
```

Pytesseract

Why did this happen?: Image processing has to be done first for complicated images!

This includes converting the image to grayscale, and finding contours as we learnt earlier!

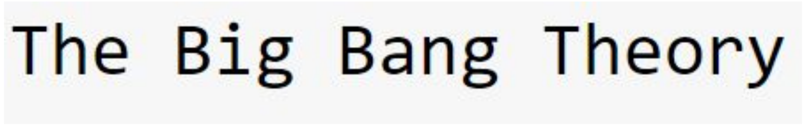




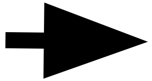
```
'y. hor?) | =\n(ay vA (a\nRe. 8 wl yy) Va hes,
eC\ny { i VY & a » ar\nis > \\"G. <eavein~ ie)
\n= nae ee Pe, LE / WARY NY i -\n® | TS i t BY
ie R. fs\n—~¥ Vij & a | aa } amd jf )\n=n i) ey
4 i < ices Df\n) i iB ( s a bp UN SO RAN Ge 7\n
A i ‘ BR ee Al i = a x\nSa), il ee = x aN Ai SS
eS <\nSe e | )\\ or ~ 3 Belle Sy EN ANN Ne < —
\\\nY il) b Mo <7 ea a —\n~ /) Ss I\neon 4" \\
oe y | = A w Y » eon J\nPed * ‘4 -*< ~ ‘  -\n|
\\ ] BEA AA i “ c\nae 7 : Sas 3\n> 2» wr e\na S
| a .\n: een eee om\n« e TaN" a See. ay ro 9\ni
3 a 5\n4 4 A Ss 3\nSa i) | om :\n\n1 Y ; eS F 4
EA —) . LN\nSy > lel = oe\nJE: LF me Pee en P
G): -- Seg ee\n\ni=) fecony | pepe a) : * =, Aa
ee oS'
```

Pytesseract

How about a simpler image?: We get a much cleaner result with Pytesseract!



The Big Bang Theory



```
'The Big Bang Theory'
```

Back to the code

- Let’s look at the Jupyter Notebook “5. Using Pytesseract on the contours”

Pytesseract

How can we further improve the OCR accuracy of Pytesseract?

By performing relevant image processing techniques, such as:

Inverting images (We taught this!)

Rescaling

Binarisation (We taught this!)

Noise Removal

Dilation and Erosion (We taught this!)

Rotation / Deskewing

Scanning border Removal

For more detailed information, Pytesseract provides further explanation:
https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html

6. Debrief

Photo Time

😊

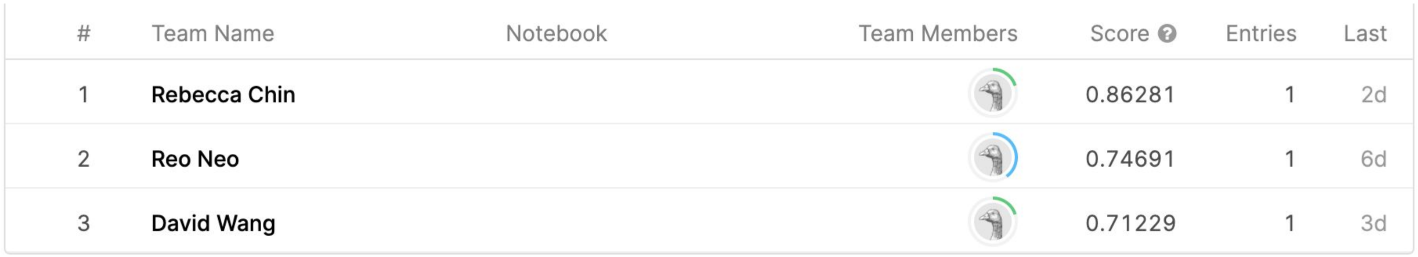
Say cheese! 🧀

Please turn on your Camera for a virtual photo shoot.

Reminder of Project 2

News Headline Sentiment Analysis
DDL: 14/11/2020 11:59 PM

https://www.kaggle.com/t/f719b905f4464c47bb4e56ca2b1386f2



#	Team Name	Notebook	Team Members	Score	Entries	Last
1	Rebecca Chin			0.86281	1	2d
2	Reo Neo			0.74691	1	6d
3	David Wang			0.71229	1	3d

Feedback for us!

We really need your feedback so we can improve the experience for future Fintech members.

Thank you for taking your time to attend the training lessons every Saturday morning. We hope that you have learnt sufficient ML knowledge for you to contribute to the numerous Fintech projects available in the next semester.

Let us spend 10 minutes to fill out this form. Feel free to ask us questions about the survey.

https://forms.gle/bCneWrw2QuTJnDyn7

ML Training Head Application

Do you want to continue our legacy?
Please reply by **21/11/2020**

https://forms.gle/Y1ryzEqrH54SLB5s9

THANKS!

And much thanks to these ppl :D

- Reon, Vanessa, Hubert, Samuel for their guidance and support
- And of course our Training Wing members