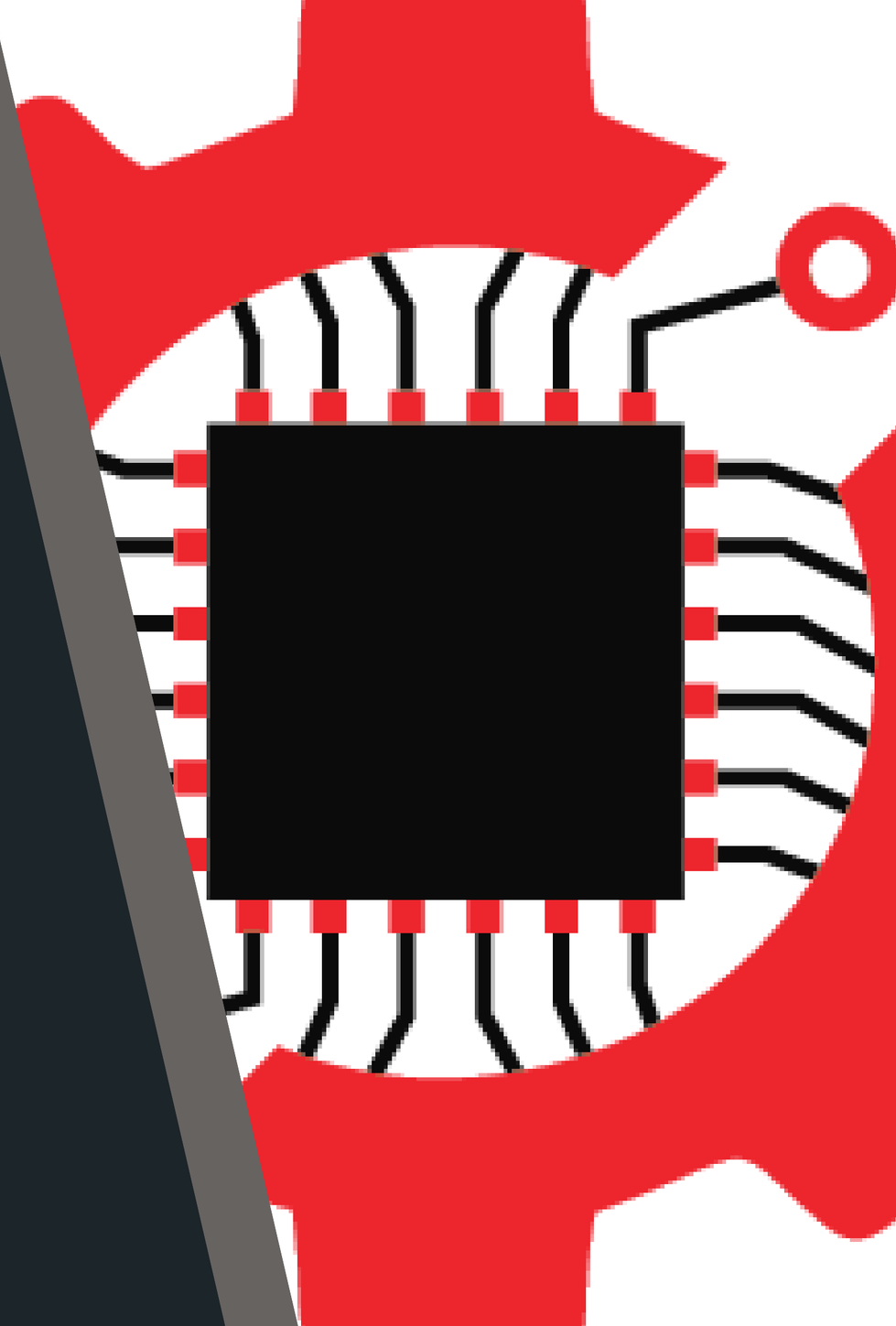
UNM ROBOTICS SOCIETY

OPENCV IMAGE PROCESSING WORKSHOP

Organizing Chairperson: Chien Yü
Email: Jwjier@gmail.com



CONTENTS

PYTHON
(MONTY PYTHON?)

WHY IS IT SO POPULAR

- Data analysis (Pandas)
- Machine Learning (TensorFlow, PyTorch, Keras)
- Web Development - Backend (Django, Flask)
- One of the simplest syntax

PYTHON LIBRARY

WHAT IS PYTHON LIBRARY? WHY SHOULD I USE IT?

Python library is a collection of related modules which contains bundles of code that can be used repeatedly in different programs.
(Remember DRY! - Don’t Repeat Yourself)

A lot of python libraries are actually written in C for improved performance (e.g. numpy)

PYTHON STANDARD LIBRARY

E.g. ”os”, ”requests”, ”time”, ”argparse”, ”logging”

(EE Year 1 students, look into ”threading” and other multithreading, concurrent library to improve your code performance for Project Week)

For more info

PYTHON PACKAGE MANAGEMENT SYSTEM - PIP

INSTALLING EXTERNAL LIBRARIES

pip install {library name}

LIST DOWN LIBRARIES INSTALLED

pip freeze > requirements.txt

INSTALL REQUIRED LIBRARIES

pip install -r requirements.txt

For more info

PYTHON VIRTUAL ENVIRONMENT

WHY USE VIRTUAL ENVIRONMENT (VENV)?

To create an isolated environment for Python projects

Each project can have its own independent dependencies

CREATE PYTHON VENV

python -m venv .venv (.venv can be replaced by other folder names)

ACTIVATE PYTHON VENV

Powershell:

. .venv/Scripts/Activate.ps1

(if there is error, run ”Set-ExecutionPolicy Unrestricted”, not recommended)

Bash:

source .venv/Scripts/activate

(when on Raspberry Pi, it might be in other directory than Scripts, such as bin)

For more info

Demonstration Session

BASH

WHY BASH?

- Swiss army knife to increase your productivity

BASH COMMANDS

sudo - ”superuser do”, run with admin privileges

ls - ”list directory” list all the files in the directory

pwd - ”print working directory”

cd - ”change working directory”

mv - move or rename files or directory

rmdir - ”remove directory”

chmod - set file permission

exit - exit out of directory

Shortcuts

Ctrl+C - terminate the process, keyboard interrupt

Ctrl+Z - suspend the process

For more info

DIGITAL IMAGE

•A representation of a real image as a set of numbers that can be stored and handled by a digital computer

•To translate the image into numbers, it is divided into small areas called pixels (picture elements)

•For example, in the figure below, the digital image is represented by pixels with different RGB data values (Red, Green, Blue)

8 bit color

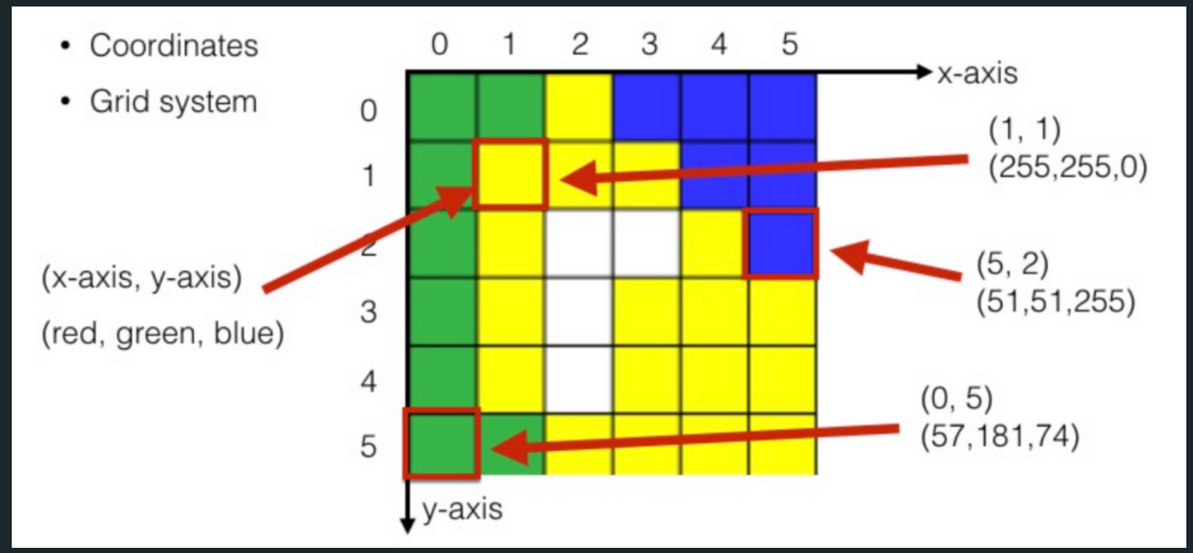
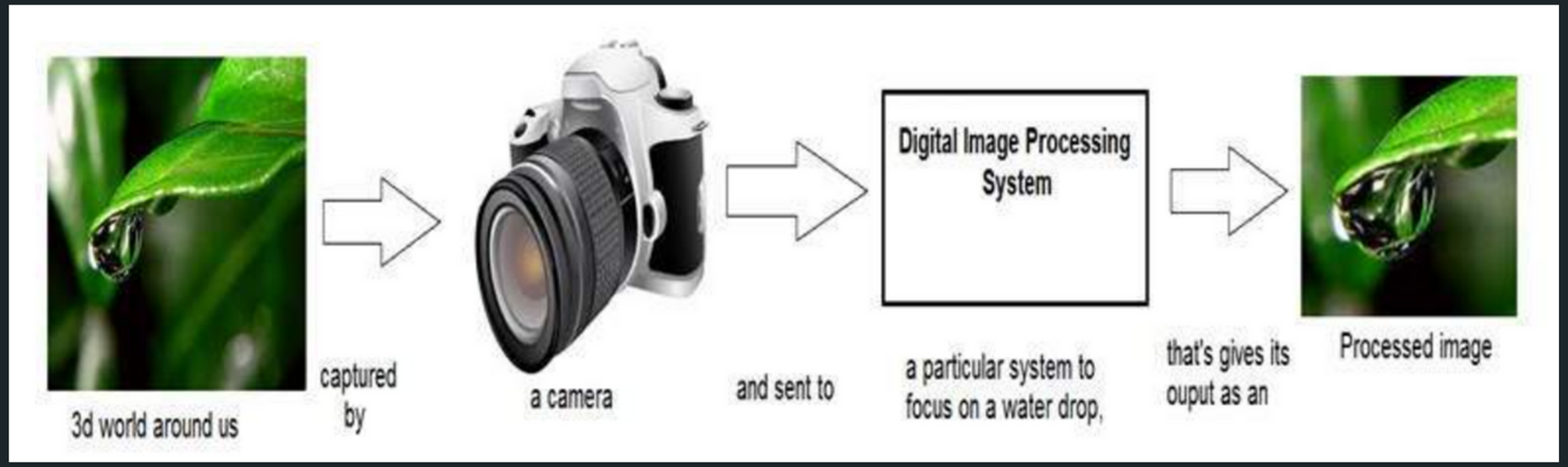


IMAGE PROCESSING

•Image Processing is used to perform some operations on the image to enhance the image or to extract some useful information from it.

•Examples of the application of Image Processing:



OPENCV

OPEN SOURCE COMPUTER VISION

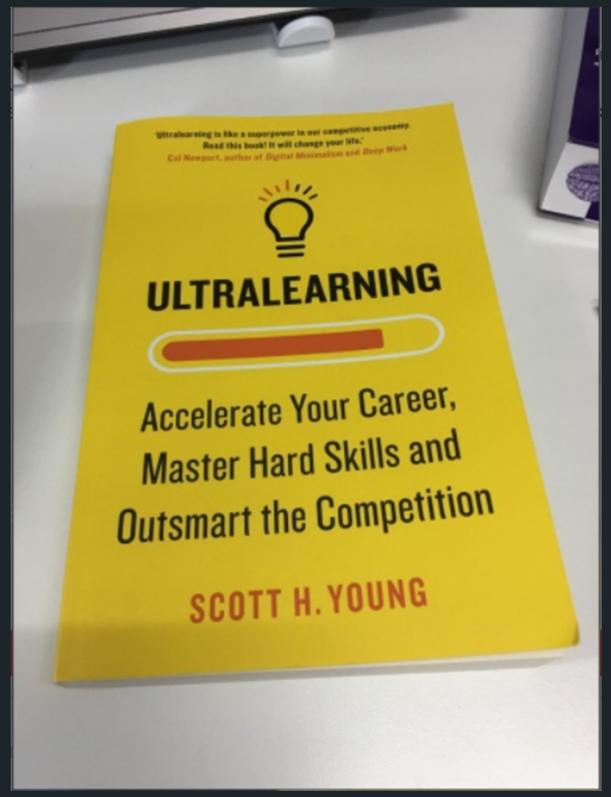
- Library of programming functions mainly aimed for real time computer vision
- Mainly written in C/C++
- opencv-python is a python wrapper around the C/C++ code

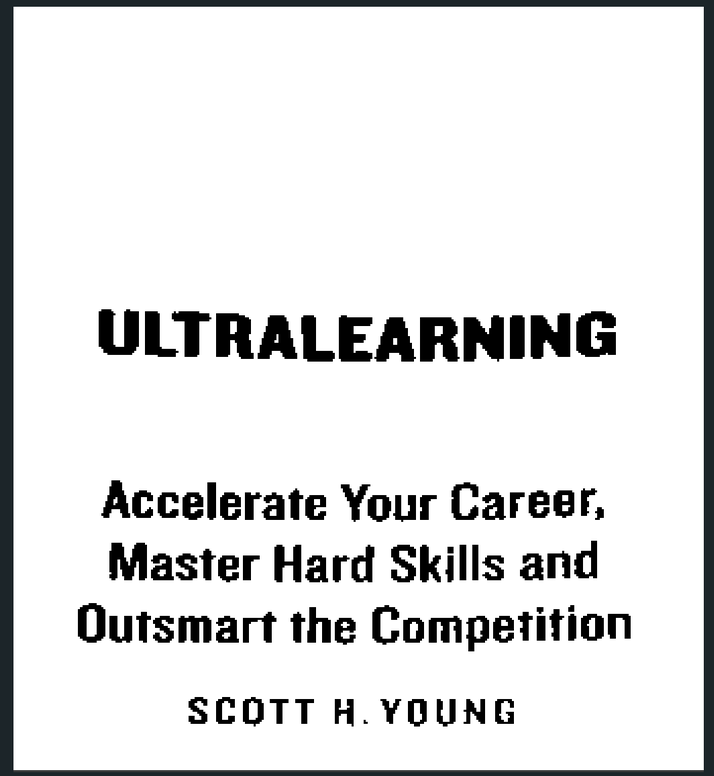
OPENCV

OPEN SOURCE COMPUTER VISION

- Library of programming functions mainly aimed for real time computer vision
- Mainly written in C/C++
- opencv-python is a python wrapper around the C/C++ code

WHY PRE-PROCESSING IS IMPORTANT





Before you bring in the ”big guns” such as Tesseract OCR (Optical Character Recognition),

it’s wiser if you do some preparation to help the program recognize the texts. Hence, it’s important for us to remove unnecessary ”noises” and information to improve the accuracy of the program

OpenCV Coding Session

READ IMAGE USING OPENCV

```
import cv2 as cv

img=cv.imread("book.jpg")

cv.imshow("Book",img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Functions	Function Arguments	Usage	Examples
cv2.imread(path, flag)	Path: Image file path (absolute or relative path**) Flag: Flag that specifies the way image should be read: •cv2.IMREAD_COLOR (default flag) •cv2.IMREAD_GRAYSCALE •cv2.IMREAD_UNCHANGED	Read image	img = cv2.imread('book.jpg',0)

Refer to : https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html

** Absolute or Full path points to the same location in a file system, regardless of the current working directory (root directory needs to be included)
Relative path starts from some given working directory

Example:
(Windows) Absolute path: D:\EEDegree\Robotics Club\Image Processing Workshop\book.jpg
(Windows) Relative path : book.jpg

READ IMAGE USING OPENCV (CONT.)

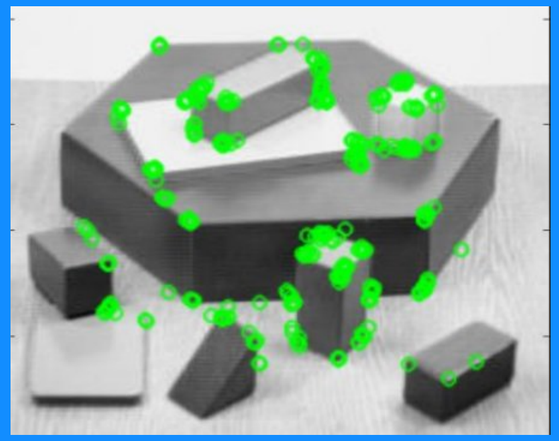
```
import cv2 as cv

img=cv.imread("book.jpg")

cv.imshow("Book",img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Functions	Function Arguments	Usage
cv2.imshow(Window,Image)	Window: Window name (string) Image: Image to be displayed	Display image in a window
cv2.waitKey(Time)	Time: Time in milliseconds. If it's 0, it will wait until a key is pressed	Wait for specified milliseconds until a key is pressed
cv2.destroyAllWindows()		Destroy all windows T_T

FEATURE MATCHING



For more info

```
orb = cv.ORB_create(nfeatures=15000,WTA_K=4)
#create ORB detector object

bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
#create BF Matcher object

kp1, des1 = orb.detectAndCompute(template_img, None)
#find the keypoints and descriptors with ORB
kp2, des2 = orb.detectAndCompute(gray, None)
```

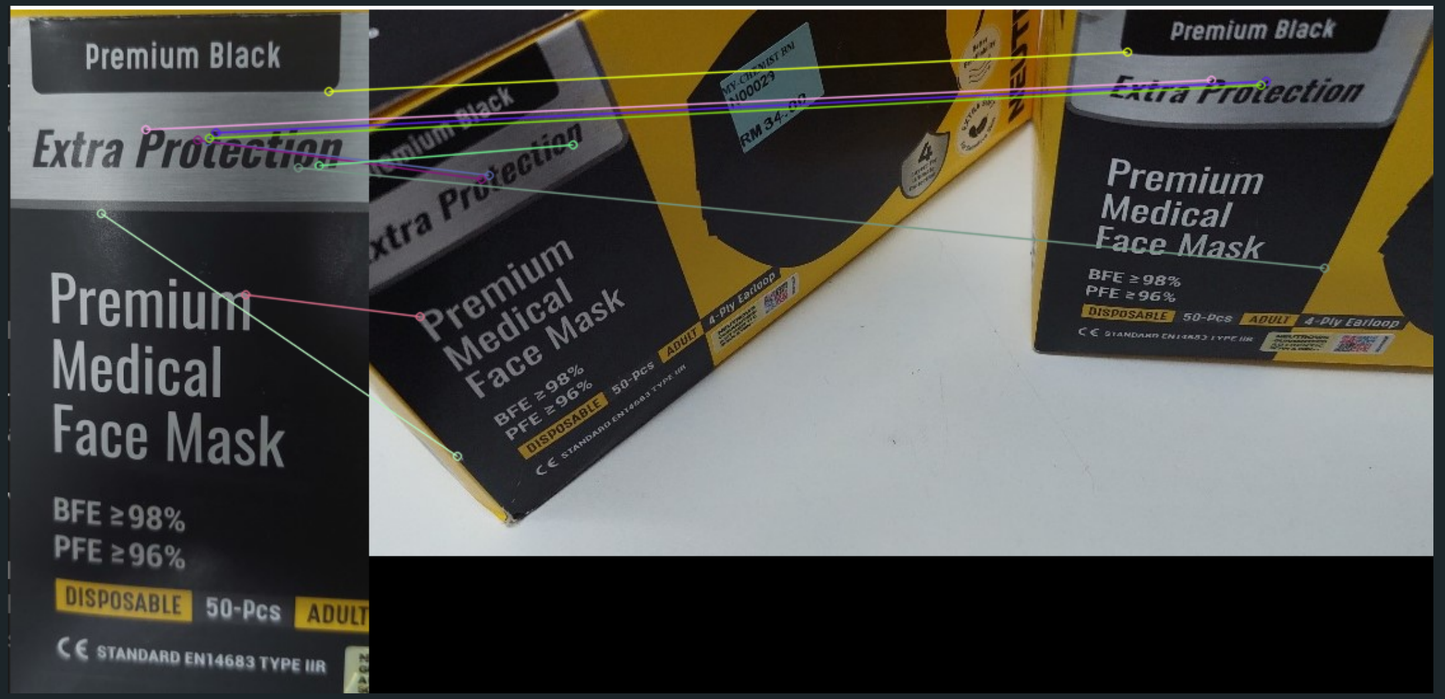
For hardworking people: https://ieeexplore.ieee.org/document/6126544

To put it in the simplest way, one of the way we want to match an image to another even though the image has different lighting, angles, etc, is through Feature Matching.

Basically, through ORB, we can determine the ”good points” of an image
For better performance, you can try SIFT which requires some workarounds to implement

FEATURE MATCHING

```
match = bf.match(des1, des2)
# match the points using Brute Force
match = sorted(match, key=lambda x: x.distance)
# sort the matches by its "distance"
ans=cv.drawMatches(template_img,kp1,src_img,kp2,match
[:10],None,flags=cv.
DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```



For more info

Try it yourself

REFER TO CODE HERE

```
python src/workshop.py -m feature_matching
```

VIDEO CAPTURE THROUGH WEBCAM

```
cap= cv.VideoCapture(0,cv.CAP_DSHOW)
# read video if input is 0, it will capture video from you

    while True:
        # loop through constantly and read each frame
        ret,frame=cap.read()

        if ret:
            gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
# convert image from BGR color space to GRAY color space
```

For more info

HAAR FEATURE-BASED CLASSIFIER

```
face_cascade = cv.CascadeClassifier(
'data/models/haarcascade_frontalface_default.xml')
```

Trained models can be found at here

```
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

for (x, y, w, h) in faces:
    cv.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0
), 2)
```

For more info

Try it yourself

REFER TO CODE HERE

```
python src/workshop.py -m face_detection
```

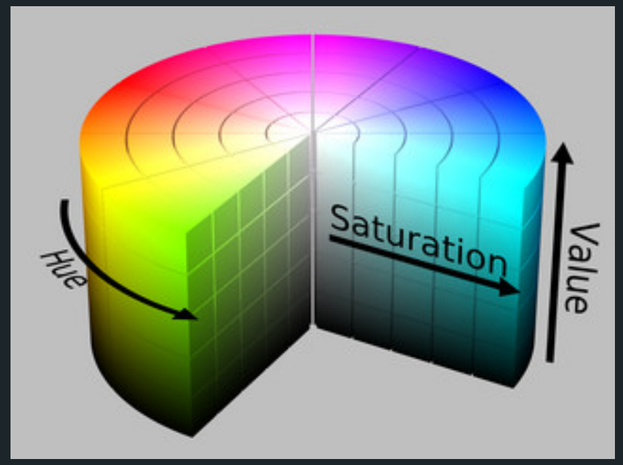
COLOR SPACE

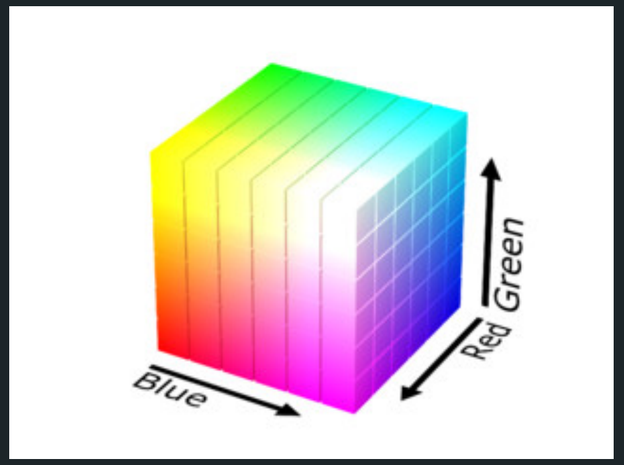
Uses numbers to represent colors

In our use case, we will be using BGR and HSV 8-bit color space

(Remember in OpenCV it's BGR instead of RGB)

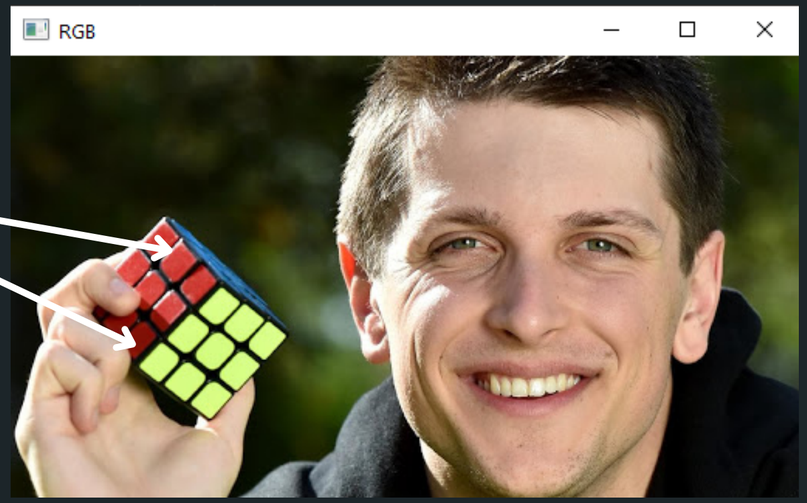
For more info

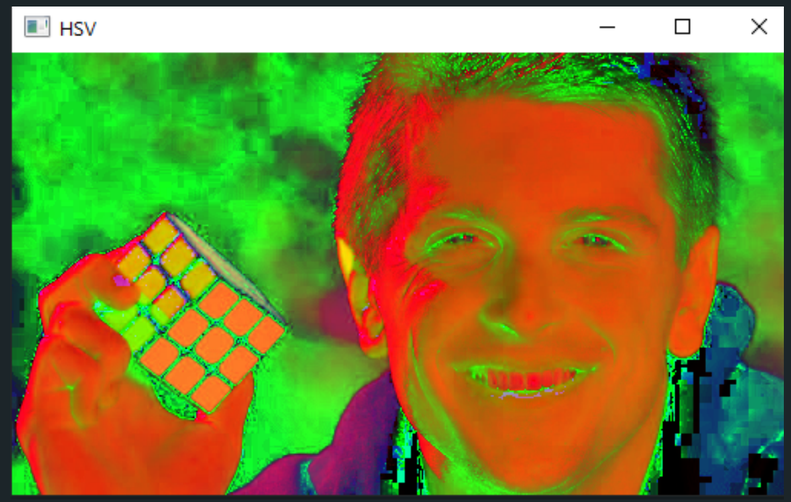




HSV	BGR
Hue, Saturation, Value (3-channel)	Blue, Green, Red (3-channel)
Hue (Color): 0 - 179 Saturation (Intensity): 0-255 Value (Brightness): 0 - 255	Blue: 0 -255 Green: 0 - 255 Red: 0 - 255
More "immune" to different lighting as the hue component which represents primary color is not affected much	All color channels will be influenced by shadows and different illumintation

COLOR SPACE (CONT.)





Refer to code here or
python workshop.py -m hsv --image Rubiks.jpg

Notice the red side of the Rubrik's cube.

The part covered by shadow will have different values for all the 3 channels (Red, Green, and Blue) compared to the lighter part if represented in RGB colour space.

While in HSV, only the Saturation (S) and Value (V) components will be affected, the Hue (H) component will not change much.

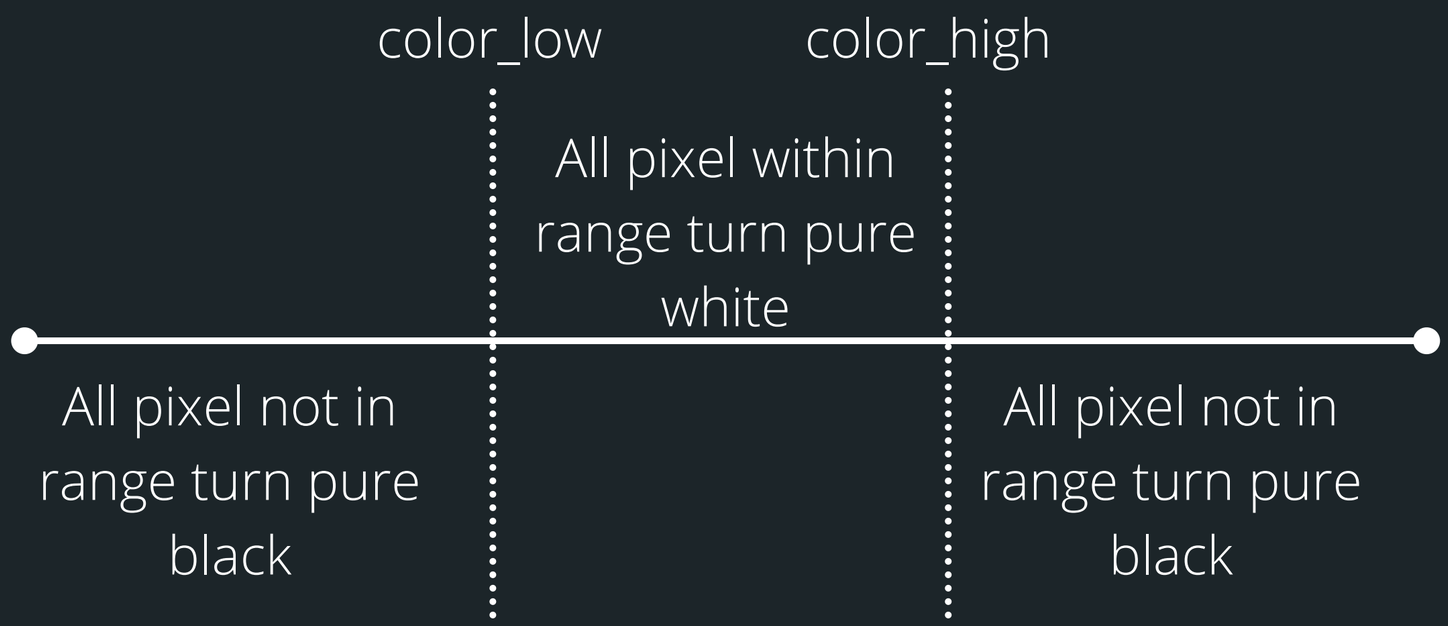
For more info

COLOR SEGMENTATION

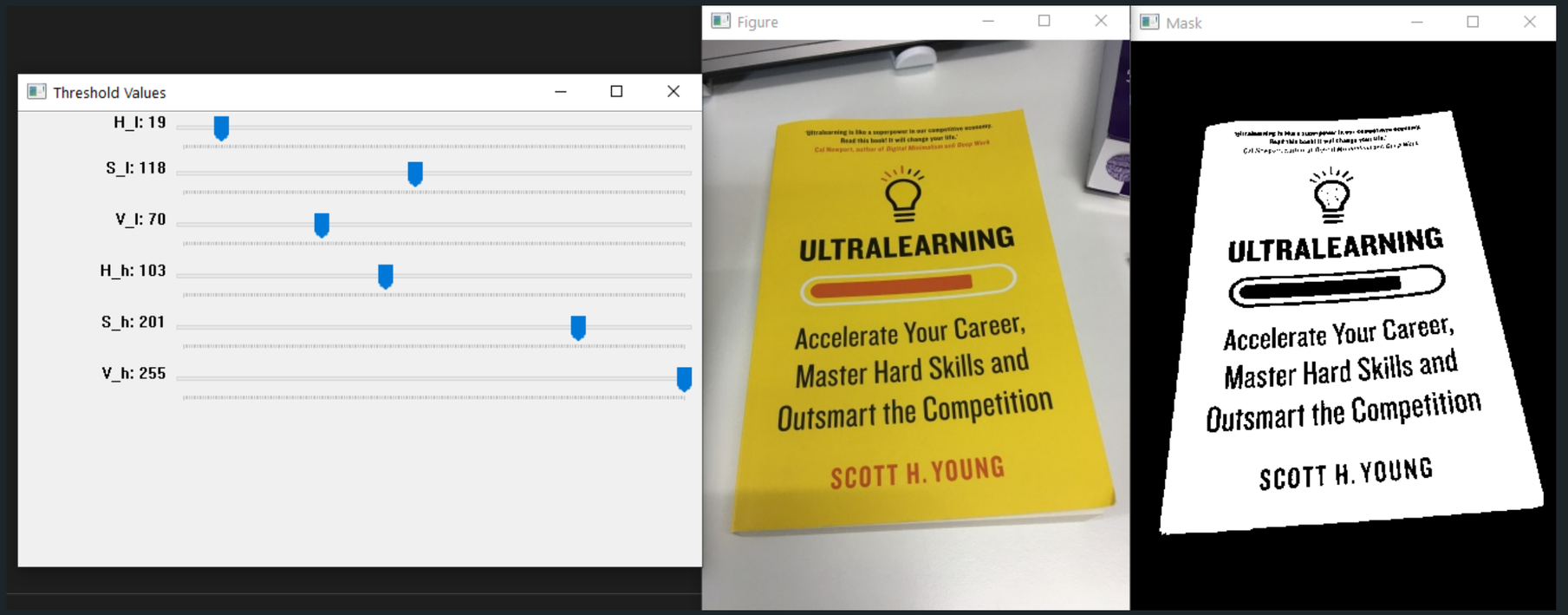
```
color_low = (10,100,0)
color_high= (90, 200, 255)
mask=cv.inRange(img,color_low,color_high)
```

color_low = (low_H_value, low_S_value, low_V_value)

color_high = (high_H_value, high_S_value, high_V_value)



For more info



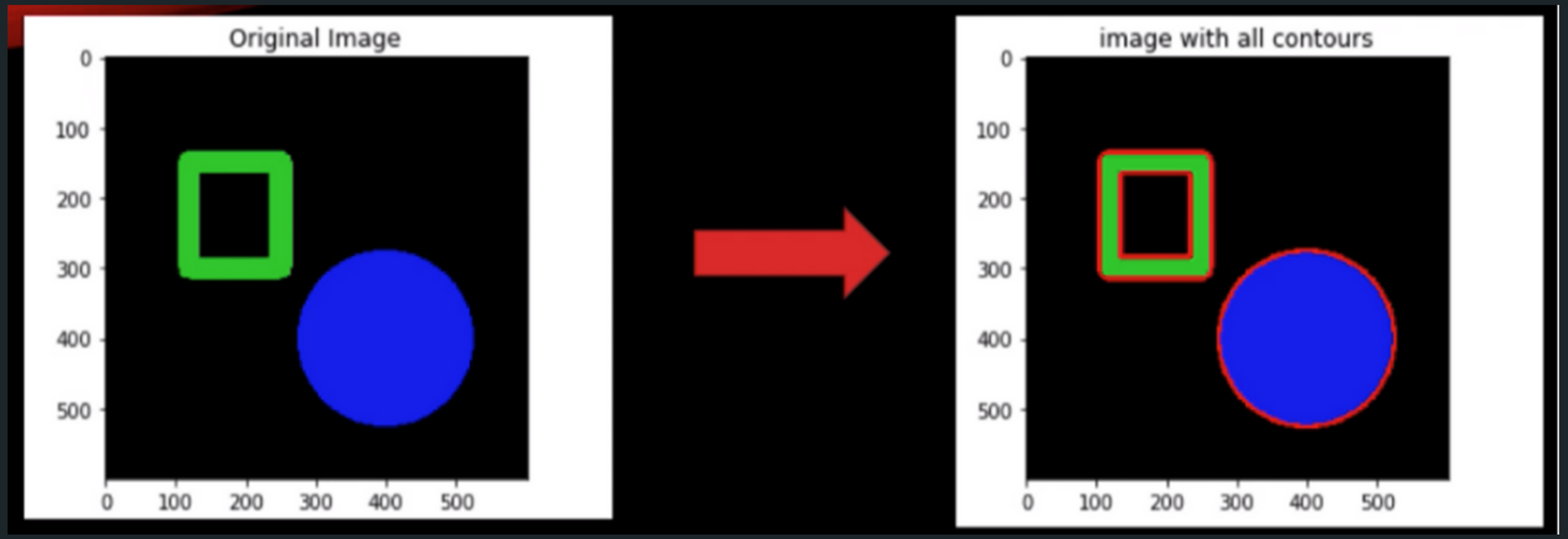
Try it yourself

REFER TO CODE HERE

python src/workshop.py -m trackbar_color

CONTOURS

- Image contouring is a process of identifying structural outlines of objects in an image
- Useful for shape analysis, object detection and recognition
- Contours – a curve joining all the continuous points (along the boundary), having same color or intensity



For more info

CONTOURS (CONT.)

•Before finding contours, need to apply operations such as thresholding or canny edge detection to turn the image into binary images for better accuracy

(In our case, we use color segmentation [inRange] to produce binary image)

•In OpenCV, object to be found should be white and background should be black

```
contours,_=cv.findContours(mask,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_SIMPLE)
```

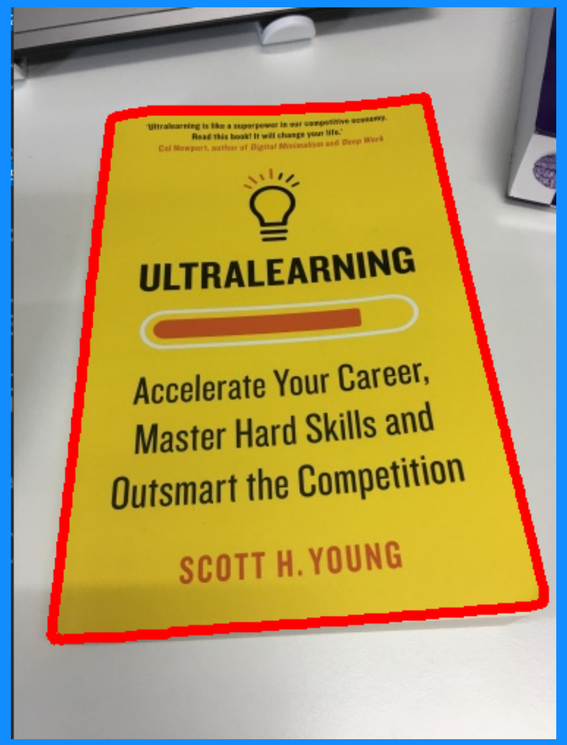
mask - source image, preferably binary / black and white only

cv.RETR_EXTERNAL - retrieval modes, retrieves only extreme outer contours

cv.CHAIN_APPROX_SIMPLE - contour approximation modes

For more info

CONTOURS (CONT.)




```python
cnt=max(contours,key=cv.contourArea)
# find the contour with the largest area
cv.drawContours(img,[cnt],-1,(0,0,255),5)
```

img - source image

[cnt] - contours (must be in a python list)

-1 - index of contours, -1 for drawing all contours

(0,0,255) - color of line of the contour line in BGR

5 - thickness of the line

For more info

PERSPECTIVE TRANSFORM

The contour we obtained is not a perfect rectangle with 4 vertices
We can apply contour approximation to approximate the contour shape to rectangular shape with only 4 vertices

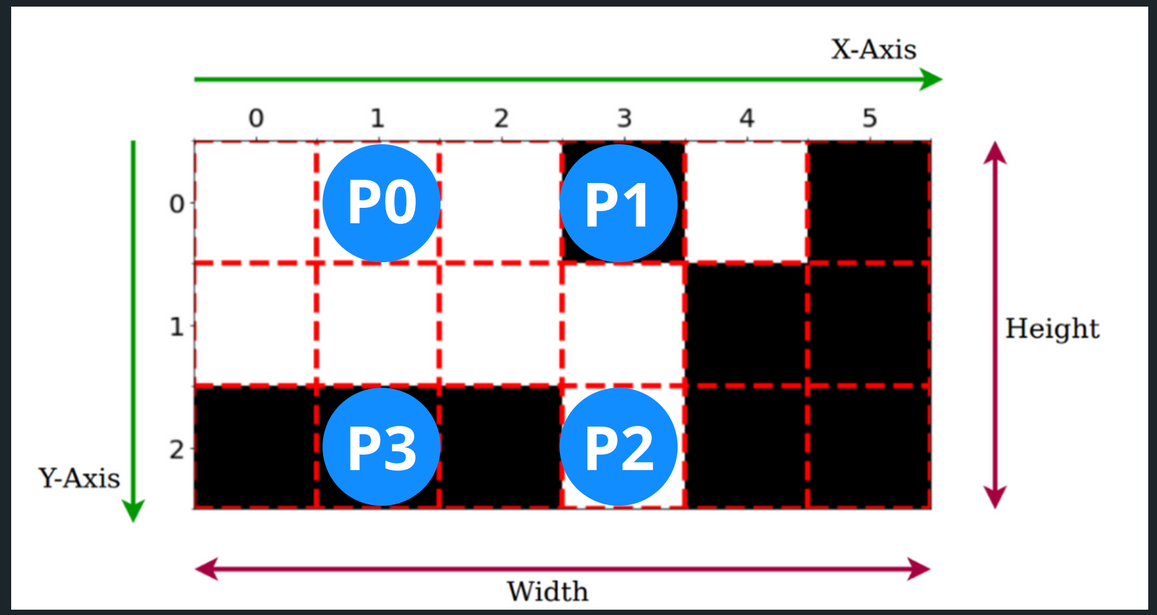
```
epsilon = 0.1*cv.arcLength(cnt, True)
approx = cv.approxPolyDP(cnt, epsilon, True)
```

epsilon (or accuracy) = 10% of book contour perimeter
cv.approxPolyDP approximates a curve with another curve with less vertices
cv.approxPolyDP is an implementation of cv.approxPolyDP of Douglas-Peucker algorithm

For more info

Douglas-Peucker algorithm animation

PERSPECTIVE TRANSFORM (CONT.)



After obtaining the 4 vertices, we need to determine which point is top-left, top-right, bottom-left, bottom-right

Sum its own coordinates' x and y

P0:1+0=1 ; P1: 3+0=3 ; P2: 3+2=5 ; P3: 1+2=3

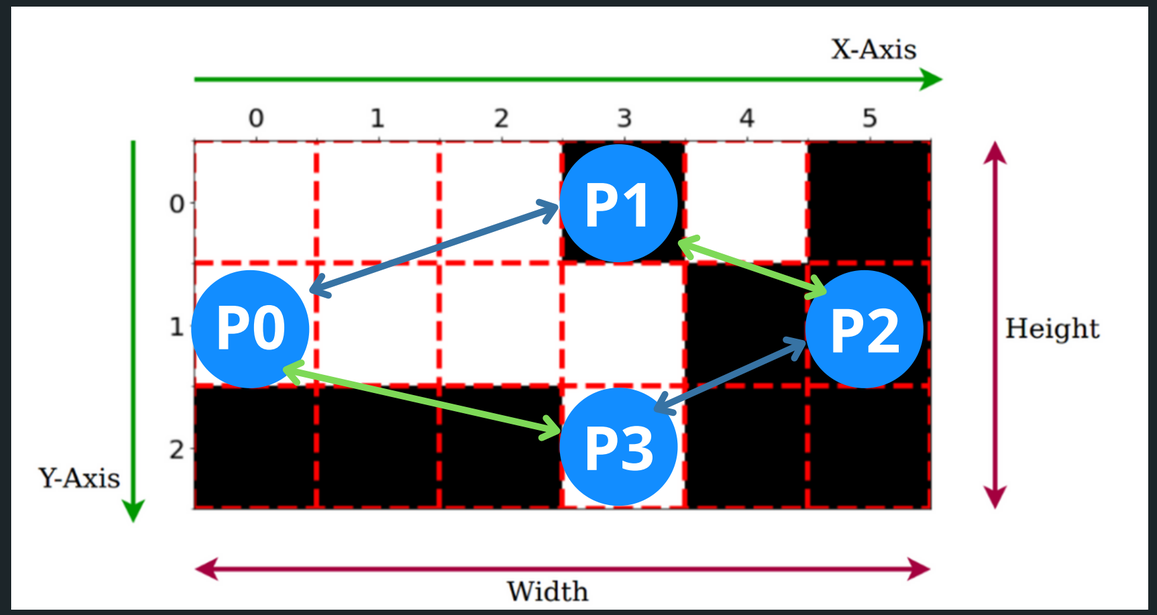
Minus its own coordinates' y and x

P0:0-1=-1 ; P1: 0-3=-3 ; P2: 2-3=-1 ; P3: 2-1=1

Top left smallest sum (P0); bottom right biggest sum (P2)

Top right smallest difference (P1); bottom left biggest difference (P3)

PERSPECTIVE TRANSFORM (CONT.)



width is distance between top left and top right

another width is distance between bottom left and bottom right

height...

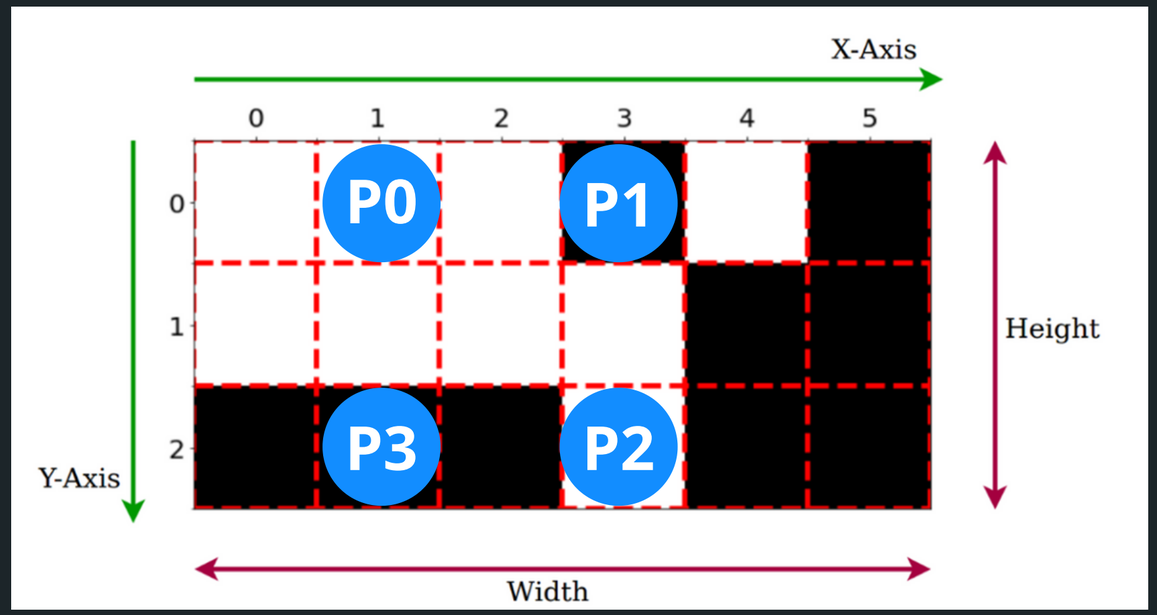
```
widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
```

[0] is for x coordinate, [1] is for y coordinate

Find the distance using Pythogoras Theorem

*br=[3,0] ; so x coordinate = br[0] = 3 ; y coordinate = br[1] = 0

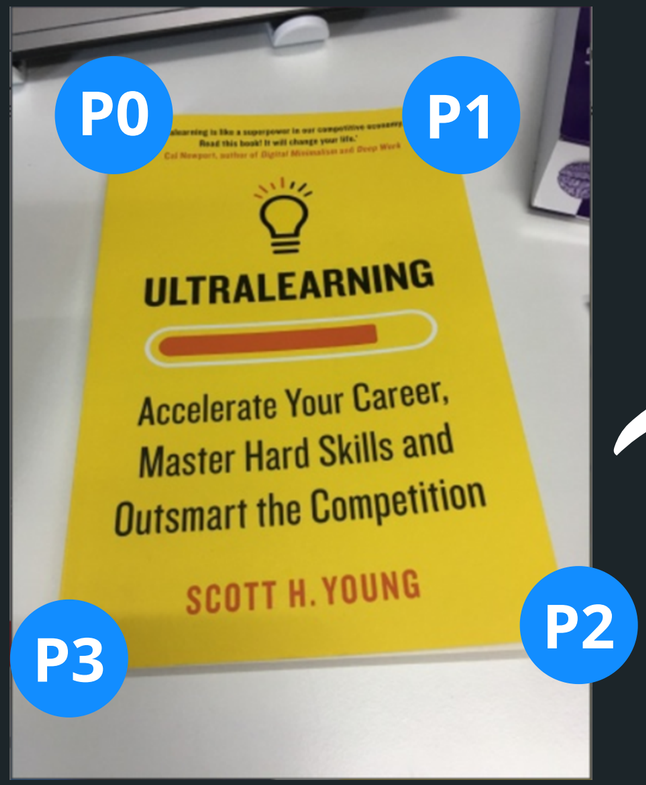
PERSPECTIVE TRANSFORM (CONT.)

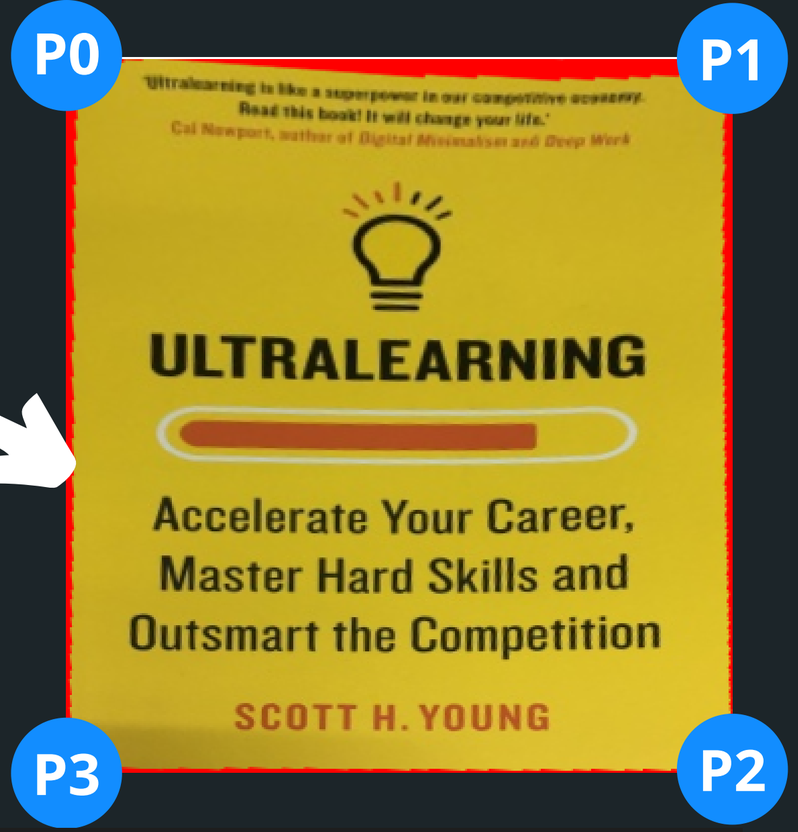


Then, we will create another blank image with a specified dimension so that the warp image can fit into it

P0: (1,0)
P1: (Max Width-1,0)
P2: (Max Width-1, Max Height-1)
P3: (0, Max Height -1)

PERSPECTIVE TRANSFORM (CONT.)






```
M = cv.getPerspectiveTransform(rect, dst)
warp = cv.warpPerspective(img, M,(maxWidth, maxHeight))
```

Original Points (after arranged):

rect=

[[66,76],
[280,62],
[361,405],
[28,431]]

Destination points:

dst=

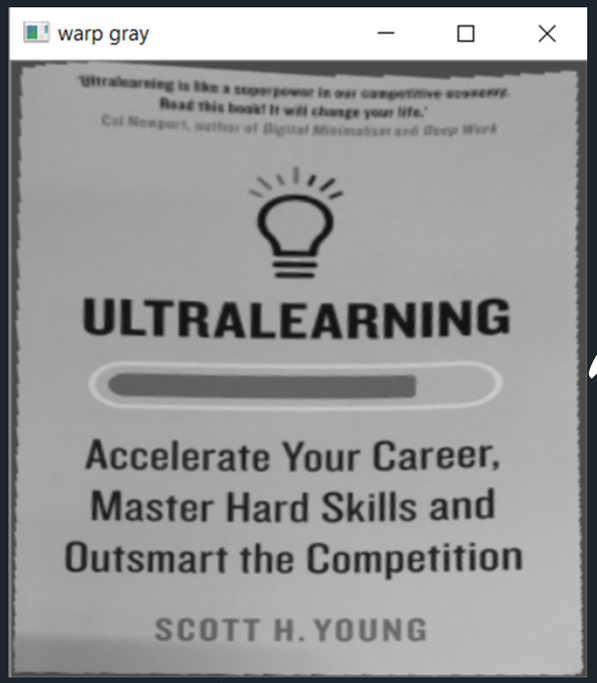
[[0,0],
[333,0],
[333,356],
[0,356]]

For more info

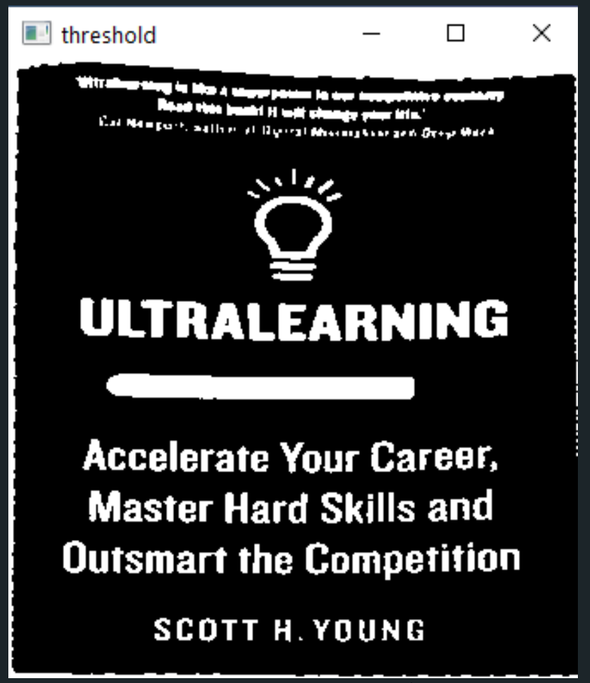
THRESHOLDING

Binary inverse because we will be doing another contouring later which object to be found should be in white and background in black

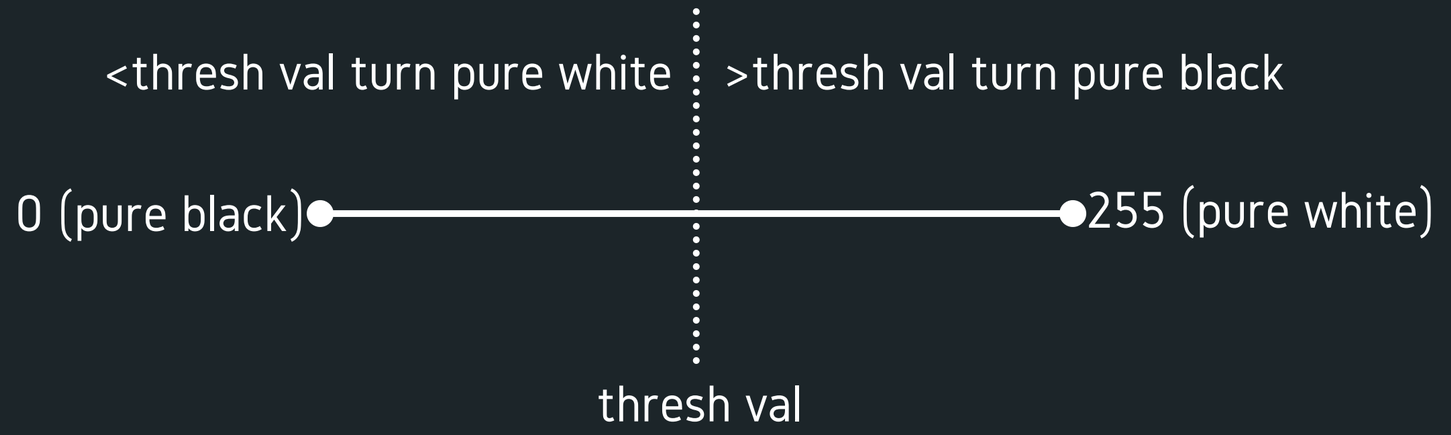
For more info



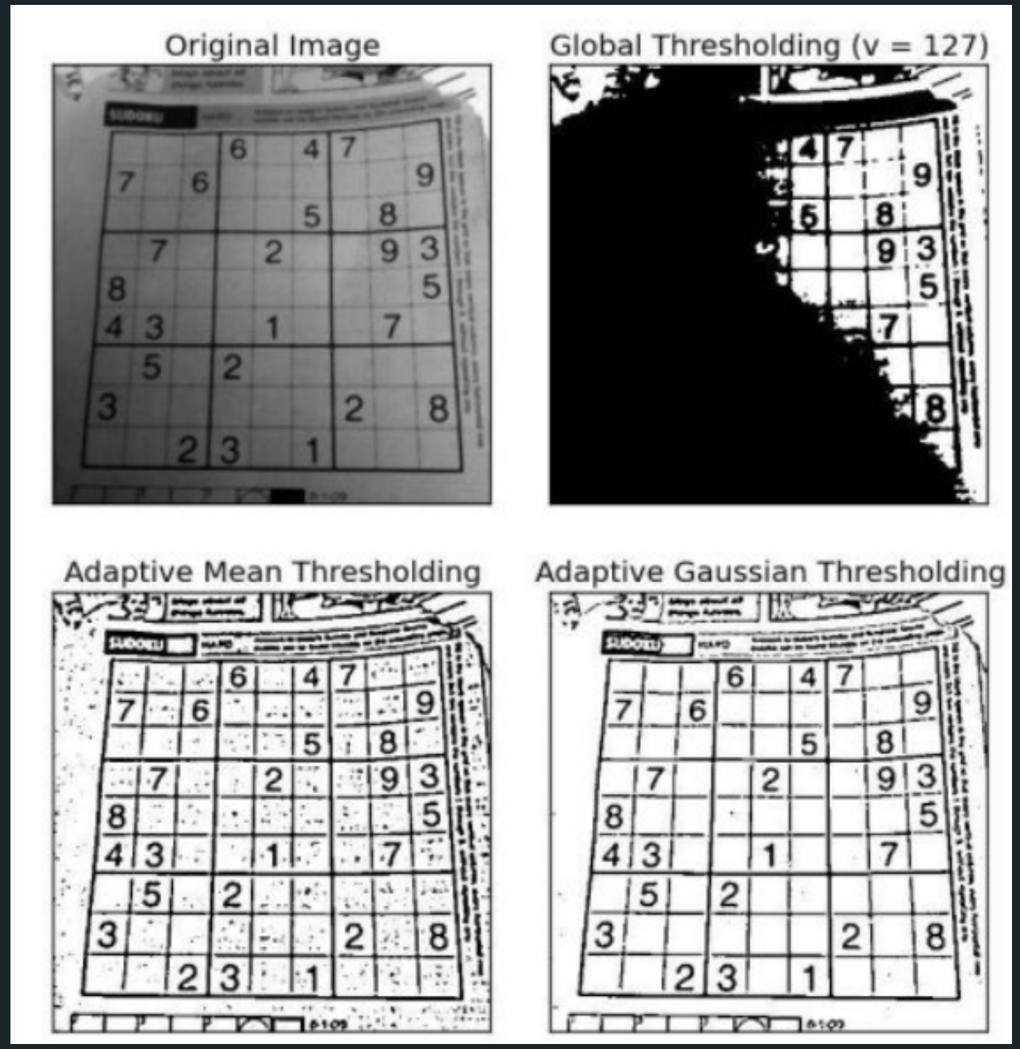
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)



_, threshold = cv.threshold(gray,thresh_val, 255, cv.THRESH_BINARY_INV)



ADAPTIVE THRESHOLDING

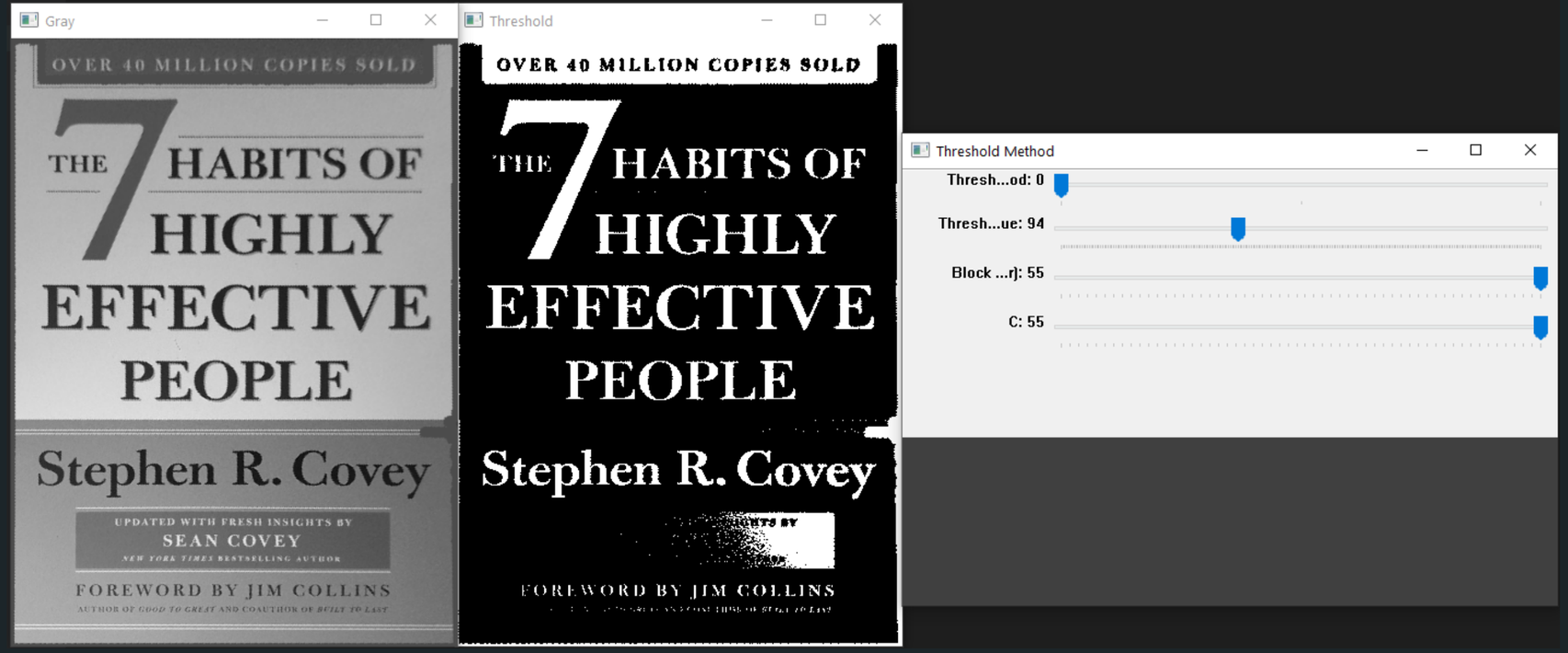


if an image has different lighting conditions in different areas, global thresholding will produce terrible results

- cv.ADAPTIVE_THRESH_MEAN_C: The threshold value is the mean of the neighbourhood area minus the constant C.
- cv.ADAPTIVE_THRESH_GAUSSIAN_C: The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant C.

block size can only be odd numbers

For more info

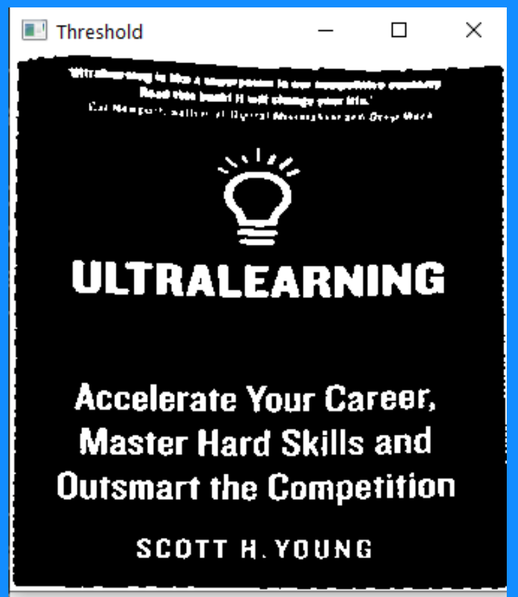


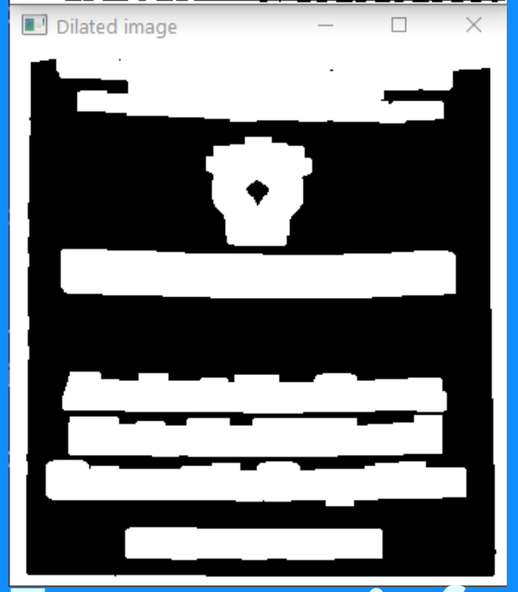
Try it yourself

REFER TO CODE HERE

python src/workshop.py -m trackbar_threshold

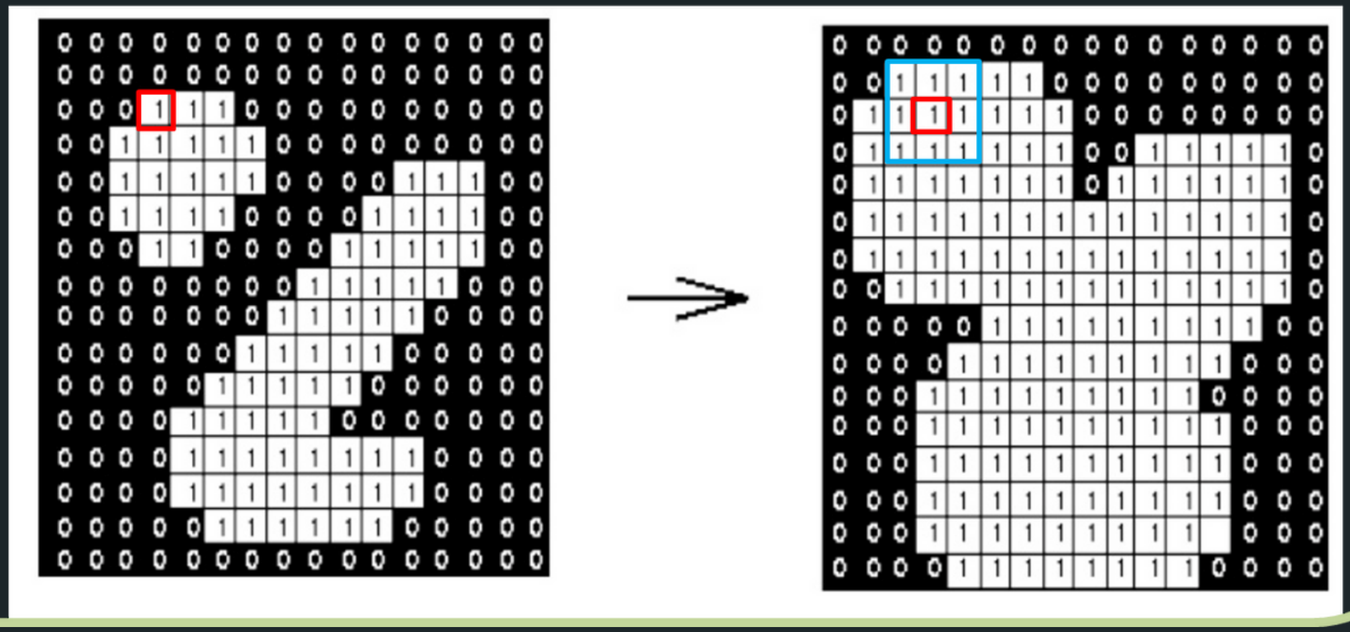
DILATION





For more info

Explaining Dilation Concept using a 3x3 square kernel:

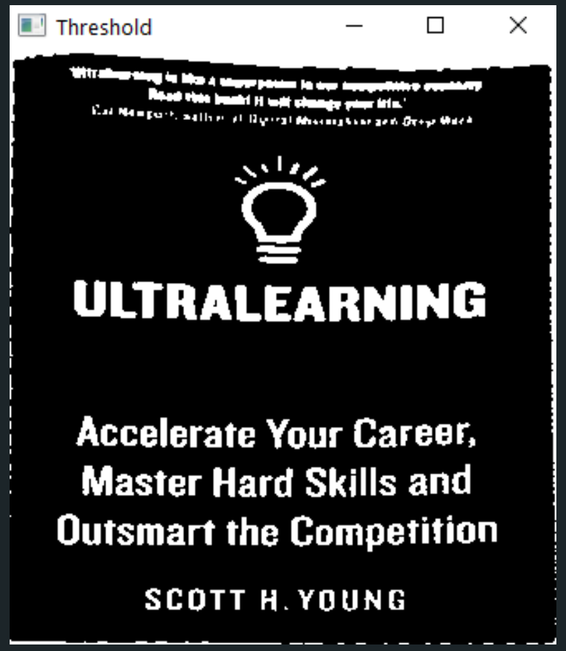


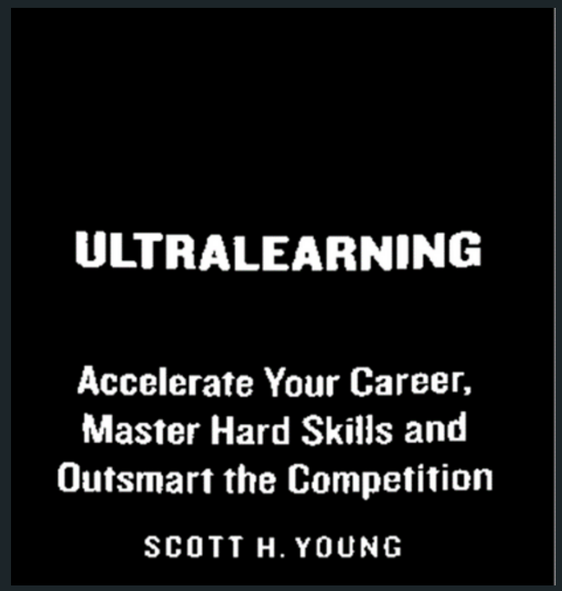
Joining the characters together into horizontal text bars through dilation

```
kernel = cv.getStructuringElement(cv.MORPH_RECT, (5, 3))
dilate = cv.dilate(threshold, kernel, iterations=4)
```

Then, using contours remove other shapes except for rectangular shape

BITWISE OPERATION





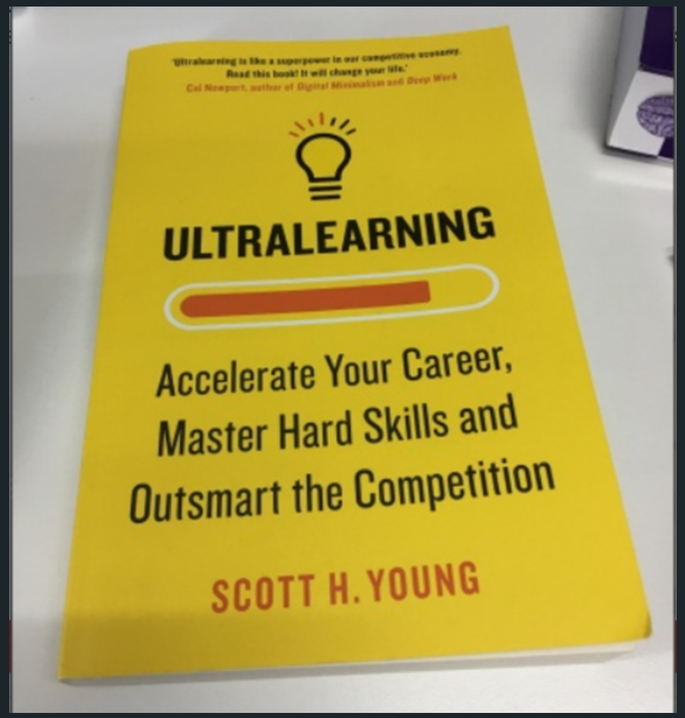
[For more info]

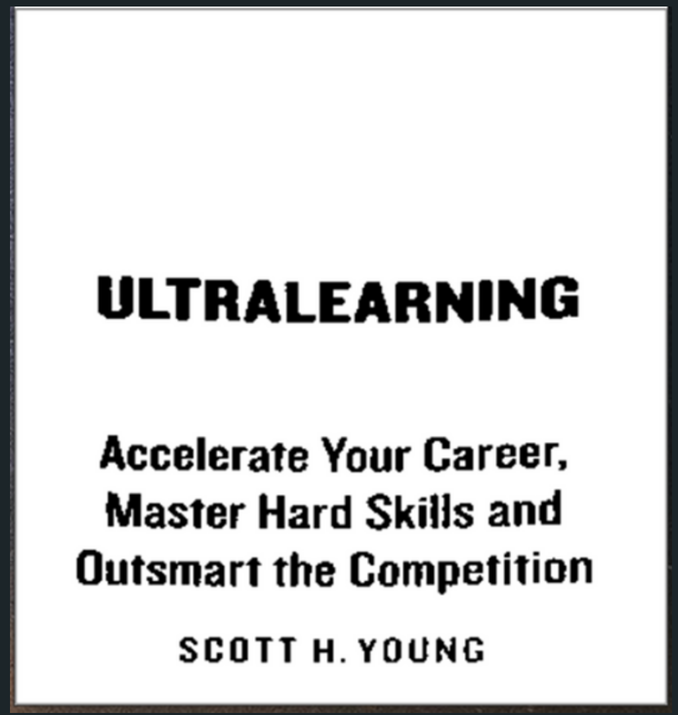
BITWISE OPERATION

result = cv.bitwise_not(cv.bitwise_and(black_mask, threshold))

For more info

COMPARISON


'Ultralearning is like a superpower in our competitive economy.
Read this book! It will change your life.'
Cal Newport, author of Digital Minimalism and Deep Work
ULTRALEARNING
Accelerate Your Career,
Master Hard Skills and
Outsmart the Competition
SCOTT H. YOUNG


ULTRALEARNING
Accelerate Your Career,
Master Hard Skills and
Outsmart the Competition
SCOTT H. YOUNG

Try it yourself

REFER TO CODE HERE

python src/workshop.py

TESSERACT

TESSERACT OCR

- Optical character recognition engine

PAGE SEGMENTATION MODE

config=r'-l eng --psm 3'

- Page segmentation is the process by which a scanned page is divided into columns and blocks which are then classified as graphics, or text.

- PSM affects how Tesseract splits image in lines of text and words.

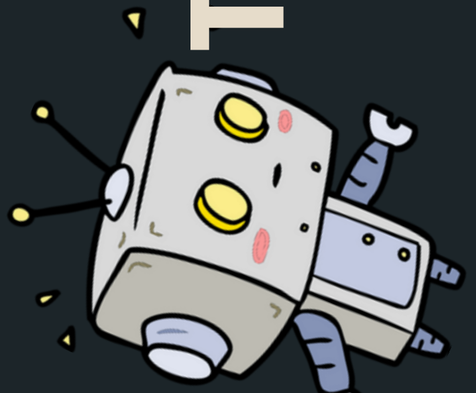
```
 0    Orientation and script detection (OSD) only.
 1    Automatic page segmentation with OSD.
 2    Automatic page segmentation, but no OSD, or OCR.
 3    Fully automatic page segmentation, but no OSD. (Default)
 4    Assume a single column of text of variable sizes.
 5    Assume a single uniform block of vertically aligned text.
 6    Assume a single uniform block of text.
 7    Treat the image as a single text line.
 8    Treat the image as a single word.
 9    Treat the image as a single word in a circle.
10    Treat the image as a single character.
11    Sparse text. Find as much text as possible in no particular order.
12    Sparse text with OSD.
13    Raw line. Treat the image as a single text line,
                    bypassing hacks that are Tesseract-specific.
```

Photo Session

“

QnA

THANK YOU



Follow us at

FB.COM/UNMCROBOTICS/

ROBOTICS_UNM