# **SI100B Project** Face Detection and emotion classification

Lecture 2

**Objectives**

- Familiarize the basic functions in OpenCV.
- Mastering the Use of Cascade Classifiers in OpenCV.

## 1 Introduction

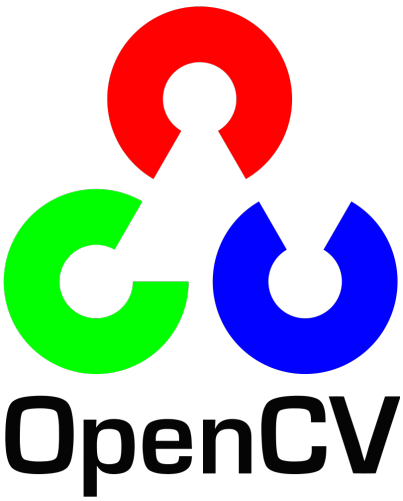


Figure 1: OpenCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly for real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). It was developed to provide a unified infrastructure for computer vision applications and to accelerate the usage of machine perception in commercial products. The library contains over 2,500 optimized algorithms, which can be used for various tasks in image processing and computer vision.

You can't memorize all the APIs in OpenCV, but an essential skill when using OpenCV is knowing how to look up the official documentation. Please use this link OpenCV to learn more about this library.

## 2 Basic functions

Read image from local:

```
import cv2
img = cv2.imread('/path/to/image')
print(img.shape)
```

**Notes**

You can use a relative path only if your files are located in the same directory (or subdirectory) as your working directory. Otherwise, you'll need to use an absolute path to access them.

```
#relative path
img = cv2.imread("img/demo.png")   #windows linx mac
img = cv2.imread(r"img\demo.png")
img = cv2.imread("img\\demo.png")
#absolute path
img = cv2.imread("~/demo.png")  #linux mac
img = cv2.imread("C:/Users/zhenghao/Documents/SI00BProject/demo.png")
img = cv2.imread(r"C:\Users\zhenghao\Documents\SI00BProject\demo.png")
img = cv2.imread("C:\\Users\\zhenghao\\Documents\\SI00BProject\\demo.png")
```

Show image:

```
cv2.imshow("src", img)
cv2.waitKey(0)
```

Draw rectangle on image:

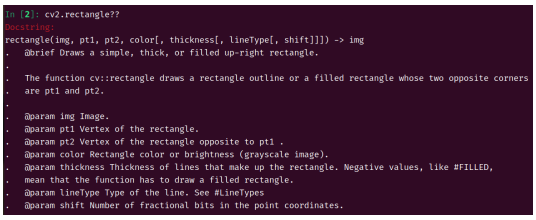


Figure 2: Draw rectangle

Write words on image:

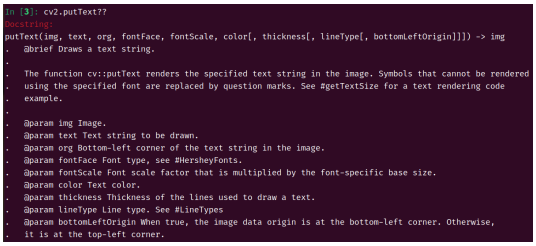


Figure 3: put text

Cropping image: Image is stored as 2D-array in computer. Use array slicing to obtain a portion of the image.
Resize:

```
Help on built-in function resize:

resize(...)
    resize(src, dsize[, dst[, fx[, fy[, interpolation]]]]) -> dst
    .   @brief Resizes an image.
    .
```

Figure 4: resize

Write to local:

```
cv2.imwrite("result.png", newImg)
```

Follow the steps below and do it step by step.

1. Read the image you generated last class.
2. Draw a rectangle on the image, the top left corner is (80, 60) and the width is 480, the height 360
3. Cropped the image $x \in [40, 600], y \in [30, 450]$
4. Reduce both the width and height of the image to half of their original size.
5. Put the text "I love SI100B" on the center of the image
6. Write it to the local.

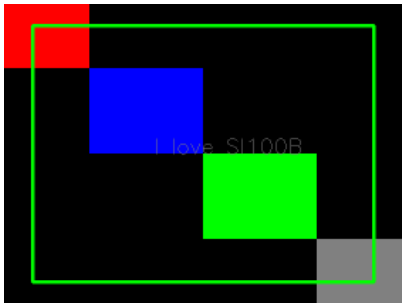


Figure 5: OpenCV

# 3 Cascade Classifier

The cascade classifier is a machine learning based approach where a cascade function is trained from a lot of positive and negative images.
It operates as a sequence of increasingly complex classifiers, arranged in a cascade structure, to efficiently determine the presence of an object in an image. How it works?

## 3.1 Features

The cascade classifier often use Haar features to identify specific patterns in the image. Haar features are patterns or rectangles used to detect specific contrasts in an image. These can include:

- edges features
- line features
- four-rectangles features

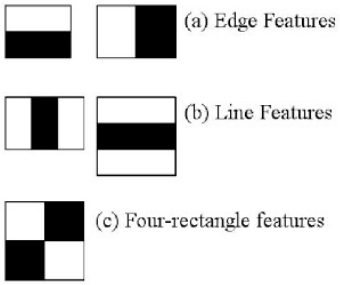


Figure 6: Haar Features

This is an eample to show how haar features works:

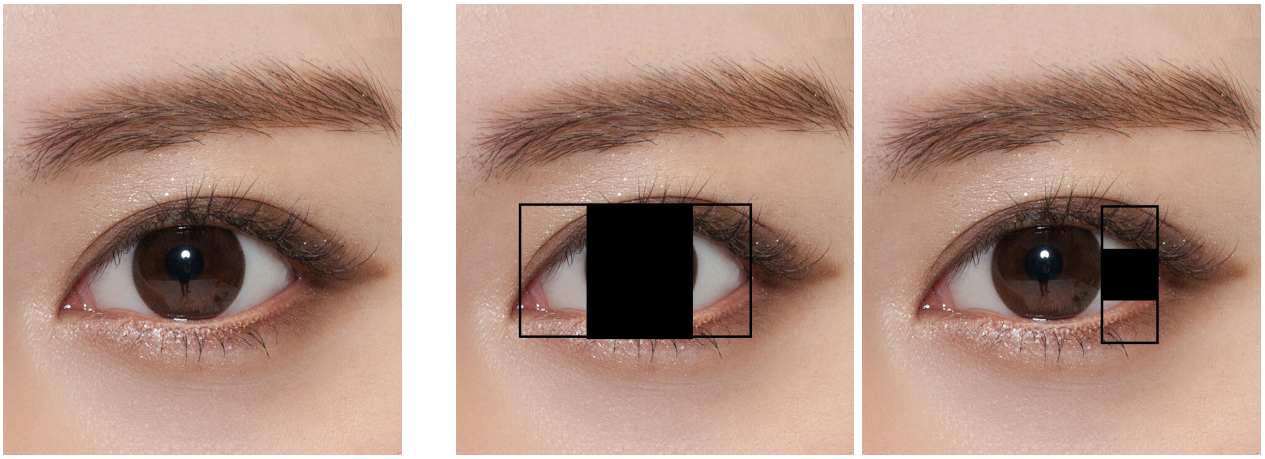

Figure 7: eye features

## 3.2 Training

The Adaboost algorithm is commonly used to select the most important features and build a strong classifier from weak learners.

## 3.3 Detection

- The classifier scans the input image at multiple scales and positions
- The classifier is organized as a series of stages, where each stage applies a simple and fast test to the input. If the input passes, it moves to the next stage; if not, it is immediately rejected.]
- Early stages focus on rejecting non-object areas quickly with minimal computation.
- Later stages are more complex, aiming to confirm positive detections with greater accuracy.

### 3.4 Test

```python
import cv2

# Load pre-trained cascade file
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Read image and convert to grayscale
image = cv2.imread('/path/to/your/image')
# Convert to gray image
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

print(faces)
# Draw rectangles around detected faces
# Display the result
```

You will see the output info as following:

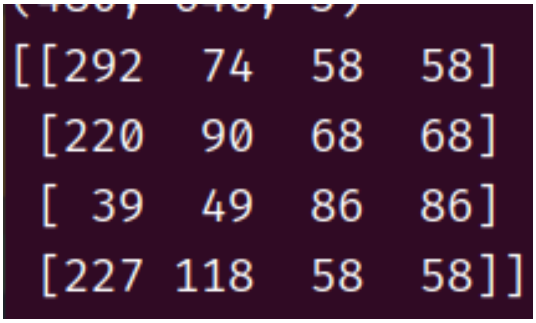


Figure 8: output

The four numbers mean: [ topleft-x, topleft-y, width, height ]

Take a frontal photo of yourself or your group members, use the program to detect the location of the faces, and then outline them with rectangles.

There are lots of models that can detect other object, try them. Models

### 3.5 Bonus

1. Can you detect eyes from the image and draw eyes with circle. (1 point)
2. Implement an algorithm to eliminate overlapping parts. (1 point) HINT: NMS


Figure 9: bonus

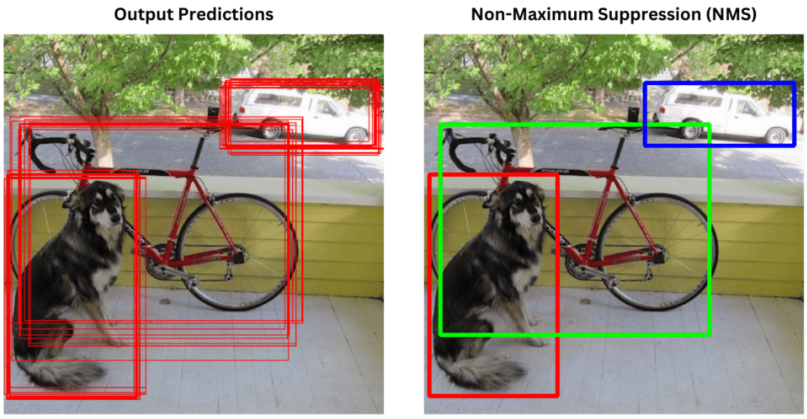


Figure 10: example