KMITL BEYOND THE LIMIT

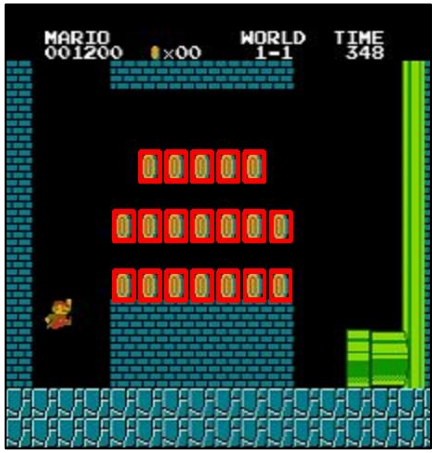
COMPUTER VISIONS

01416500
Course Code

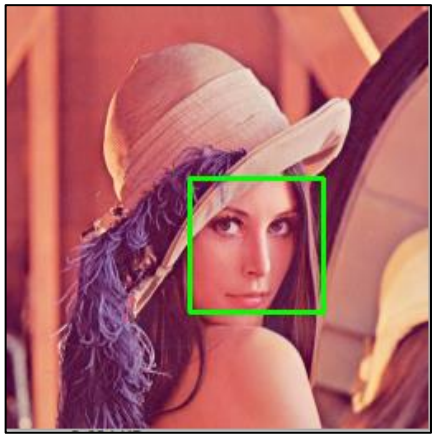
Image Recognition

Overview

1. Template Matching



2. Face Detection



Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang

1. Template Matching

What is template matching?

Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch).

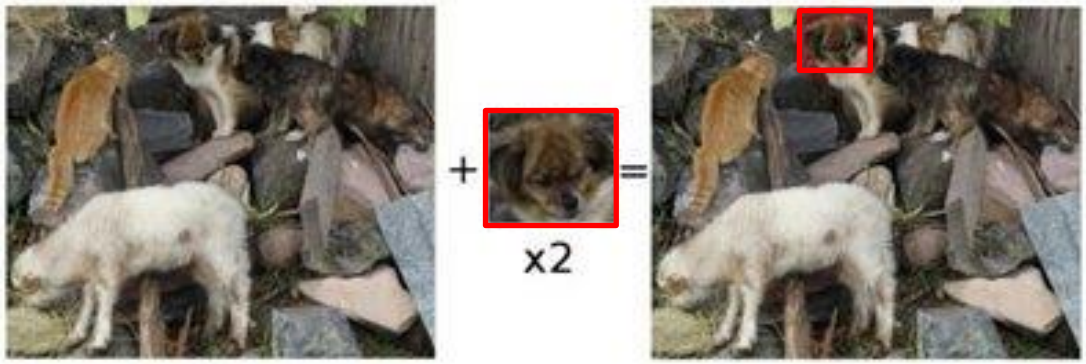
We need ***two*** *primary components*:

I. **Source image (I):** The image in which we expect to find a match to the template image

II. **Template image (T):** The patch image which will be compared to the template image

1. Template Matching

Our goal is to detect the highest matching area:



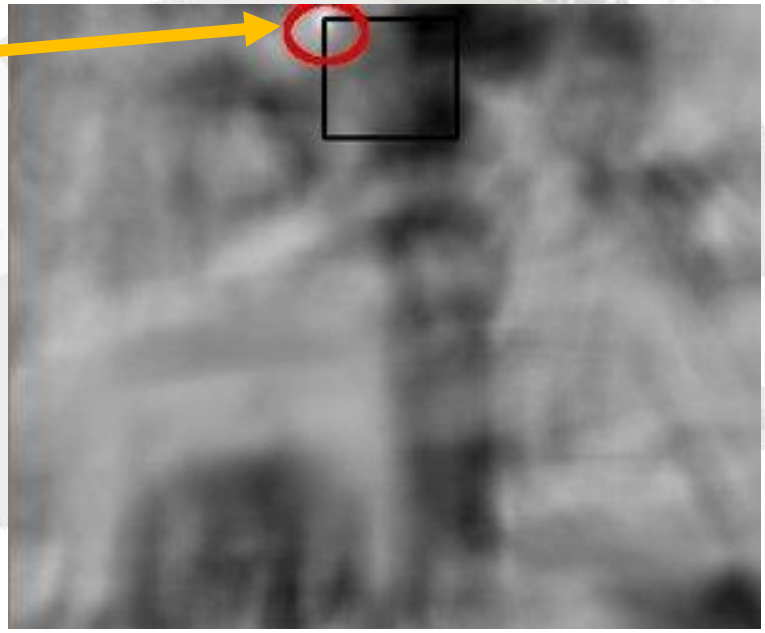
By **sliding**, we mean moving the patch one pixel at a time (left to right, up to down).

To identify the ***matching area***, we have to *compare* the template image against the source image by sliding it:

1. Template Matching

- **Several comparison methods** are implemented in OpenCV,
- It returns a grayscale image, where each pixel denotes how much does the neighborhood of that pixel match with template.

Some location maximum result



For each location of T over I, you store the metric in the result matrix (R). Each location (x,y) in R contains the match metric:

1. Template Matching

Which are the matching methods available in OpenCV

1. **method=CV_TM_SQDIFF**

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \quad \text{(Min)}$$

2. **method=CV_TM_SQDIFF_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad \text{(Min)}$$

3. **method=CV_TM_CCORR**

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')) \quad \text{(Max)}$$

1. Template Matching

4. method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$ **(Max)**

5. method=CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))$$ **(Max)**

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

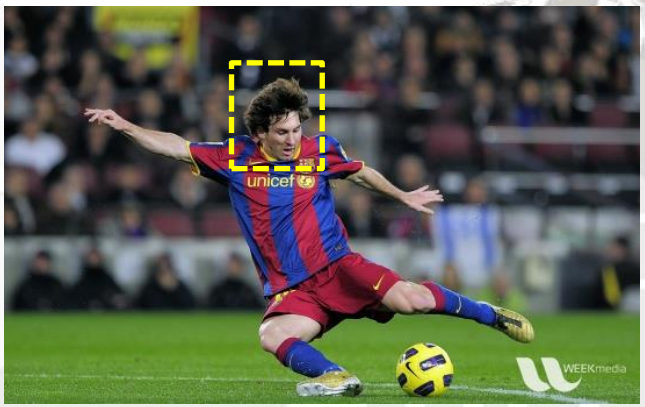
6. method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$ **(Max)**

1. Template Matching

Here, as an example, we will search for Messi's face in his photo.

So if created **a template** as below:



We will try all the comparison methods so that we can see how their results look like:

1. Template Matching

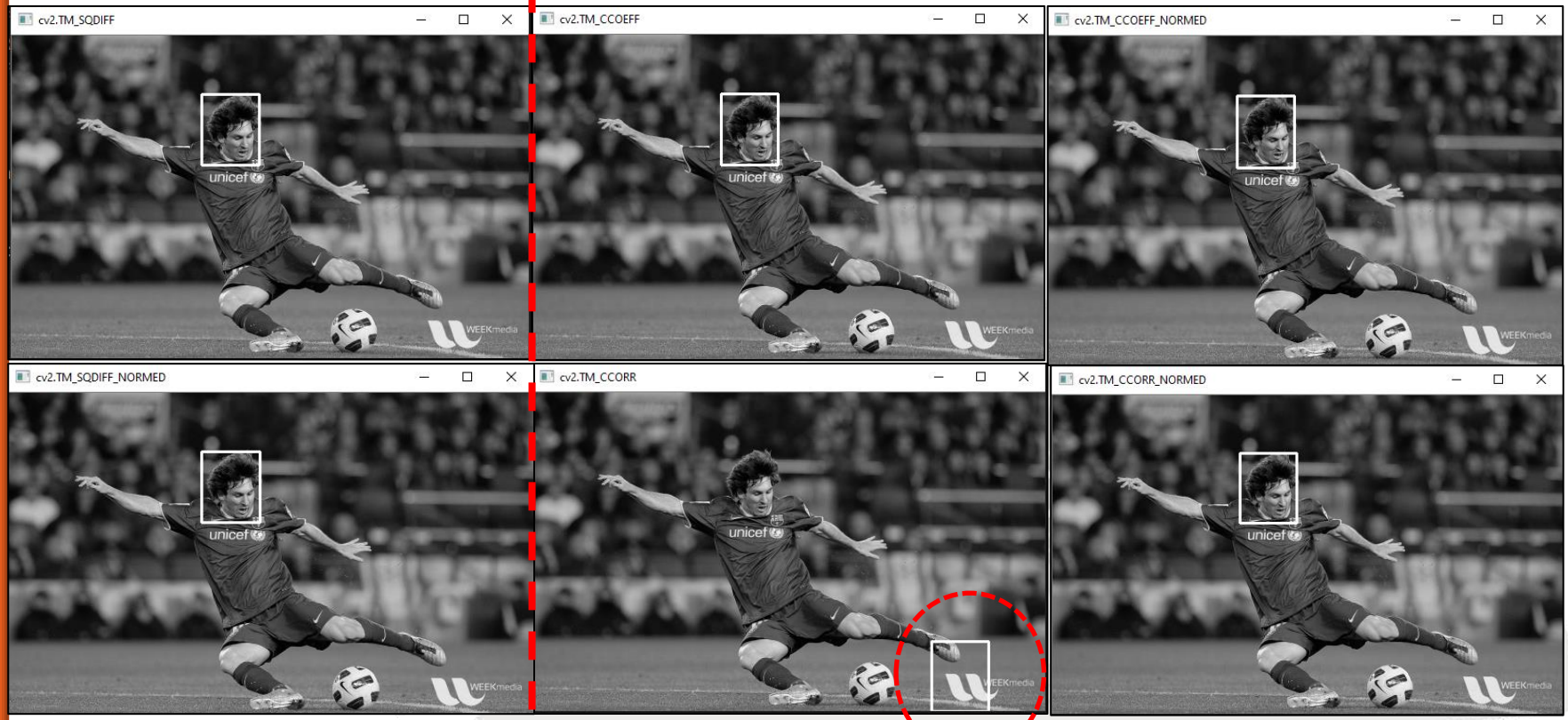
OpenCV function :

cv2.matchTemplate()

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('dataset/messi.jpg',0)
template = cv2.imread('dataset/template.jpg',0)
w= template.shape[1]
h= template.shape[0]

method = cv2.TM_SQDIFF

res = cv2.matchTemplate(img,template,method)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

top_left = min_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img,top_left, bottom_right, 255, 2)

# cv2.imshow("Gray",res)
cv2.imshow("cv2.TM_SQDIFF",img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Template image

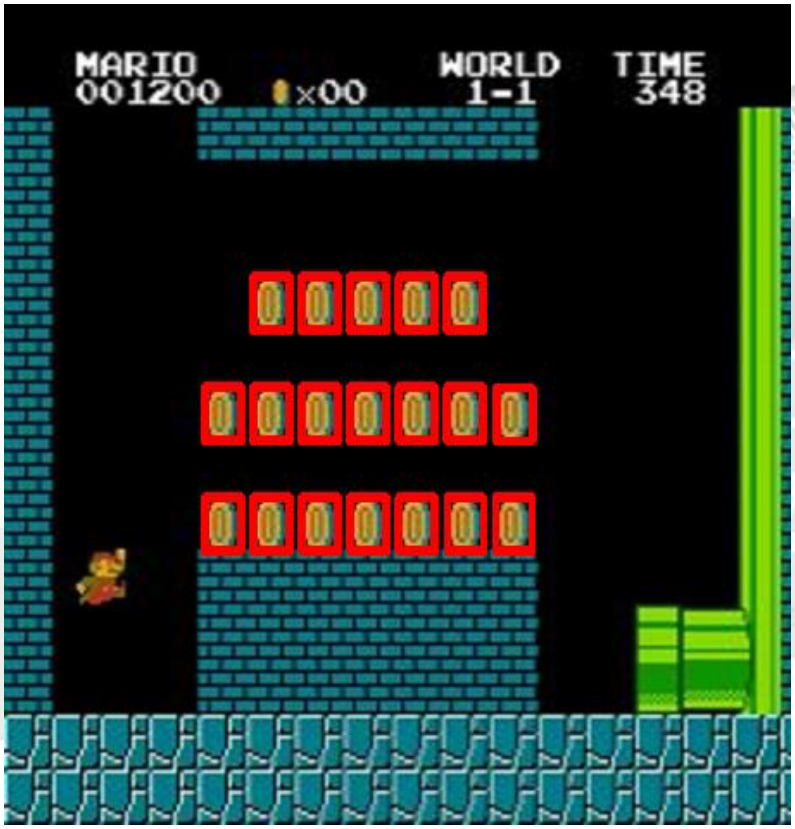
1. Template Matching



Quiz 1 (15 min)

Let you show the location of coins in "**mario.png**" and template image "**mario_coin.png**" (free to select template matching method)

Output image



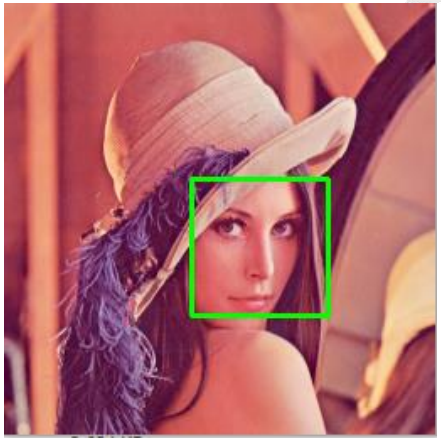
KMITL BEYOND THE LIMIT

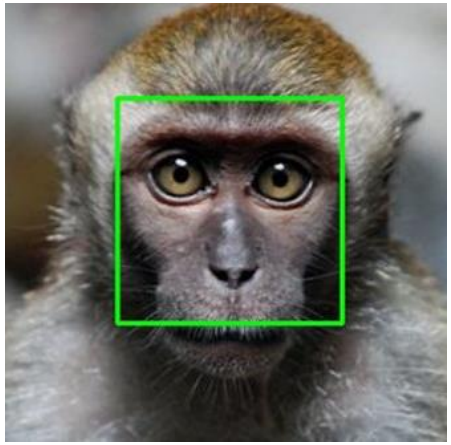
COMPUTER VISIONS

01416500
Course Code

KMITL ROBOTICS and AI International Program

Image Recognition





Face Detection

Asst. Prof. Dr. Anakkapon Saenthon
King Mongkut's Institute of Technology Ladkrabang

2. Face Detection

What is “Face Detection”?

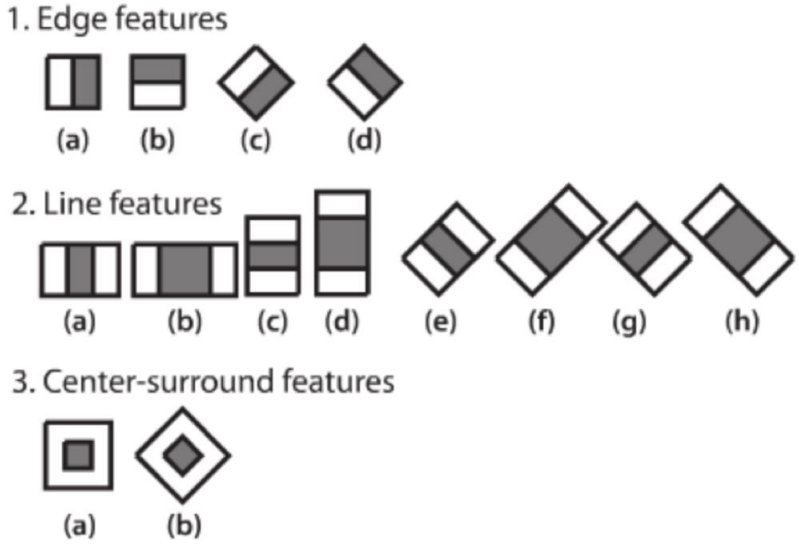
Face detection is a type of application classified under “**computer vision**” technology. It is the process in which algorithms are developed and trained to properly locate faces or objects (in object detection, a related system), in images.

Face detection uses ***classifiers***, which are algorithms that detects what is either a face(1) or not a face(0) in an image. Classifiers have been trained to detect faces using thousands to millions of images in order to get more accuracy.

2. Face Detection

Understanding Haar Cascades

A sequence of rescaled "**square-shaped**" functions which together form a wavelet family or basis.

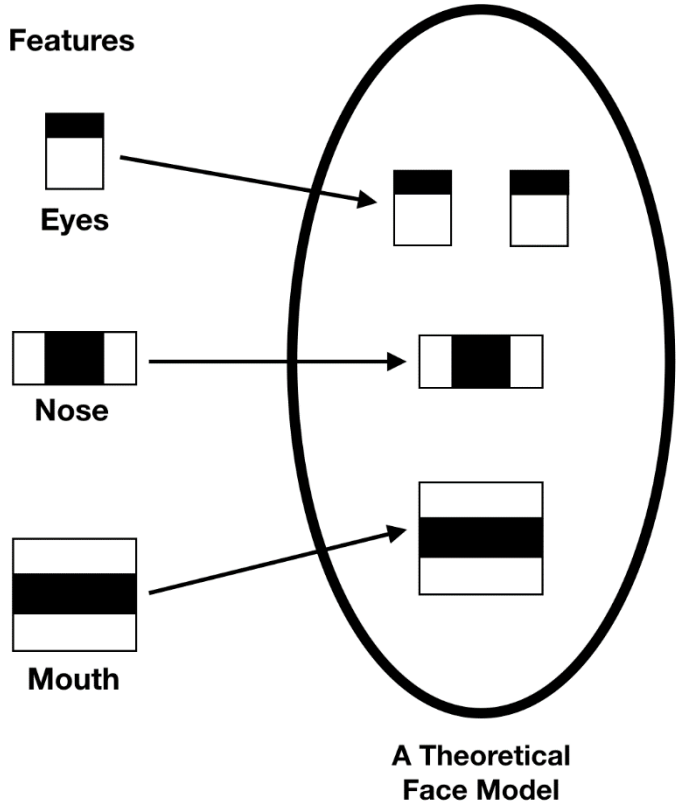


Face Detection determines the locations and sizes of human faces in arbitrary (digital) images.

In **Face Recognition**, the use of Face Detection comes first to determine and isolate a face before it can be recognized.

2. Face Detection

Feature Extraction



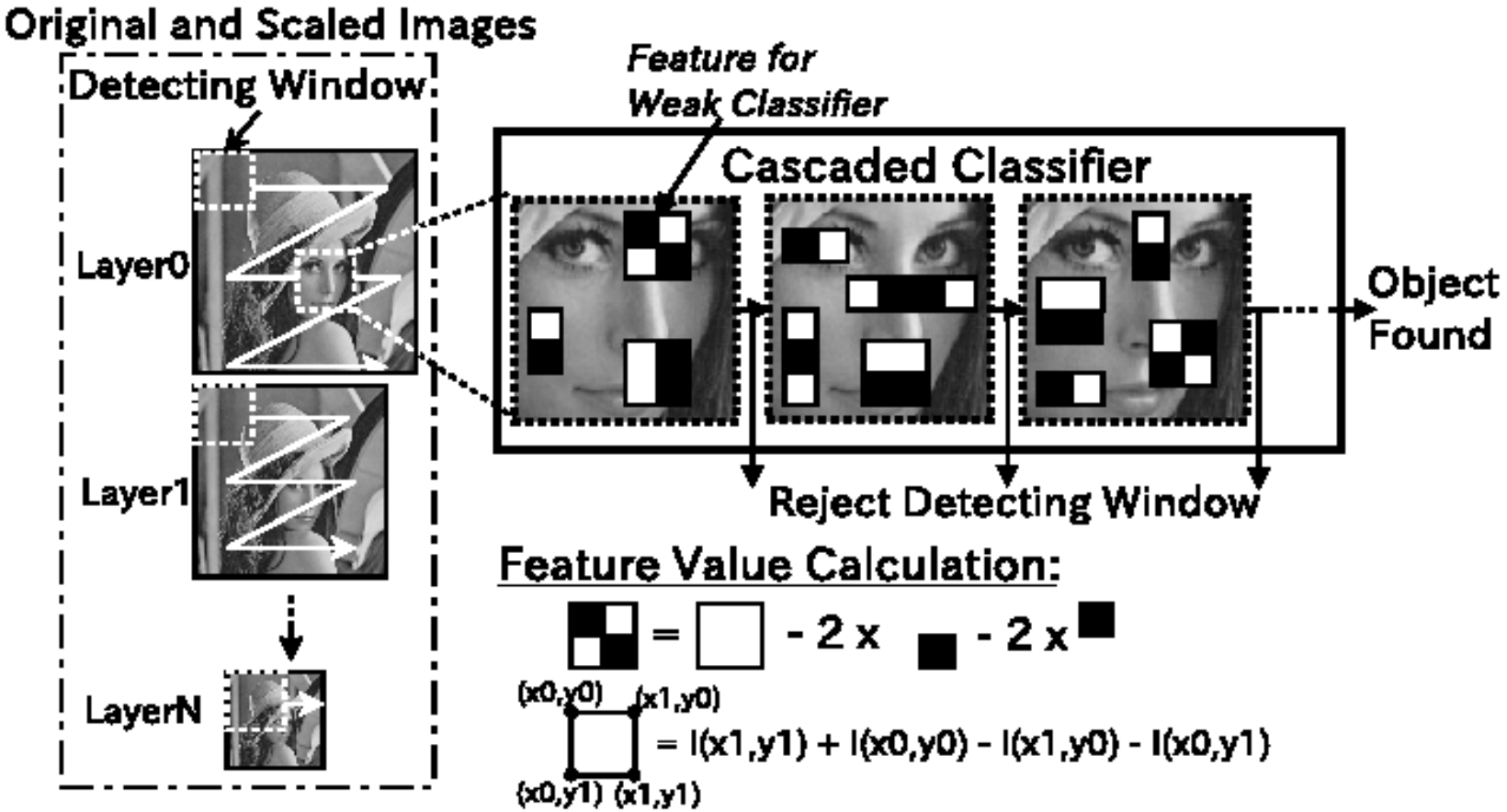
Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images.

This process in the algorithm is feature extraction.

The training data used in this project is an XML file called:

haarcascade_frontalface_default.xml

2. Face Detection



Ref : A low-power Adaboost-based object detection processor using Haar-like features

2. Face Detection

This contains code parameters that are the most important to consider.

scaleFactor: The value indicates how much the image size is reduced at each image scale. A lower value uses a **smaller step for downscaling**. This allows the algorithm to detect the face. It has a value of x.y, where x and y are arbitrary values, you can set.

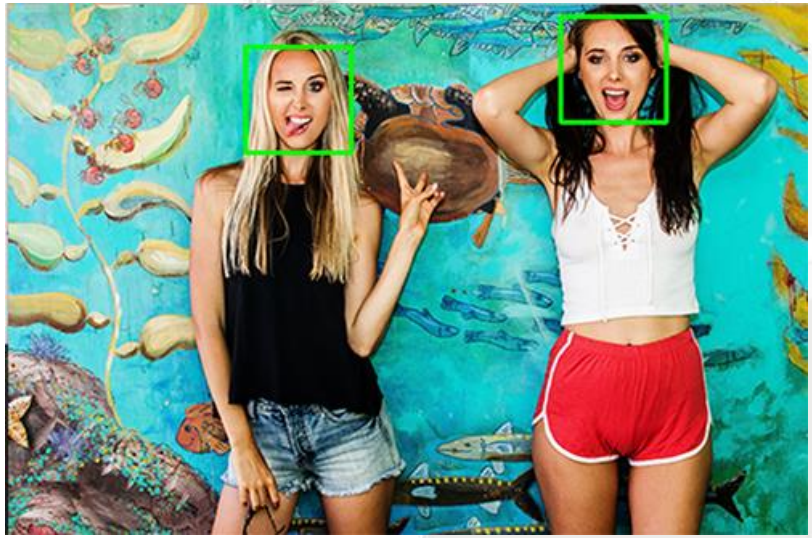
minNeighbors: This parameter specifies how many "**neighbors**" each candidate rectangle should have. A higher value results in less detections but it detects higher quality in an image.

minSize: The minimum **object size**. By default it is (30,30). The smaller the face in the image, it is best to adjust the minSize value lower.

2. Face Detection

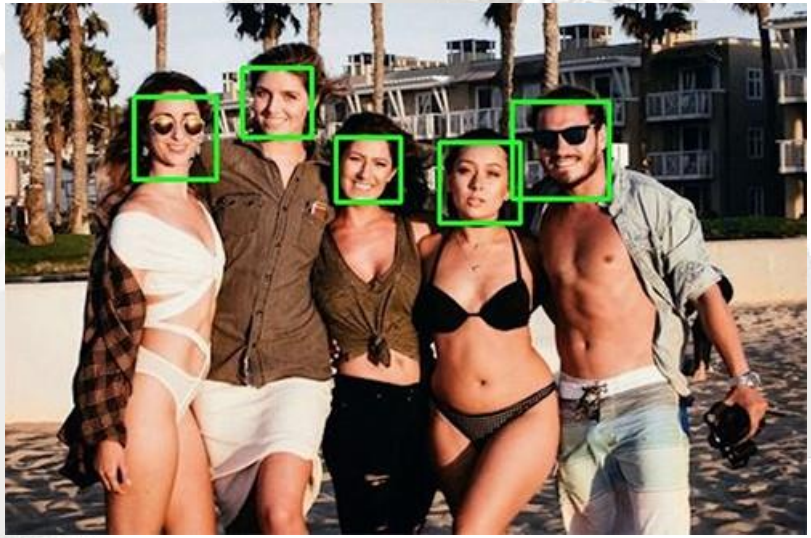
Parameters:

scaleFactor=1.4,
minNeighbors=1,
minSize=(10,10),



Parameters:

scaleFactor=1.6,
minNeighbors=2,
minSize=(20,20),



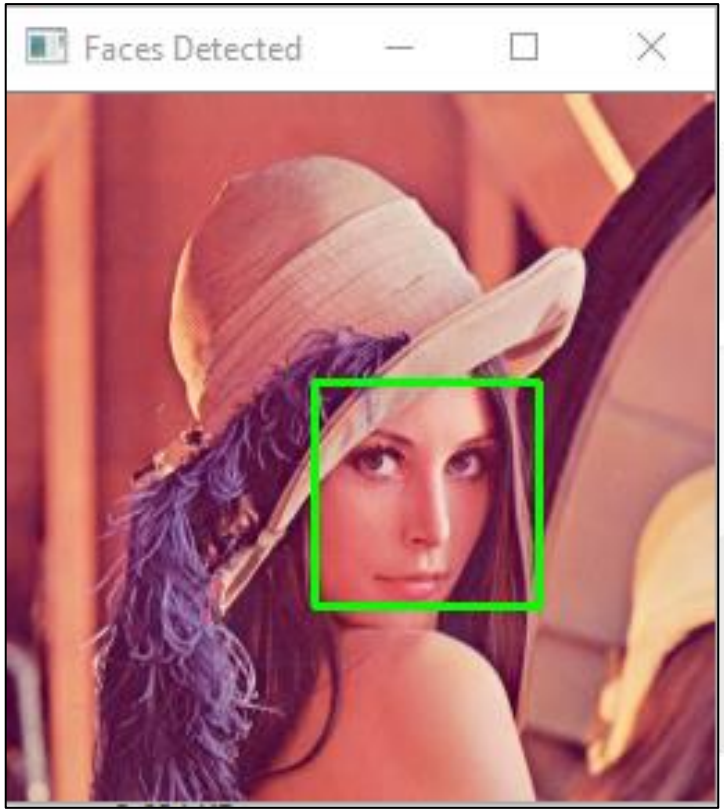
Ref : https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177

2. Face Detection

Running OpenCV

OpenCV function :

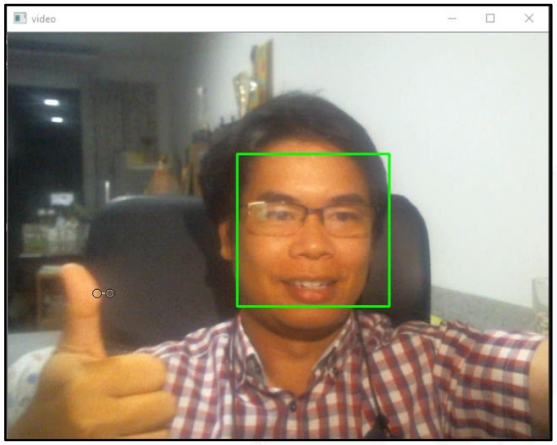
faceCascade.detectMultiScale()

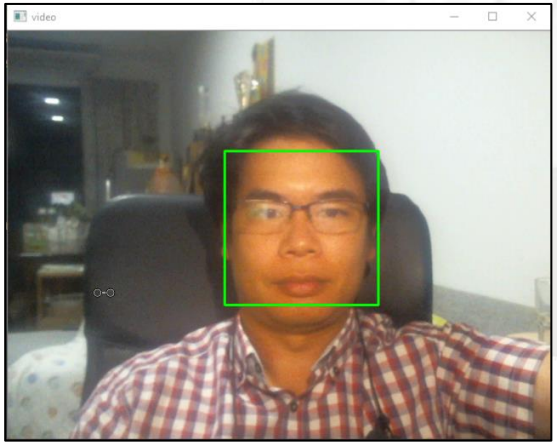



```python
import cv2

cascPath = "dataset/haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascPath)

image = cv2.imread("dataset/lena_color_256.tif")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(1,1),
    flags = cv2.CASCADE_SCALE_IMAGE
)
print("Detected {0} faces!".format(len(faces)))

for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow("Faces Detected", image)
cv2.waitKey(0)
```

2. Face Detection

This example for video capture

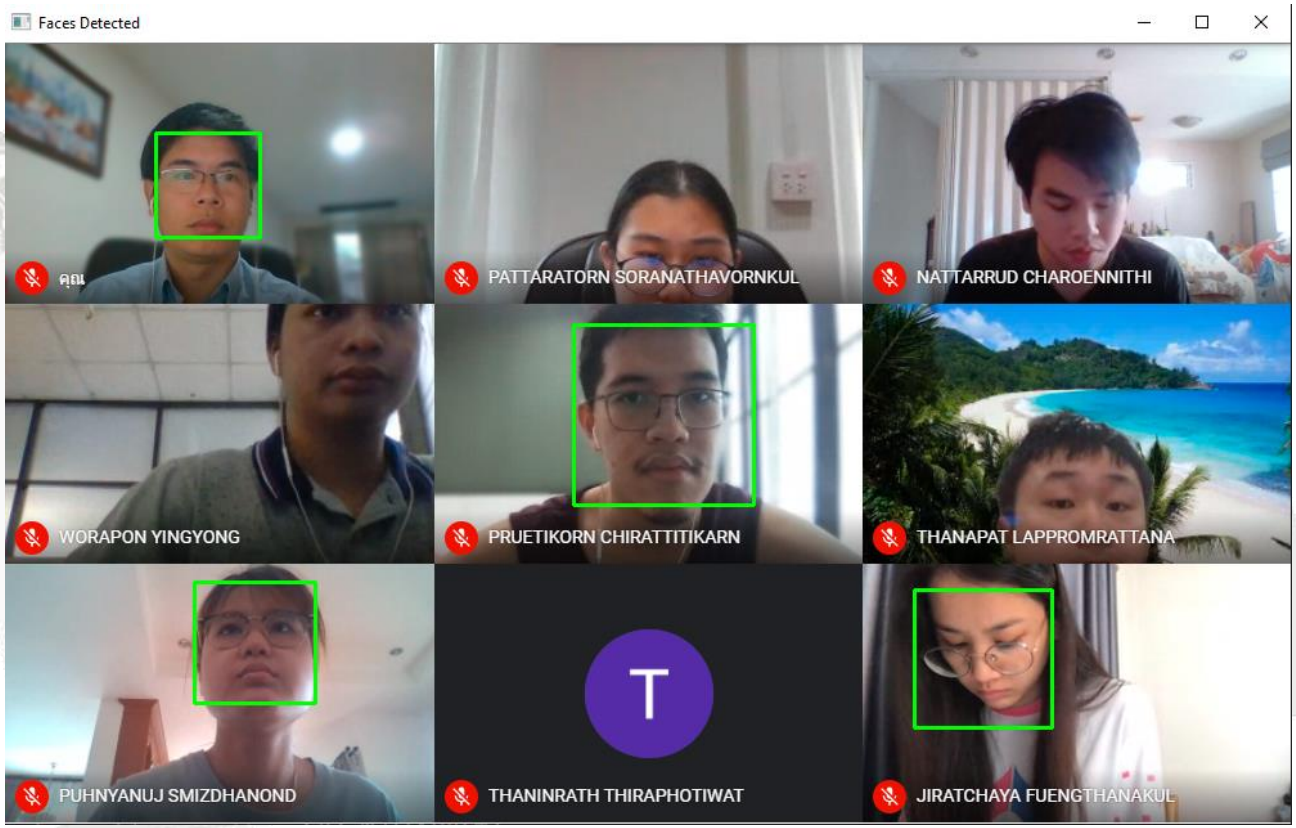





```python
import numpy as np
import cv2

faceCascade = cv2.CascadeClassifier(
    'dataset/haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height
while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20)
    )
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
    cv2.imshow('video',img)
    k = cv2.waitKey(30) & 0xff
    if k == 27: # press 'ESC' to quit
        break
cap.release()
cv2.destroyAllWindows()
```

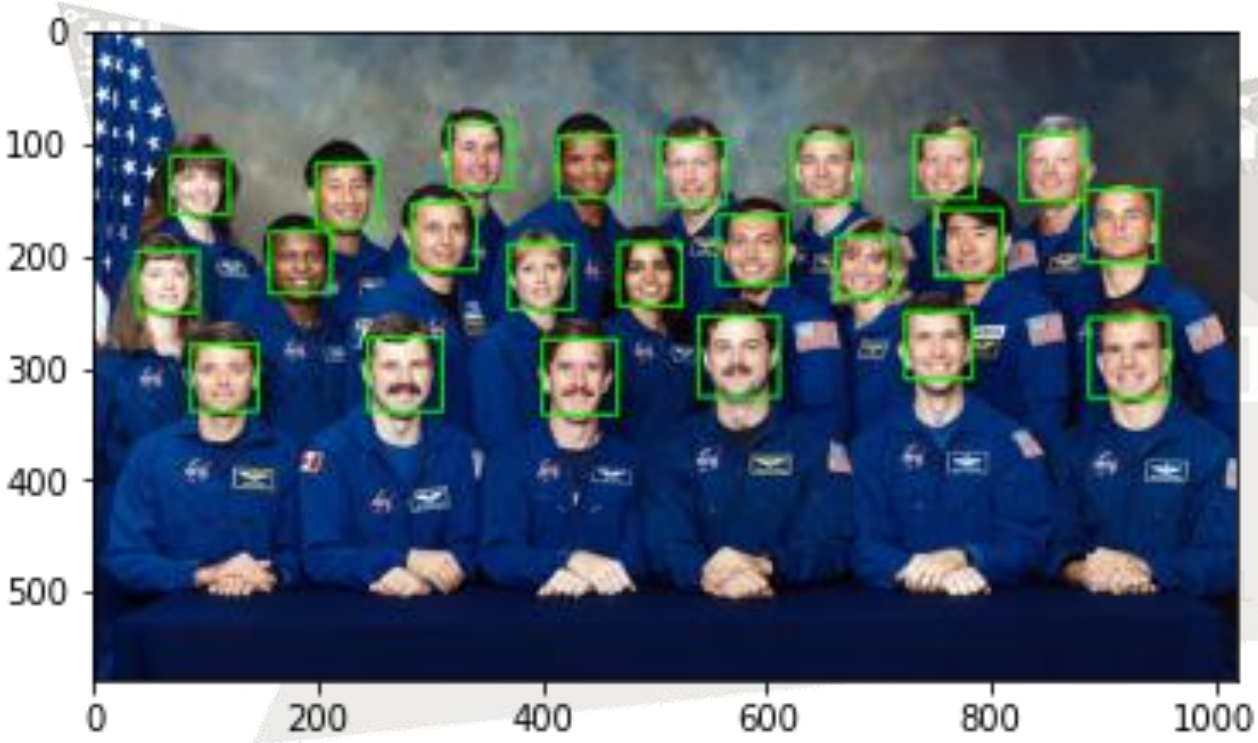
2. Face Detection

```
gray,
scaleFactor=1.2,
minNeighbors=6,
minSize=(20,20)
```



Quiz 2 (15 min)

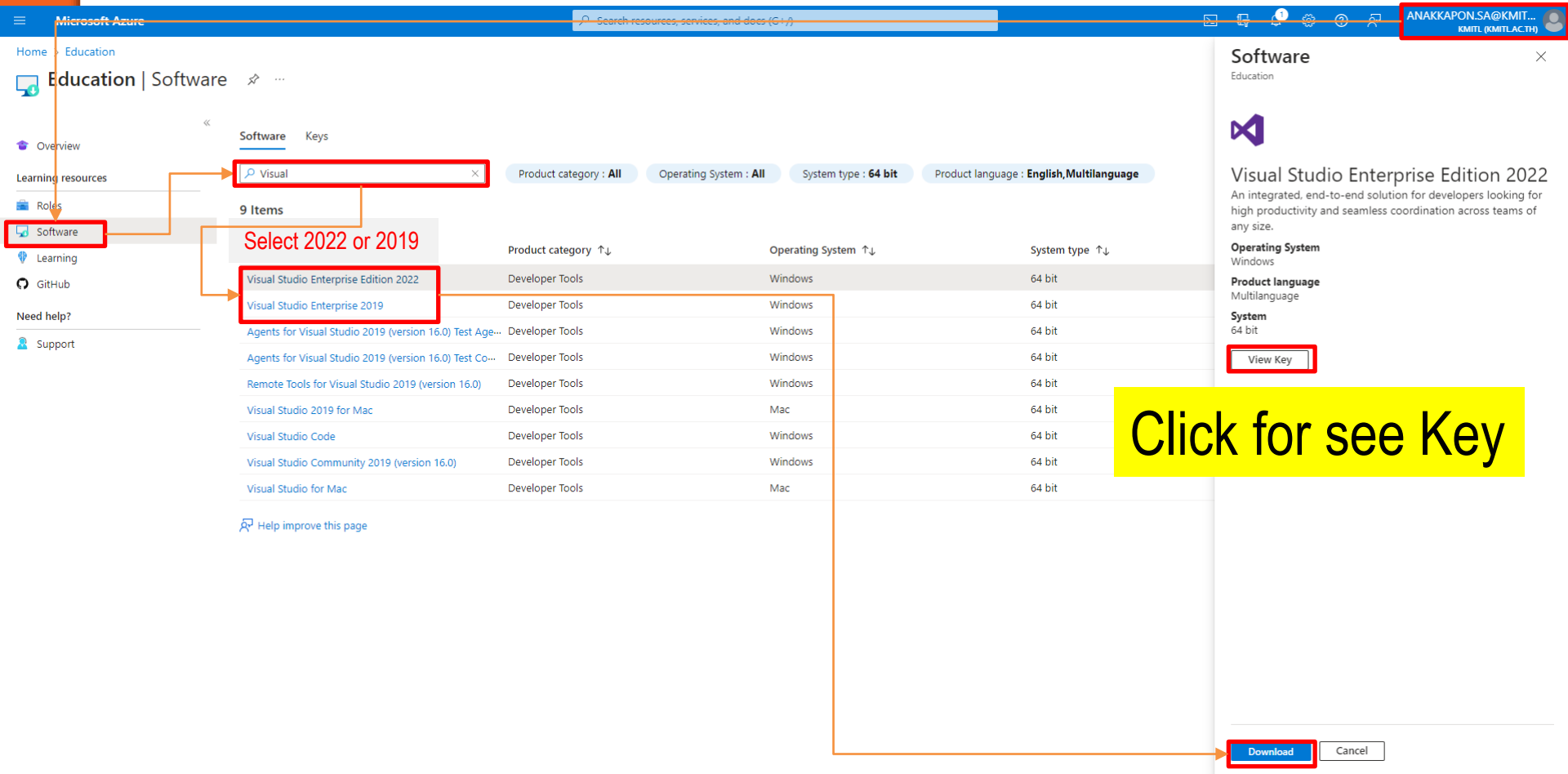
Find all face in "NASA.jpg" image

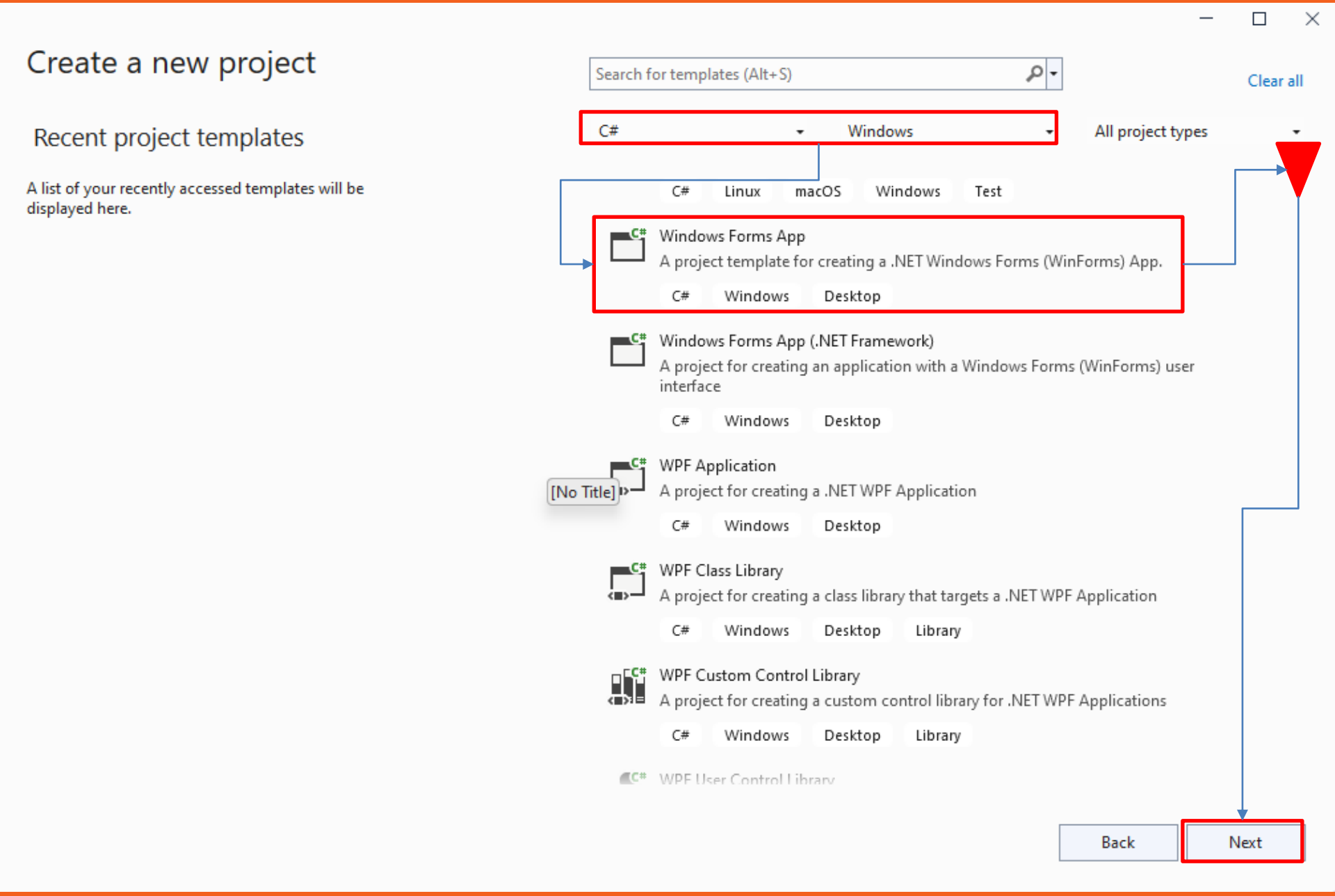


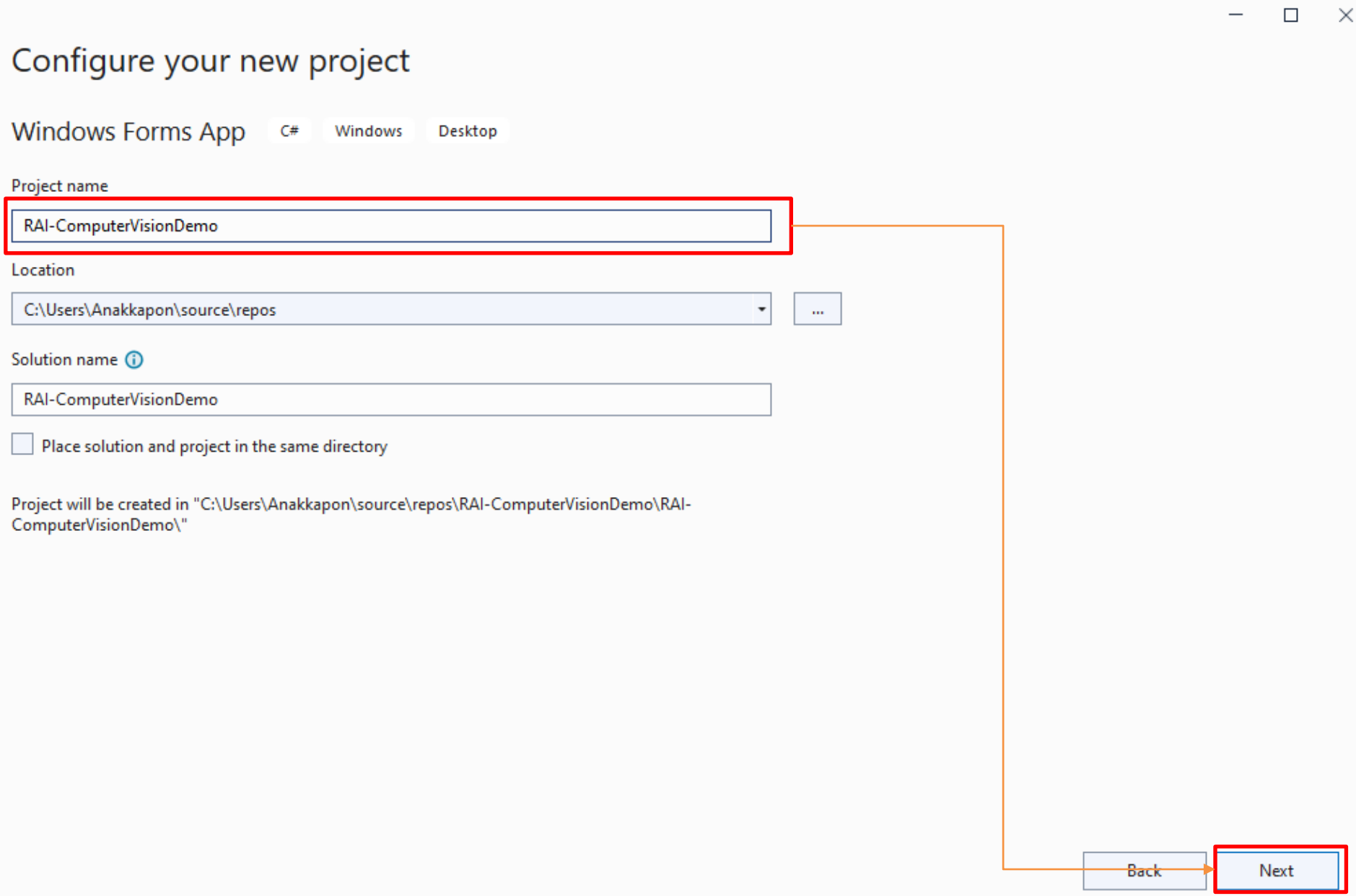
Next class

Download **Visual Studio C# .NET**
https://azureforeducation.microsoft.com/devtools

KMITL account login




Create a new project
Recent project templates
A list of your recently accessed templates will be displayed here.
Search for templates (Alt+S)
Clear all
C#
Windows
All project types
C# Linux macOS Windows Test
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop
Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop
WPF Application
[No Title]
A project for creating a .NET WPF Application
C# Windows Desktop
WPF Class Library
A project for creating a class library that targets a .NET WPF Application
C# Windows Desktop Library
WPF Custom Control Library
A project for creating a custom control library for .NET WPF Applications
C# Windows Desktop Library
WPF User Control Library
Back
Next


Configure your new project
Windows Forms App
C#
Windows
Desktop
Project name
RAI-ComputerVisionDemo
Location
C:\Users\Anakkapon\source\repos
...
Solution name
RAI-ComputerVisionDemo
Place solution and project in the same directory
Project will be created in "C:\Users\Anakkapon\source\repos\RAI-ComputerVisionDemo\RAI-ComputerVisionDemo\"
Back
Next

Assignment#9.3

1. Preparation software of Visual Studio .NET C# (version 2019 or 2022)
2. Capture Visual Studio .NET C# screen
3. Submitting an image file to MS-TEAM