

A
WINTER LIVE PROJECT
[FACE DETECTION SYSTEM]

BY
Mani Sharma
(0221BCA060)
Aman Kumar
(0221BCA045)
Sajan Kumar
(0221BCA028)
Ravi Kumar
(0221BCA001)



Under the supervision of
Prof. Sanjay Kumar
FACULTY OF IT DEPARTMENT,
DOON BUSINESS SCHOOL
(DEHRADUN)

CANDIDATE'S DECLARATION

We hereby declare that the mini project work being presented in this report entitled **"FACE DETECTION SYSTEM"** submitted in the department of computer application, FACULTY OF TECHNOLOGY, *Doon business school, Dehradun* is the authentic work carried out us under the guidance of **PROF. SANJAY KUMAR**, Professor, Department of computer application, *Doon business school, Dehradun*.

Date: 25/02/2023

Mani Sharma

Aman Kumar

Ravi Kumar

Sajan Kumar

B.C.A

Doon Business School,Dehradun

ACKNOWLEDGEMENT

We would like to express my special thanks of gratitude to my teacher **PROF. SANJAY KUMAR** who gave us the golden opportunity to do this wonderful project “**FACE DETECTION SYSTEM**” which also helped us doing a lot of research and we came to know about so many new things. We are really thankful to them.

Date: 25/02/2023

Mani Sharma (0221BCA060)

Aman Kumar (0221BCA081)

Ravi Kumar (0221BCA001)

Sajan Kumar (0221BCA028)

B.C.A

Doon Business School, Dehradun

Table of Content

SYNOPSIS	5
OBJECTIVE OF THE PROJECT:	6
SDLC	7
LANGUAGE AND TOOLS USED	10
STRUCTURE OF THE PROJECT	12
CODE.....	19
DECODING OF CODE	20
FLOWCHART OF FACE DETECTION:	22
CONCLUSION-.....	23
BIBLIOGRAPHY:	24

SYNOPSIS

TITLE OF THE PROJECT:

Face detection -- also called facial detection -- is an artificial intelligence (AI) based computer technology used to find and identify human faces in digital images. Face detection technology can be applied to various fields -- including security, biometrics, law enforcement, entertainment and personal safety -- to provide surveillance and tracking of people in real time.

Face detection has progressed from rudimentary computer vision techniques to advances in machine learning (ML) to increasingly sophisticated artificial neural networks (ANN) and related technologies; the result has been continuous performance improvements. It now plays an important role as the first step in many key applications -- including face tracking, face analysis and facial recognition. Face detection has a significant effect on how sequential operations will perform in the application.

In face analysis, face detection helps identify which parts of an image or video should be focused on to determine age, gender and emotions using facial expressions. In a facial recognition system -- which maps an individual's facial features mathematically and stores the data as a faceprint -- face detection data is required for the algorithms that discern which parts of an image or video are needed to generate a faceprint. Once identified, the new faceprint can be compared with stored faceprints to determine if there is a match.

OBJECTIVE OF THE PROJECT:

➤ Improved security:-

Face detection improves surveillance efforts and helps track down criminals and terrorists. Personal security is also enhanced since there is nothing for hackers to steal or change, such as passwords.

➤ Easy to integrate:-

Face detection and facial recognition technology is easy to integrate, and most solutions are compatible with the majority of security software.

➤ Automated identification:-

In the past, identification was manually performed by a person; this was inefficient and frequently inaccurate. Face detection allows the identification process to be automated, thus saving time and increasing accuracy.

SDLC

- * SDLC stands for " *Software development life cycle*".
- * SDLC is a collection of processes which are followed to develop a software.
- * SDLC is a methodology that defines some processes which are followed to develop a high quality software.
- * It covers the detailed plan for building, deploying and maintaining the software.
- * The main of SDLC is to define all the tasks for developing and maintaining software.
- * It is followed for a software project within a software developing organization.

PHASES OF SDLC

- * PHASE-1- Requirement analysis
- * PHASE-2-Feasibility study
- * PHASE-3-Design
- * PHASE-4- Coding
- * PHASE-5- Testing
- * PHASE-6- Deployment
- * PHASE-7-Maintenance

***PHASE-1- REQUIREMENT ANALYSIS**

- * In this all the necessary information is collected from the customer to develop software as per their expectations
- * Some important questions like- what is need of software, who will be end user, what is the future scope.

***PHASE-2- FEASIBILITY STUDY**

- * In this organization discuss about the cost and benefits of software.
- * It measures how much beneficial the product is for organization.

TYPES OF FEASIBILITY

1- Technical feasibility

In this we check whether we have required technical resources

2- Economical feasibility:

The cost and benefits of the project is analysed.

3- Legal feasibility:

- * we investigate whether the project is legal or not.

4- Operational feasibility:

*we examine whether project satisfies the requirements identified in the requirement analysis phase.

5- Scheduling feasibility:

- *In this we estimate the time necessary to complete the project.

***PHASE-3-DESIGN**

- *It includes the design of everything that has to be coded.

- * In this process SRS(system requirement specification) document is created which has all the information like- language to be used, database types.

***PHASE-4-CODING**

- * A number of developers code modules and then all modules are arranged together to work efficiently.

***PHASE-5- TESTING**

- * In this software is tested for bugs and errors.

***PHASE-6- DEPLOYMENT**

- *After the testing the software that is bug free it is launched and available for the users.

***PHASE-7- MAINTENANCE**

- * Maintenance team look over the software and users feedback.

- * Maintenance necessary to eliminate errors in the system during the working life and to tune the software.

LANGUAGE AND TOOLS USED

Language used is PYTHON

1. Python is a high level language like other languages like Java, C++, PHP, Ruby, Basic and Perl.
2. Python is developed by “Guido van’ Rossum” in 1991 at CWI in Netherland.
3. Python is an object oriented programming language.
4. Python provides security.
5. It is very easy to understand by human while computer needs compiler or interpreter to understand python language.

WHY NAME PYTHON?

Guido van Rossum was a fan of the popular comedy show “Monty Python’s flying circus” broadcasted by BBC (1967-74), so he decided to pick name his language as “Python”.

FEATURES OF PYTHON

- **EASY TO READ**- It is very to read python a normal people who don’t know python can also read the language. It is like a normal English language code.
- **EASY TO LEARN** - It is programmer friendly language.
- **INTRERPRETED LANGUAGE**- It is an interpreted language i.e. interpreter executed the code line by line at a time. This makes debugging easy and thus suitable for beginners.

- **CROSS PLATFORM or PORTABLE LANGUAGE**- Python can run Equally on different platforms such as Windows, Linux, Unix, Macintosh etc.
- **OPEN SOURCE**- It is freely available. The source code is also available.
- **OBJECT ORIENTED LANGUAGE**- We can use all the features of oops .we can use concepts of classes.
- **STANDARD LIBRARY**- Python has a large and broad library.
- **EXTENSIBLE**- It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in your python code.
- **SHORT CODES**- In this codes are not very long so it is easy to learn and understand this language.

Software used:

- **ANACONDA**-Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux. People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.
- **JUPYTER NOTEBOOK**- The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

STRUCTURE OF THE PROJECT

Face detection applications use algorithms and ML to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes -- one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face.

To help ensure accuracy, the algorithms need to be trained on large data sets incorporating hundreds of thousands of positive and negative images. The training improves the algorithms' ability to determine whether there are faces in an image and where they are.

The methods used in face detection can be knowledge-based, feature-based, template matching or appearance-based. Each has advantages and disadvantages:

Knowledge-based, or rule-based methods, describe a face based on rules. The challenge of this approach is the difficulty of coming up with well-defined rules. Feature invariant methods -- which use features such as a person's eyes or nose to detect a face -- can be negatively affected by noise and light.

Template-matching methods are based on comparing images with standard face patterns or features that have been stored previously and correlating the two to detect a face. Unfortunately these methods do not address variations in pose, scale and shape.

Appearance-based methods employ statistical analysis and machine learning to find the relevant characteristics of face images. This method, also used in feature extraction for face recognition, is divided into sub-methods.

Some of the more specific techniques used in face detection include:

Removing the background. For example, if an image has a plain, mono-color background or a pre-defined, static background, then removing the background can help reveal the face boundaries.

In color images, sometimes skin color can be used to find faces; however, this may not work with all complexions.

Using motion to find faces is another option. In real-time video, a face is almost always moving, so users of this method must calculate the moving area. One drawback of this method is the risk of confusion with other objects moving in the background.

A combination of the strategies listed above can provide a comprehensive face detection method.

Detecting faces in pictures can be complicated due to the variability of factors such as pose, expression, position and orientation, skin color and pixel values, the presence of glasses or facial hair, and differences in camera gain, lighting conditions and image resolution. Recent years have brought advances in face detection using deep learning, which presents the advantage of significantly outperforming traditional computer vision methods.

Major improvements to face detection methodology came in 2001, when computer vision researchers Paul Viola and Michael Jones proposed a framework to detect faces in real time with high accuracy. The Viola-Jones framework is based on training a model to understand what is and is not a face. Once trained, the model extracts specific features, which are then stored in a file so that features from new images can be compared with the previously stored features at various stages. If the image under study passes through each stage of the feature comparison, then a face has been detected and operations can proceed.

Although the Viola-Jones framework is still popular for recognizing faces in realtime applications, it has limitations. For example, the framework might not work if a face is covered with a mask or scarf, or if a face is not properly oriented, then the algorithm might not be able to find it.

FIRST STEP TOWARDS PROJECT

INSTALL ANACONDA:

- 1) Go to anaconda website <https://www.anaconda.com/products/individual> and click on the "Download" button for your operating system (Windows, macOS, or Linux).
- 2) Once the download is complete, double-click the downloaded file to start the installation process.
- 3) Follow the installation wizard instructions to select the installation location, agree to the license agreement, and select the advanced options you want to enable. You can usually leave the default options selected unless you have specific requirements.
- 4) Once the installation is complete, you should see the Anaconda Navigator icon on your desktop or in your Start menu. Click on it to open the Navigator, which is a graphical user interface that allows you to manage your Anaconda environments and packages.
- 5) You can also open a terminal or command prompt and type "anacondanavigator" to launch the Navigator from the command line.

That's it! You have successfully installed Anaconda on your computer

How to install open cv:

OpenCV (Open Source Computer Vision) is a widely-used library for computer vision and image processing. Here are the steps to install OpenCV:

Install Python: OpenCV supports Python 3.x versions. You can download and install the latest version of Python from the official Python website.

Install pip: pip is the package installer for Python. You can check if pip is installed by typing "pip" in the command prompt/terminal. If it is not installed, you can download and install it from the official pip website.

Install OpenCV: There are different ways to install OpenCV, but the easiest way is to use pip. Open the command prompt/terminal and type the following command:

```
pip install opencv-python
```

This command will download and install the latest version of OpenCV for Python.

Verify the installation: To verify that OpenCV is installed correctly, open a Python interpreter or a Jupyter Notebook and type the following command:

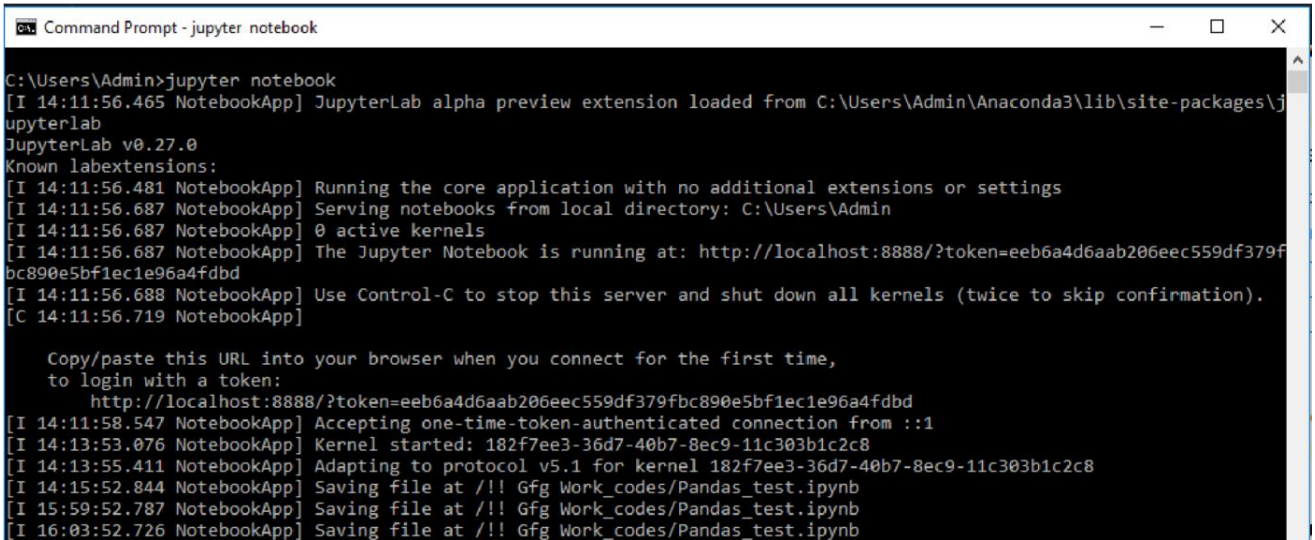
```
python  
  
import cv2  
print(cv2.__version__)
```

This command will import the OpenCV library and print its version.

That's it! You have successfully installed OpenCV on your system

Command to run the Jupyter notebook: jupyter notebook

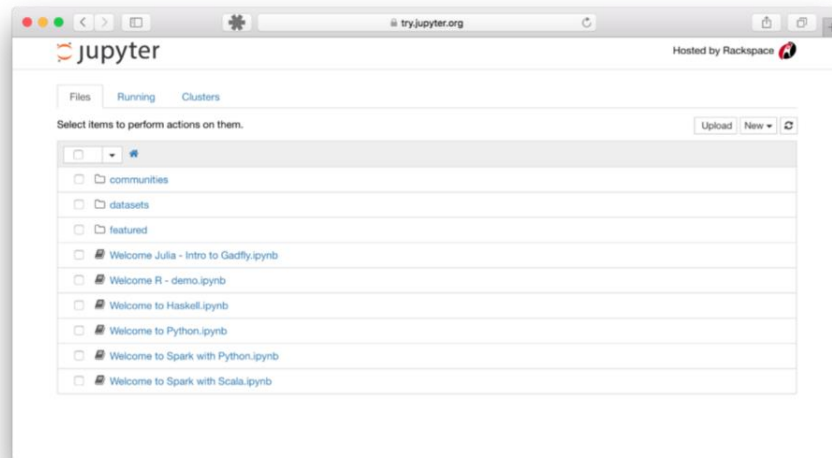
This will print some information about the notebook server in your terminal, including the URL of the web application (by default, <http://localhost:8888>) and then open your default web browser to this URL.



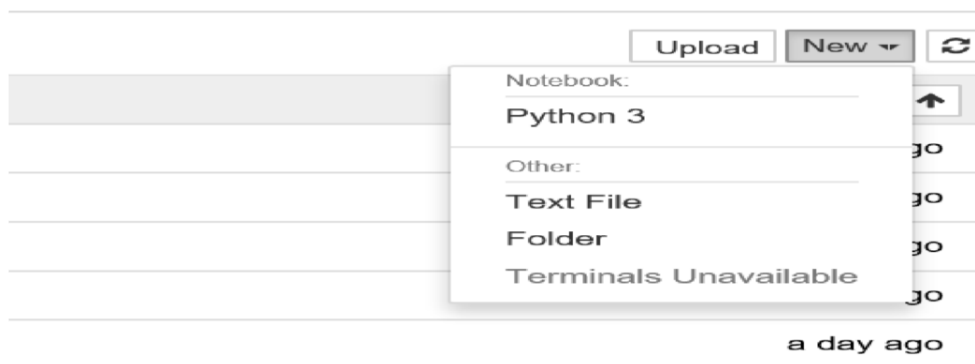
```
Command Prompt - jupyter notebook
C:\Users\Admin>jupyter notebook
[I 14:11:56.465 NotebookApp] JupyterLab alpha preview extension loaded from C:\Users\Admin\Anaconda3\lib\site-packages\jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 14:11:56.481 NotebookApp] Running the core application with no additional extensions or settings
[I 14:11:56.687 NotebookApp] Serving notebooks from local directory: C:\Users\Admin
[I 14:11:56.687 NotebookApp] 0 active kernels
[I 14:11:56.687 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=eeb6a4d6aab206eec559df379fbc890e5bf1ec1e96a4fdbd
[I 14:11:56.688 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:11:56.719 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=eeb6a4d6aab206eec559df379fbc890e5bf1ec1e96a4fdbd
[I 14:11:58.547 NotebookApp] Accepting one-time-token-authenticated connection from ::1
[I 14:13:53.076 NotebookApp] Kernel started: 182f7ee3-36d7-40b7-8ec9-11c303b1c2c8
[I 14:13:55.411 NotebookApp] Adapting to protocol v5.1 for kernel 182f7ee3-36d7-40b7-8ec9-11c303b1c2c8
[I 14:15:52.844 NotebookApp] Saving file at /!! Gfg Work_codes/Pandas_test.ipynb
[I 15:59:52.787 NotebookApp] Saving file at /!! Gfg Work_codes/Pandas_test.ipynb
[I 16:03:52.726 NotebookApp] Saving file at /!! Gfg Work_codes/Pandas_test.ipynb
```

When the notebook opens in your browser, you will see the Notebook Dashboard, which will show a list of the notebooks, files, and sub directories in the directory where the notebook server was started. Most of the time, you will wish to start a notebook server in the highest level directory containing notebooks. Often this will be your home directory.



Now on the dashboard, you can see a new button at the top right corner. Click it to open a drop-down list and then if you'll click on Python3, it will open a new notebook.



```
Documents/project/ x facedetect - Jupyter Notebook x +
localhost:8888/notebooks/Documents/project/facedetect.ipynb
jupyter facedetect Last Checkpoint: 12 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)
In [1]: import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

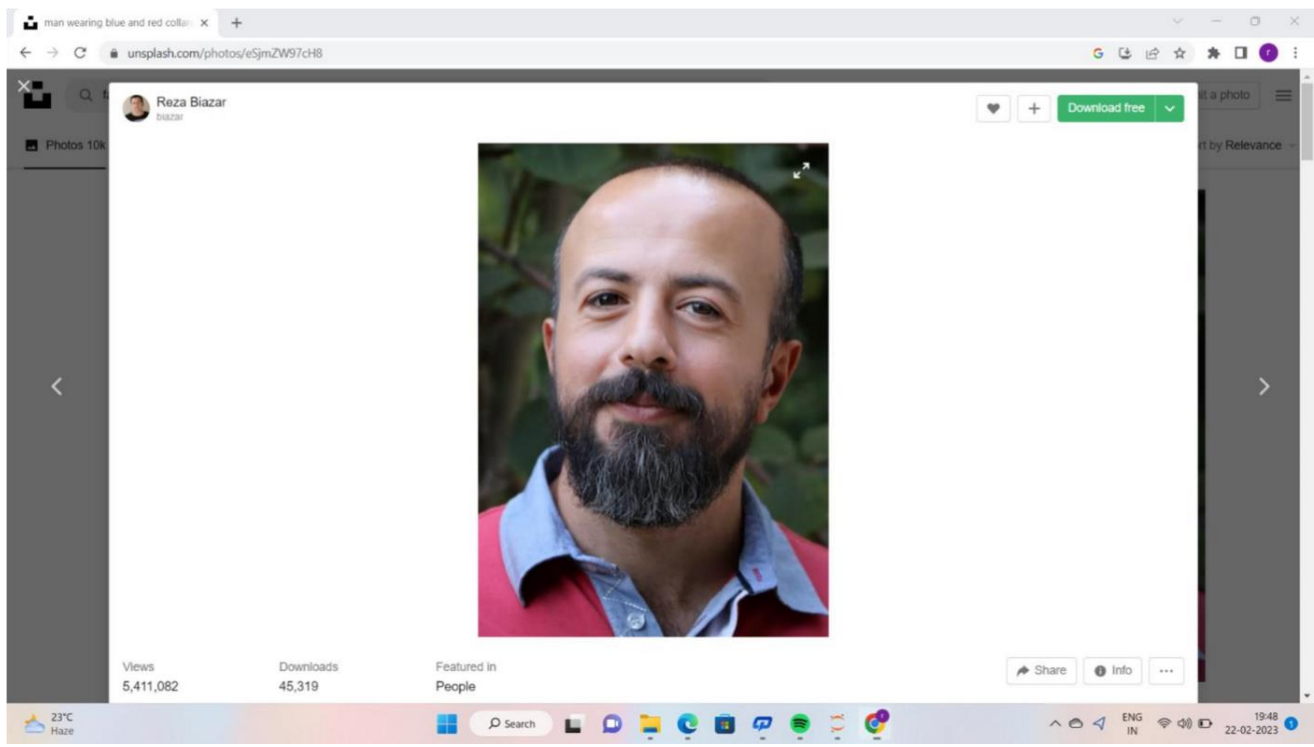
img = cv2.imread('C:/Users/Ravi kumar/Documents/project/face3.png')
print (img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#print(gray)

faces = face_cascade.detectMultiScale(img)
faces = face_cascade.detectMultiScale(gray)

for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),5)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_color)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    cv2.imshow('img',img)
k = cv2.waitKey(0)
if k == 27: # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
    cv2.imwrite('messigray.png',img)
    cv2.destroyAllWindows()

[[[ 0 0 0]
[ 0 0 0]
[ 0 0 0]]]
```



CODE

```
import numpy as np import cv2

face_cascade =

cv2.CascadeClassifier('haarcascade_frontalface_default.xml'
)
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('lena.jpg')

# print (img)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#print(gray)
faces = face_cascade.detectMultiScale(img) faces =
face_cascade.detectMultiScale(gray)
for (x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]      roi_color = img[y:y+h,
x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray) for
(ex,ey,ew,eh) in eyes:
    cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh), (0,255,0)
,2)
    cv2.imshow('img',img)

k = cv2.waitKey(0)
if k == 27:          # wait for ESC key to exit
cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
cv2.imwrite('messigray.png',img)
cv2.destroyAllWindows()
```

DECODING OF CODE

- Imports the numpy and cv2 modules.
- Loads two cascade classifiers for face and eye detection.
- Reads an image file (lena.jpg) using the cv2.imread() function. ➤ Converts the image to grayscale using cv2.cvtColor() function.
- Detects faces in the grayscale image using the detectMultiScale() method of the face_cascade object.
- For each face detected, it draws a rectangle around it using cv2.rectangle() function.
- Selects the region of interest (ROI) for each face, i.e., the area within the rectangle.
- Detects eyes in the ROI using the detectMultiScale() method of the eye_cascade object.
- For each eye detected, it draws a rectangle around it using cv2.rectangle() function.
- Displays the resulting image using the cv2.imshow() function.
- Waits for a key event using the cv2.waitKey() function.
- If the ESC key is pressed, it closes all windows and exits the program.
- If the 's' key is pressed, it saves the resulting image to a file and exits the program.
- Note: Make sure to have the required Haar cascades (haarcascade_frontalface_default.xml and haarcascade_eye.xml) in the same directory as your Pythons

Documents/project/ x facedetect - Jupyter Notebook x +

localhost:8888/notebooks/Documents/project/facedetect.ipynb

Jupyter facedetect Last Checkpoint: 02/09/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [*]: import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread("C:/Users/Ravi kumar/Documents/project/face3.png")
print(img)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

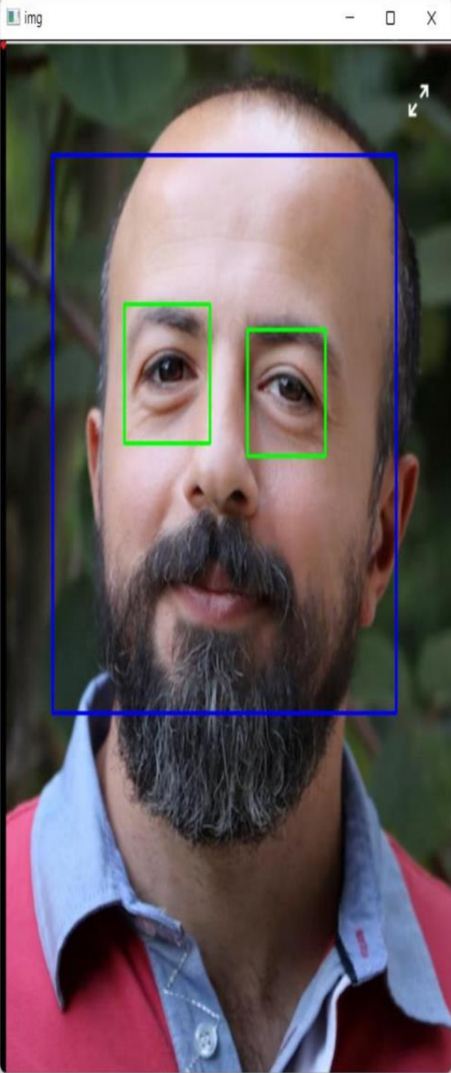
#print(gray)

faces = face_cascade.detectMultiScale(img)
faces = face_cascade.detectMultiScale(gray)

for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    cv2.imshow('img',img)

k = cv2.waitKey(0)
if k == 27: # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
    cv2.imwrite('messigray.png',img)
    cv2.destroyAllWindows()
```

[[[0 0 0]
[0 0 0]
[0 0 0]]



FLOWCHART OF FACE DETECTION

Here is a basic flowchart of face detection:

- 1)Start
- 2)Load input image
- 3)Convert the image to grayscale
- 4)Apply a Haar Cascade Classifier to detect faces in the grayscale image
- 5)For each detected face:
 - a. Draw a rectangle around the face
 - b. Extract the face region from the grayscale image
 - c. Apply any additional processing to the face region (e.g. facial landmark detection)
- 6)Display the output image with detected faces and any additional processing

CONCLUSION

Face recognition technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

To achieve this goal computers must be able to reliably identify nearby people in a manner that fits naturally within the pattern of normal human interactions. They must not require special interactions and must conform to human intuitions about when recognition is likely. This implies that future smart environments should use the same modalities as humans, and have approximately the same limitations. These goals now appear in reach -- however, substantial research remains to be done in making person recognition technology work reliably, in widely varying conditions using information from single or multiple modalities

BIBLIOGRAPHY

www.google.com --www.anaconda.com

www.geeksforgeeks.org/face-detection-cascade

www.unsplash.com/photos/image

www.pythonbasic.org/face-detection www.w3schools.com