Mini Project Report on

Face Detection

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Graphic Era
Deemed to be University

SUBMITTED BY:

Aditya Bahl

2016580

5th SEMESTER

DEHRADUN

2022-2023

GUIDED BY:

Mr. Saurabh Kumar Mishra

DECLARATION

I, **Aditya Bahl** student of **B-tech, Semester 5**, Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, declare that the technical project work titled "**Face Detection**" has been carried out by me and submitted in partial fulfilment of the course requirements for the award of degree in B. Tech of Graphic Era Deemed University during the academic year **2022-2023.** The matter embodied in this synopsis has not been submitted to any other university or institution for the award of any other degree or diploma.

Date: 07/01/23

GraphicEra
(Deemed to be University)
Accredited by NAAC with Grade A

CERTIFICATE

This is to certify that the project report titled "**Face Detection**" is a bona fide project work carried out by **Aditya Bahl, 2016580**, in partial fulfilment of award of degree in **B. Tech CSE** of **Graphic Era Deemed University, Dehradun** during the academic year **2022-2023**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated. The project has been approved as it satisfies the academic requirements associated with the degree mentioned.

Dr. Devesh Pratap Singh

HOD (Computer Science)

Project Repo:

AdityaBahl/Face-Detection-using-OpenCV (github.com)

Problem Statement:

Making a Face Detection Application using OpenCV and Tkinter.

Motivation:

There are several motivations for using face detection with OpenCV, some of which are:

Automating tasks: It can be used to automate tasks that require identifying and processing images of human faces. For example, a face detection system could be used to automatically tag friends in a photo album or to index and search a database of images for specific people.

Security and surveillance: Face detection can be used in security and surveillance applications to identify and track individuals in real-time or in recorded video footage. This can help to detect and prevent crimes, as well as to gather evidence for investigations.

Human-computer interaction: Face detection can be used to enable more natural and intuitive forms of human-computer interaction, such as facial recognition login systems or facial expression analysis for emotional computing.

Research and development: Face detection is a widely studied problem in computer vision and machine learning, and there is ongoing research and development in this area to improve the accuracy and efficiency of face detection algorithms. OpenCV provides an open source platform for researchers and developers to experiment with and contribute to the advancement of face detection technology.

Abstract

Face detection is the process of identifying and locating human faces in images or videos. OpenCV (Open Source Computer Vision) is a popular computer vision library that includes various functions for image and video processing, including face detection.

The face detection function in OpenCV uses a pre-trained classifier to detect faces in images. The classifier is trained on a large dataset of images containing faces and non-faces, and it uses machine learning techniques to learn common patterns in images that are characteristic of faces. When the classifier is applied to a new image, it looks for these patterns and returns a list of rectangles, each representing a face in the image.

To use the face detection function in OpenCV, you need to first import the necessary libraries, such as cv2 for OpenCV and numpy for numerical processing. Then you can load the image using the imread function from the cv2 library and convert it to grayscale using the

cvtColor function. Next, you can load the pre-trained face detection classifier using the CascadeClassifier function and use the detectMultiScale method to detect faces in the image. Finally, you can iterate through the list of rectangles and draw a rectangle around each face using the rectangle function.

There are many applications for face detection, including automating tasks, security and surveillance, human-computer interaction, and research and development. OpenCV provides an open source platform for researchers and developers to experiment with and contribute to the advancement of face detection technology.

Why OpenCV?

There are several reasons why OpenCV is a good choice for face detection:

1. OpenCV is open source: OpenCV is an open source library, which means that it is free to use and distribute. This makes it an attractive option for developers who want to use it in their projects without incurring any licensing fees.
2. OpenCV is widely used: OpenCV is a popular library with a large user base and a active developer community. This means that there is a wealth of documentation, tutorials, and examples available online, as well as a strong support network of users and developers who can help with any questions or issues that may arise.
3. OpenCV is fast and efficient: OpenCV is optimized for real-time performance and is designed to be fast and efficient. This makes it a good choice for applications that require fast face detection, such as security and surveillance systems or human-computer interaction.
4. OpenCV has a wide range of features: In addition to face detection, OpenCV includes a wide range of other features for image and video processing, such as object detection, image segmentation, and image stitching. This makes it a versatile and powerful library for a wide range of computer vision tasks.

   Overall, OpenCV is a good choice for face detection due to its open source nature, wide user base, fast performance, and extensive feature set.

Methods And Tools Required

To use the face detection function in OpenCV, you need to first import the necessary libraries, such as **cv2** for OpenCV and **numpy** for numerical processing. Then you can load the image using the **imread** function from the **cv2** library and convert it to grayscale using the **cvtColor** function. Next, you can load the pre-trained face detection classifier using the **CascadeClassifier** function and use the **detectMultiScale** method to detect faces in the image. Finally, you can iterate through the list of rectangles and draw a rectangle around each face using the **rectangle** function.

Steps Or Algorithm

Here are the steps to detect faces in an image using OpenCV:

1. Import the necessary libraries, such as **cv2** for OpenCV and **numpy** for numerical processing.

2. Load the image using the **imread** function from the **cv2** library. This function returns a NumPy array representing the image.
3. Convert the image to grayscale using the **cvtColor** function from the **cv2** library. This is typically done because face detection algorithms are more sensitive to grayscale images.
4. Load the pre-trained face detection classifier using the **CascadeClassifier** function from the **cv2** library. This classifier is trained to recognize common patterns in images that are characteristic of faces.
5. Use the **detectMultiScale** method of the **CascadeClassifier** object to detect faces in the image. This method returns a list of rectangles, each representing a face in the image.
6. Iterate through the list of rectangles and draw a rectangle around each face using the **rectangle** function from the **cv2** library. You can specify the color and thickness of the rectangle using the **color** and **thickness** parameters.
7. Display the image using the **imshow** function from the **cv2** library and wait for the user to close the window using the **waitKey** function.

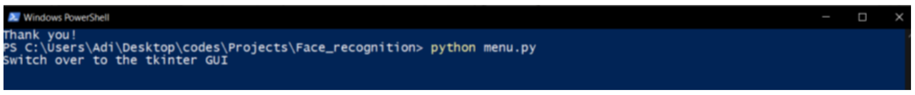
Importing Libraries

Python libraries like :-

- Tkinter :- used for GUI(Graphical User Interface)
- cv2 :- OpenCV used for face detection
- subprocess:- used to run system level scripts

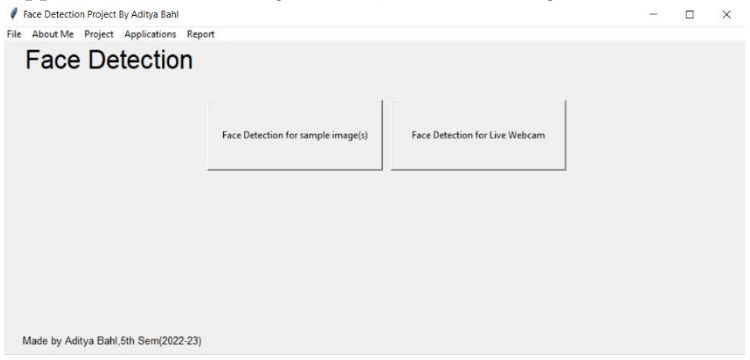
Run Python File(After Installing the above listed libraries)

Run Python File from Power Shell



```
Thank you!
PS C:\Users\Adi\Desktop\codes\Projects\Face_recognition> python menu.py
Switch over to the tkinter GUI
```

Application(made using Tkinter) starts running


Face Detection Project By Aditya Bahl
File
About Me
Project
Applications
Report
Face Detection
Face Detection for sample image(s)
Face Detection for Live Webcam
Made by Aditya Bahl,5th Sem(2022-23)
.

There are 2 Options:

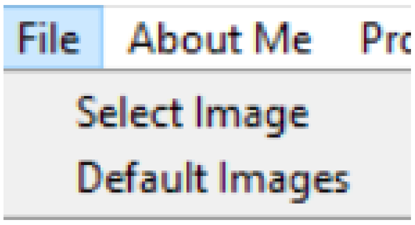
Face Detection for sample image(s): used for image processing

Face Detection for Live Webcam: used for live on the spot webcam based facial detection

Menu Section

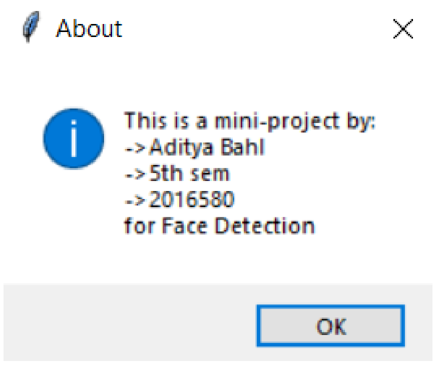
‘File’ Menu Section

Allows the User to select whether the default 3 images be used via the *Default Images* sub-section or select any image with faces given by the User via the *Select Image* sub-section.



‘About’ Menu Section

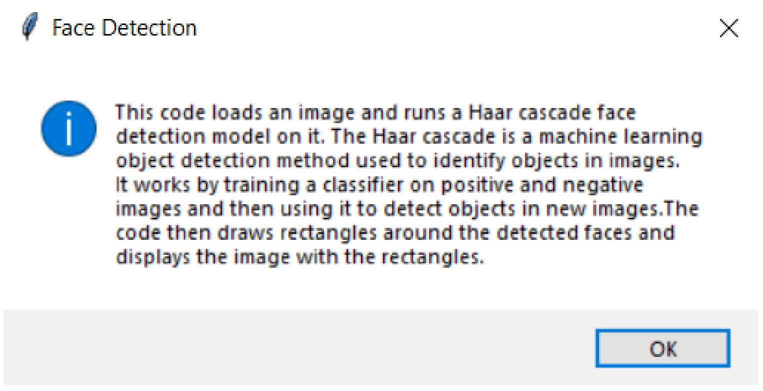
Gives some information about the developer(can be modified during or before deployment)



This is a mini-project by:
->Aditya Bahl
->5th sem
->2016580
for Face Detection

‘Project’ Menu Section

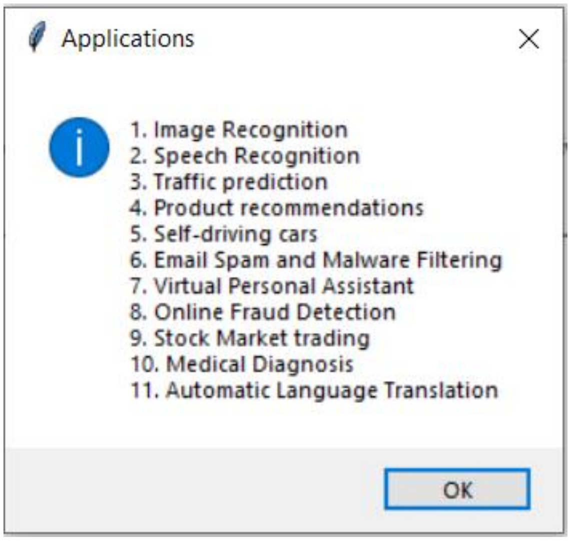
Gives some information about Haar Cascade Face detection model(used in this project for detection, provided by OpenCV and this information can be modified during or before deployment)



This code loads an image and runs a Haar cascade face detection model on it. The Haar cascade is a machine learning object detection method used to identify objects in images. It works by training a classifier on positive and negative images and then using it to detect objects in new images.The code then draws rectangles around the detected faces and displays the image with the rectangles.

‘Application’ Menu Section

Gives some information about various applications for Face Detection Programs(this information can be modified during or before deployment)


Applications

1. Image Recognition
2. Speech Recognition
3. Traffic prediction
4. Product recommendations
5. Self-driving cars
6. Email Spam and Malware Filtering
7. Virtual Personal Assistant
8. Online Fraud Detection
9. Stock Market trading
10. Medical Diagnosis
11. Automatic Language Translation

‘Report’ Menu Section

Opens the project Report in a PDF Viewer via system calls(this information can be modified during or before deployment)

OpenCV

OpenCV (Open Source Computer Vision) is a popular computer vision library that includes various functions for image and video processing. One of the functions available in OpenCV is face detection, which allows you to detect faces in images and videos.

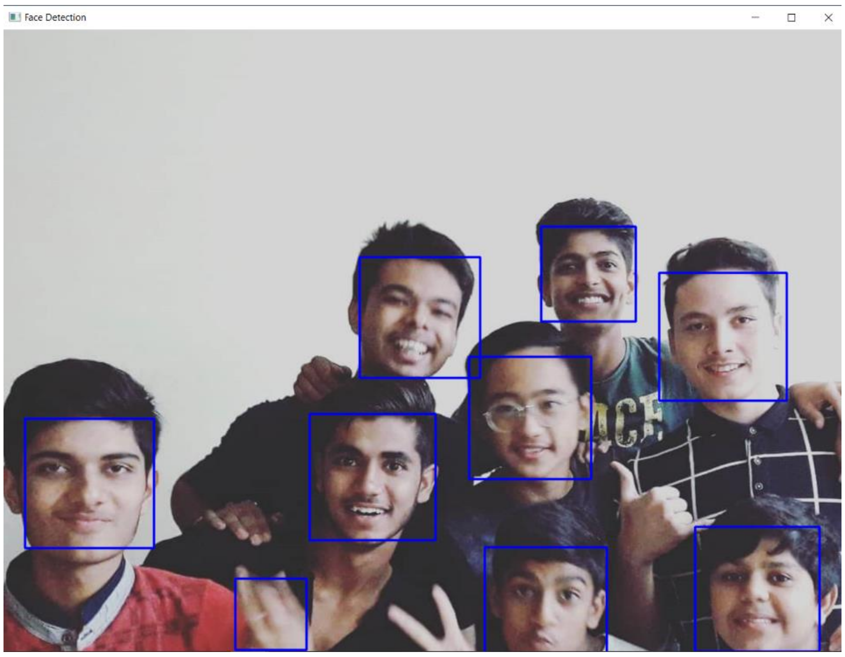
The face detection function in OpenCV uses a pre-trained classifier to detect faces in images. The classifier is trained on a large dataset of images containing faces and non-faces, and it uses machine learning techniques to learn common patterns in images that are characteristic of faces. When the classifier is applied to a new image, it looks for these patterns and returns a list of rectangles, each representing a face in the image.

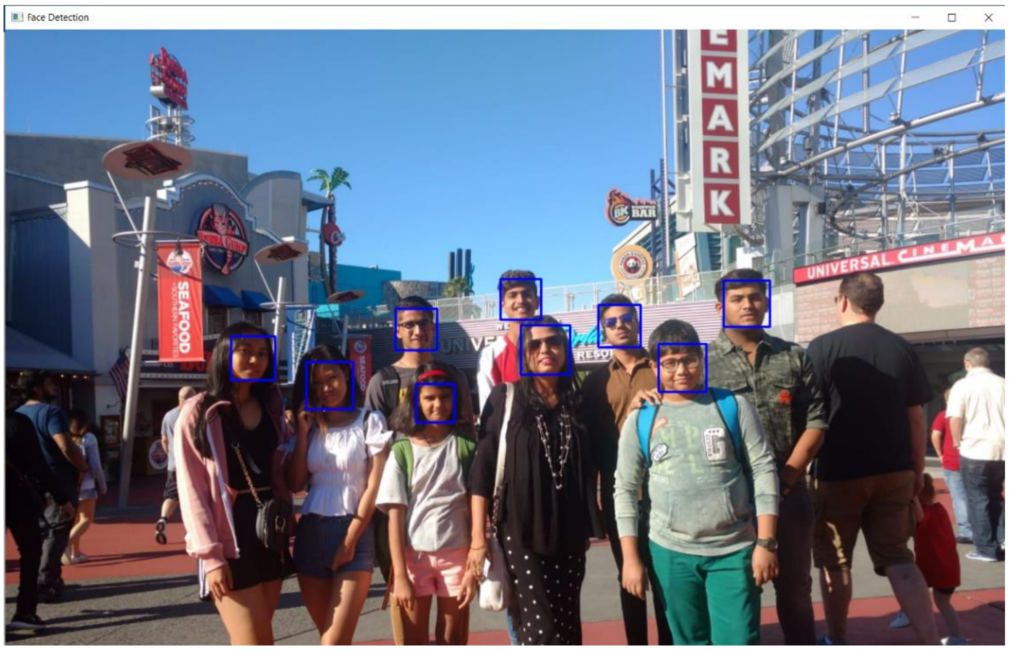
OpenCV is helpful in face detection in several ways:

1. OpenCV provides a pre-trained classifier for face detection
2. OpenCV provides functions for loading and processing images
3. OpenCV provides functions for visualizing and displaying results

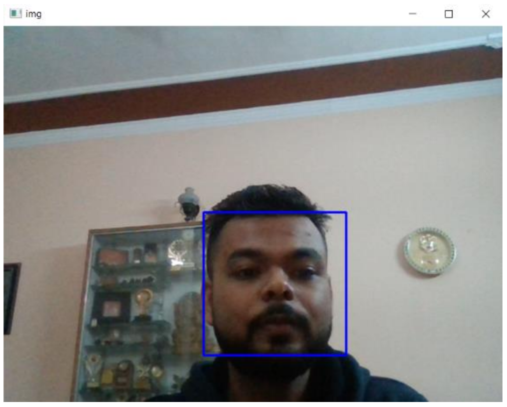
Here Are some results:

Selected Images:





Web Cam Result:



PROJECT LIMITATIONS AND CHALLENGES

There are a few limitations to using OpenCV for face detection:

1. Performance depends on the quality of the classifier: The performance of the face detection function in OpenCV depends on the quality of the classifier used. If the classifier is not trained well or if it is not well-suited to the images it is being applied to, it may not perform as well. This can lead to false positives (detecting a face where there is none) or false negatives (failing to detect a face that is present).

2. Limited ability to customize the classifier: The face detection classifier included with OpenCV is pre-trained and cannot be easily customized. This means that you cannot easily adjust the classifier to better suit your specific needs or to improve its performance on a particular dataset.

3. May not work well on images with low resolution or poor lighting: Face detection algorithms can be sensitive to the resolution and quality of the input images. If the images are low resolution or have poor lighting, the classifier may not perform as well.

4. May not work well on faces with unusual or extreme appearance: Face detection algorithms are designed to detect "typical" faces, and they may not perform as well on faces with unusual or extreme appearance, such as very large or small faces, or faces with unusual facial features or expressions.

Overall, while OpenCV is a powerful tool for face detection, it does have some limitations in terms of performance, customization, and robustness to variations in the input images. These limitations should be taken into consideration when using OpenCV for face detection.

CONCLUSION

I learned many new techniques and enjoyed the process. There were a lot of problems, but removing errors is what we must learn. The project may not give accurate results in some cases due to the reasons as mentioned above, but there are quite a few correct solutions too, I will explore this domain further.

REFERENCES

- Stack Overflow - Where Developers Learn, Share, & Build Careers
- Home - OpenCV