

IMPLEMENTATION OF FACE RECOGNITION AND DETECTION SYSTEM USING PYTHON.



VIKASH KUMAR GAUTAM

Face detection and face recognition.

What is face detection?

Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belongs to a given class. Examples include upper torsos, pedestrians, and cars. Face detection simply answers two questions, 1. are there any human faces in the collected images or video? 2. where is the located?

Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. The image matches the image stored in the database. Any facial feature changes in the database will invalidate the matching process.

A reliable face-detection approach based on the genetic algorithm and the eigenface technique:

Firstly, the possible human eye regions are detected by testing all the valley regions in the gray-level image. Then the genetic algorithm is used to generate all the possible face regions which include the eyebrows, the iris, the nostril, and the mouth corners.

Each possible face candidate is normalized to reduce both the lighting effect, which is caused by uneven illumination; and the shirring effect, which is due to head movement. The fitness value of each candidate is measured based on its projection on

the eigenfaces. After several iterations, all the face candidates with a high fitness value are selected for further verification.

At this stage, the face symmetry is measured and the existence of the different facial features is verified for each face candidate.

Few Applications of face detection:

- Facial motion capture
- Facial recognition
- Photography
- Marketing
- Emotional Inference
- Lip Reading

What is face recognition?

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

Development began on similar systems in the 1960s, beginning as a form of computer application. Since their inception, facial recognition systems have seen wider uses in recent times on smartphones and other forms of technology, such as robotics.

Because computerized facial recognition involves the measurement of a human's physiological characteristics, facial recognition systems are categorized as biometrics. Although the accuracy of facial recognition systems as biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless process. Facial recognition systems have been deployed in advanced human-computer interaction, video surveillance, and automatic indexing of images.

Few applications of face recognition:

- ID Verification
- Social Media
- Deployment in security services
- Deployment in retail stores

Related Works

Real-Time Objects Recognition Approach for Assisting Blind People:

Sunil Vaidya, Niti Shah, and Radha Shankarmani have proposed research entitled “Real-Time Objects Recognition Approach for Assisting Blind People” (in 2010) on object recognition for assisting blind people. In their study, two cameras placed on blind person's glasses, GPS free service, and ultrasonic sensors are employed to provide information about the surrounding environment. Object detection is used to find objects in the real world such as faces, bicycles, chairs, doors, or tables that are common in the scenes of a blind. Here, GPS service is used to create groups of objects based on their locations, and the sensor detects an obstacle at a medium to long distance. The descriptor of the Speeded-Up Robust Features (SURF) method is optimized to perform the recognition. The use of two cameras on glasses can be sophisticated.

Wearable Object Detection System for the Blind:

A.Dionisi, E. Sardini, and M. Serpelloni have proposed research entitled “Wearable Object Detection System for the Blind” (in 2012) on wearable object detection systems for assisting blind people. In their research, the RFID device is designed as a support for the blind for the disclosure of objects; especially, it is developed for searching the medicines in a cabinet at home. This device can provide information about the distance of a defined object, and how near or far it is and simplifies the search. For identifying the medicines, the device can provide the user with an

acoustic signal to find the desired product as soon as possible. The measure of the distance, in particular the movement of the antenna concerning the tag, is made using the RSSI value. This application uses the RSSI (Received Strength Signal Indicator) value, to measure the power of the received signal of the tag.

Smart Obstacle Detector for Blind Person:

Faheem Ahmed, Habib Ahmed, and Er Zakir Ahmed Shaikh have proposed research entitled “Smart Obstacle Detector for Blind Person” (in 2014) on smart obstacle detectors for assisting blind people. In their research, he focuses on giving information about what are the different types of obstacles in front of the user, their size, and their distance from the user. MATLAB Software is used for signal processing. The camcorder is used for recording videos. Video processing methods are used after that. The output of this system not only gives output in audio format but also vibration. A vibrating motor has been connected with an ultrasonic sensor. The ultrasonic sensor detects objects coming in its range and this makes the vibrating motor vibrate. The use of Camcorder, a stick with an Ultrasonic sensor makes this system bulky and dependent on the stick.

Importance

The human visual system plays an important role in recognizing information regarding surroundings. Since visual signal provides more data than auditory information, visual signals are more effective than auditory signals when the human being perceives information.

However, in the case of blind people, the lack of visual information constrains them from recognizing information. For a blind person to recognize a subject around him depends on the subject to speak something.

Successful implementation would lead to assisting visually impaired people in daily life example recognizing a person in front of them, detecting obstacles on the path, identifying objects in the surroundings, etc.

With growing technology, innovations, and computation power, modern technology can be used to perform tasks that were never thought to be possible a decade ago. Artificial intelligence and machine learning have grown rapidly and are now used by most tech companies for various tasks.

With the help of AI and ML companies like Amazon can show recommended products based on the buying history of many people, Self-driving cars are possible because AI and ML are widely used in medical fields to diagnose various medical conditions.

In this project, we intend to develop a basic working model of a face recognition system combined with audio feedback.

Who's working and ideology can be used in detailed future implementations to aid visually disabled people in various tasks in their day-to-day life and wellbeing.

Role in society

This project could be of huge importance to visually challenged people in day-to-day life. Not only can it upgrade their sense of the surroundings as compared to traditional cane, but also can make them more informative and immersed in the surrounding, which will ensure security, ease in daily life, assurance to them of the surroundings, and provide overall better life by using modern technologies which include modern software.

Successful implementation could upgrade the quality of people who eagerly need these kinds of developments to overcome the limitations brought by their disability.

SOFTWARE REQUIREMENTS

1. Anaconda:-

Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment. Package versions in Anaconda are managed by the package management system, conda, which analyzes the current environment before executing an installation to avoid disrupting other frameworks and packages.

The Anaconda distribution comes with over 250 packages automatically installed. Over 7500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI (graphical user interface), Anaconda Navigator, as a graphical alternative to the command-line interface. Anaconda Navigator is included in the Anaconda distribution and allows users to launch applications and manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages, install them in an environment, run the packages and update them.

2. Jupyter Notebook:-

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The “notebook” term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. According to the official website of Jupyter, Project Jupyter exists to develop open-source software, open standards, and services for interactive computing across dozens of programming languages.

Jupyter Book is an open-source project for building books and documents from computational material. It allows the user to construct the content in a mixture of Markdown, an extended version of Markdown called MyST, Maths & Equations using MathJax, Jupyter Notebooks, and reStructuredText, the output of running Jupyter Notebooks at build time.

Multiple output formats can be produced (currently single files, multipage HTML web pages, and PDF files). Now that we have a brief understanding of the concept of Project Jupyter and the Jupyter Notebooks, and we have established that these Notebooks are quite revolutionary in the modern ages, let us proceed to understand the more in-depth details of this amazing interactive environment in the next sections.

3. Python Libraries:-

- **Open CV:-**

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, and stitch images together to the product high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 18 million. The library is used extensively by companies, research groups, and governmental bodies. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, and Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV.

OpenCV's deployed uses span the range from stitching Streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detecting swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available.

A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

- **Face Recognition Library:-**

Recognize and manipulate faces from Python or the command line with the world's simplest face recognition library.

Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple `face_recognition` command-line tool that lets you do face recognition on a folder of images from the command line!

Some Special Features of Face Recognition:-

- It can Find all the faces that appear in a picture.
- It can Get the locations and outlines of each person's eyes, nose, mouth, and chin.
- Recognize who appears in each photo.

• Pyttsx3:-

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the `pyttsx3.init()` factory function to get a reference to a `pyttsx3`. Engine instance. is a very easy-to-use tool that converts the entered text into speech. The `pyttsx3` module supports two voices first is female and the second is male which is provided by "sapi5" for windows. It supports three TTS engines:

- * sapi5 – SAPI5 on Windows
- * nsss – NSSpeechSynthesizer on Mac OS X
- * speak – eSpeak on every other platform

- **OS:-**

This module provides a portable way of using operating system-dependent functionality. If you just want to read or write a file see `open()`, if you want to manipulate paths, see the `os.path` module, and if you want to read all the lines in all the files on the command line see the `fileinput` module. For creating temporary files and directories see the `tempfile` module, and for high-level file and directory handling see the `shutil` module.

Notes on the availability of these functions:

- The design of all built-in operating system-dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function `os.stat(path)` returns stat information about the *path* in the same format (which happens to have originated with the POSIX interface).
- Extensions peculiar to a particular operating system are also available through the `os` module, but using them is of course a threat to portability.
- All functions accepting path or file names accept both bytes and string objects and result in an object of the same type if a path or file name is returned.
- On VxWorks, `os.open`, `os.fork`, `os.execv` and `os.spawn*p*` are not supported.

- **Glob:-**

Glob is a general term used to define techniques to match specified patterns according to rules related to the Unix shell. Linux and Unix systems and shells also support glob and also provide function glob() in system libraries.

In Python, the glob module is used to retrieve **files/pathnames** matching a specified pattern. The pattern rules of glob follow standard Unix path expansion rules. It is also predicted that according to benchmarks it is faster than other methods to match pathnames in directories. With glob, we can also use wildcards ("*, ?, [ranges]) apart from exact string search to make path retrieval more simple and convenient.

- **Numpy:-**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is a Python package. It stands for 'Numerical Python. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. **Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created the NumPy package by incorporating the features of Numarray into the Numeric package. There are many contributors to this open-source project.

Using NumPy, a developer can perform the following operations –

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Matplotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, the Python alternative to MatLab is now seen as a more modern and complete programming language. It is open-source, which is an added advantage of NumPy. The most important object defined in NumPy is an N-dimensional array type called **ndarray**. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index. Every item in an array takes the same size as the block in the memory. Each element in arrays an object of the data-type object (called **dtype**).

- **Cmake:-**

CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools was created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

- **Visual Studio:-**

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity). Begin your journey with VS Code with these introductory videos.

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing your ideas. At its heart, Visual Studio Code features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS

Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization, and community-contributed keyboard shortcut mappings let you navigate your code with ease.

For serious coding, you'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring.

And when the coding gets tough, the tough get debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so you can work with source control without leaving the editor including viewing pending changes diffs.

4. Detection System:

The main goal of the detection system is to detect faces. When a face is detected, a bounding box is drawn around it. This bounding box is used to extract and save the face of a person by cropping the area inside it. Once the face is detected, the next detection is performed after a delay to avoid overwhelming the user with constant detection notifications.

Faces detected by the system are limited to frontal faces since the application is designed to identify persons in front of the user. Detection is accomplished

through the use of the object detector built into the OpenCV Android library. The object detector uses a cascade classifier to detect faces. The cascade classifier is made up of a cascade of boosted classifiers with Haar-like features. Training of the classifier is done using a set of positive and negative images.

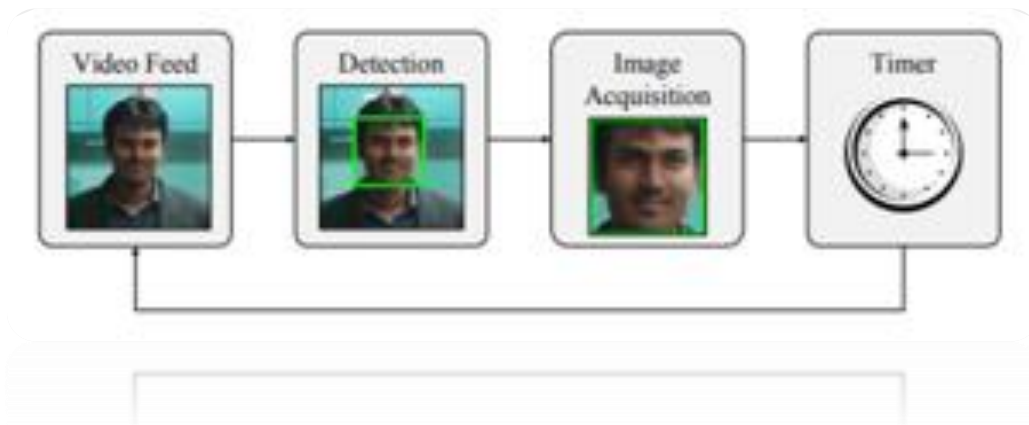


Fig2.Detecting Procedure Of Face.

5. Recognition System:

The recognition system complements the detection system by identifying the person using the detected face. This is done by running the recognition program which will search the internal database for a match using the saved face image as input. The recognition program will utilize C++ functions from the OpenCV library to identify people. This is due to limited support for Java-based face recognition in the OpenCV Android library. To enable the use of the C++ functions, the Java Native Interface4 (JNI) will be used by the application. The recognition program will use the Local Binary Patterns Histograms (LBPH) algorithm to perform recognition. It will use the Local Binary Patterns (LBP) method which creates a summary of the local structure in an image by comparing each pixel with its adjacent pixel. If the intensity of the Centre pixel is greater or equal to its adjacent pixel, it is assigned a value of 1 or 0 if not. The resulting binary numbers for each pixel are called Local Binary Patterns. In this approach, the LBP image is divided into local regions and a histogram is extracted from each image. The feature vector is then obtained by concatenating the local histograms which are called Local Binary Patterns Histograms. The use of the LBPH approach allows for fast feature extraction.

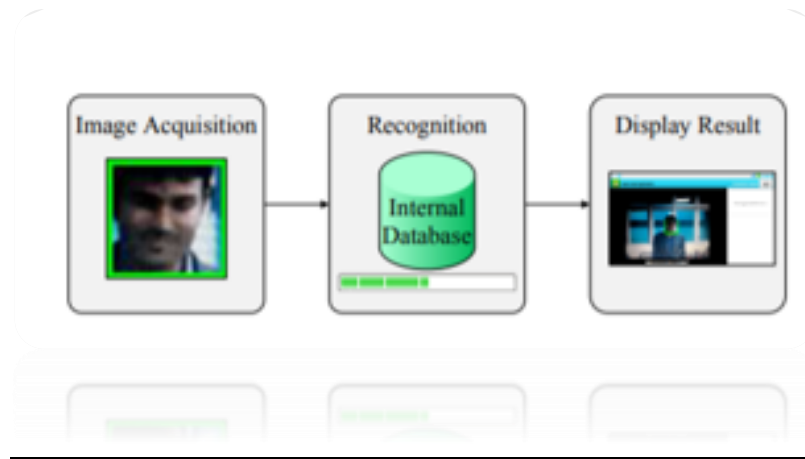


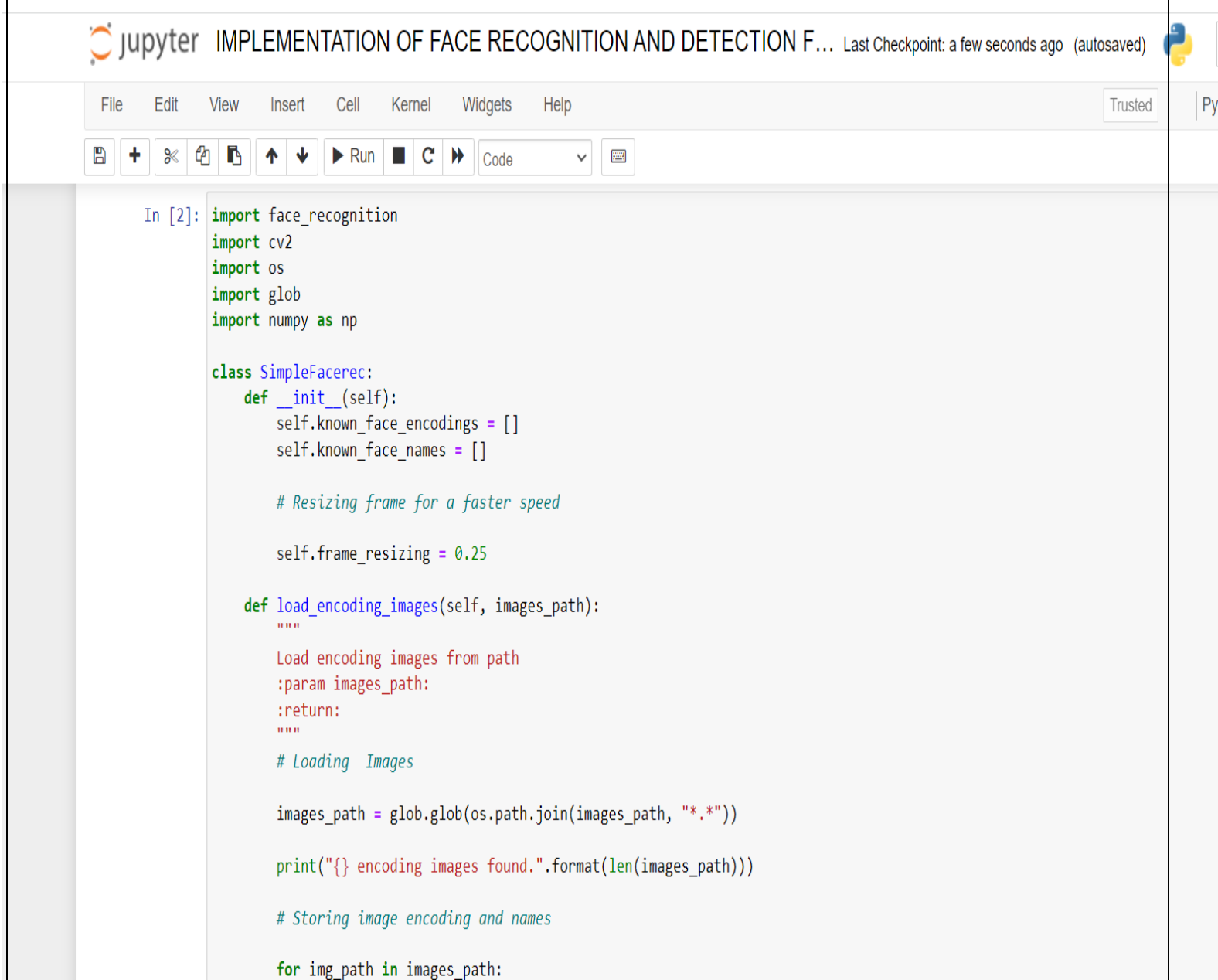
Fig3.The recognition procedure of Face.

How does it work :

1. Install all the software from the list into your system.
2. Make a folder in which you will store all the images of the persons you want to recognize.
3. Copy the images of the person you want to recognize into that folder.
4. Copy the address of that folder and pass it inside the `SFR.load_encoding_images()` function as a string.
5. Turn the camera towards the person you want to recognize.
6. Run the code.
7. Automatically camera opens and detects the face from the camera.
8. Now, it will start matching from the folder in which we have stored all the images.

9. Now, if the match is found from the folder then the matched image name will be output in the form of audio and a frame showing your name onto the screen.
10. If the match is not found from the folder then the audio output will be “Unknown Person” and the frame on the screen will also show “Unknown Person”.

CODE IMPLEMENTATION OF FACE RECOGNITION SYSTEM FOR VISUALLY IMPAIRED PEOPLE:



The image shows a Jupyter Notebook interface. At the top, there's a title bar with the Jupyter logo and the text "IMPLEMENTATION OF FACE RECOGNITION AND DETECTION F...". To the right of the title bar, it says "Last Checkpoint: a few seconds ago (autosaved)". Below the title bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help. To the right of the menu bar is a "Trusted" button. Below the menu bar is a toolbar with various icons for file operations, navigation, and execution. The main area of the notebook contains a code cell with the following Python code:

```
In [2]: import face_recognition
import cv2
import os
import glob
import numpy as np

class SimpleFacerec:
    def __init__(self):
        self.known_face_encodings = []
        self.known_face_names = []

        # Resizing frame for a faster speed

        self.frame_resizing = 0.25

    def load_encoding_images(self, images_path):
        """
        Load encoding images from path
        :param images_path:
        :return:
        """
        # Loading Images

        images_path = glob.glob(os.path.join(images_path, "*.*"))

        print("{} encoding images found.".format(len(images_path)))

        # Storing image encoding and names

        for img_path in images_path:
```

```

img = cv2.imread(img_path)
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Get the filename only from the initial file path.

basename = os.path.basename(img_path)
(filename, ext) = os.path.splitext(basename)

# Get encoding

img_encoding = face_recognition.face_encodings(rgb_img)[0]

# Store file name and file encoding

self.known_face_encodings.append(img_encoding)
self.known_face_names.append(filename)
print("Encoding images loaded")

def detect_known_faces(self, frame):
    small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)

    # Find all the faces and face encodings in the current frame of video
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)

    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:

```




```

        # See if the face is a match for the known face(s)

        matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
        name = "Unknown Person"

        #use the known face with the smallest distance to the new face

        face_distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = self.known_face_names[best_match_index]
            face_names.append(name)

        # Convert to numpy array to adjust coordinates with frame resizing quickly

        face_locations = np.array(face_locations)
        face_locations = face_locations / self.frame_resizing
        return face_locations.astype(int), face_names

```

```
In [ ]: # to close all the windows and camera
```

```

cap.release()
cv2.destroyAllWindows()

```

```
In [ ]: import cv2
        from simple_facerec import SimpleFacerec
        import face_recognition
        import pytsx3

        # Encoding faces from a folder

        sfr = SimpleFacerec()
        sfr.load_encoding_images(r"C:\Users\vikas\OneDrive\Desktop\face_recognition\images/")

        # for Loading Camera

        cap = cv2.VideoCapture(0)

        while True:
            ret, frame = cap.read()

            # Detecting Faces

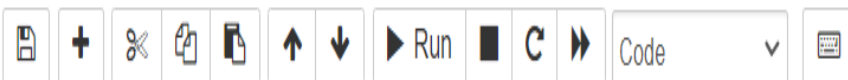
            face_locations, face_names = sfr.detect_known_faces(frame)
            for face_loc, name in zip(face_locations, face_names):
                y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]

                cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

            cv2.imshow("Frame", frame)
```



File Edit View Insert Cell Kernel Widgets Help



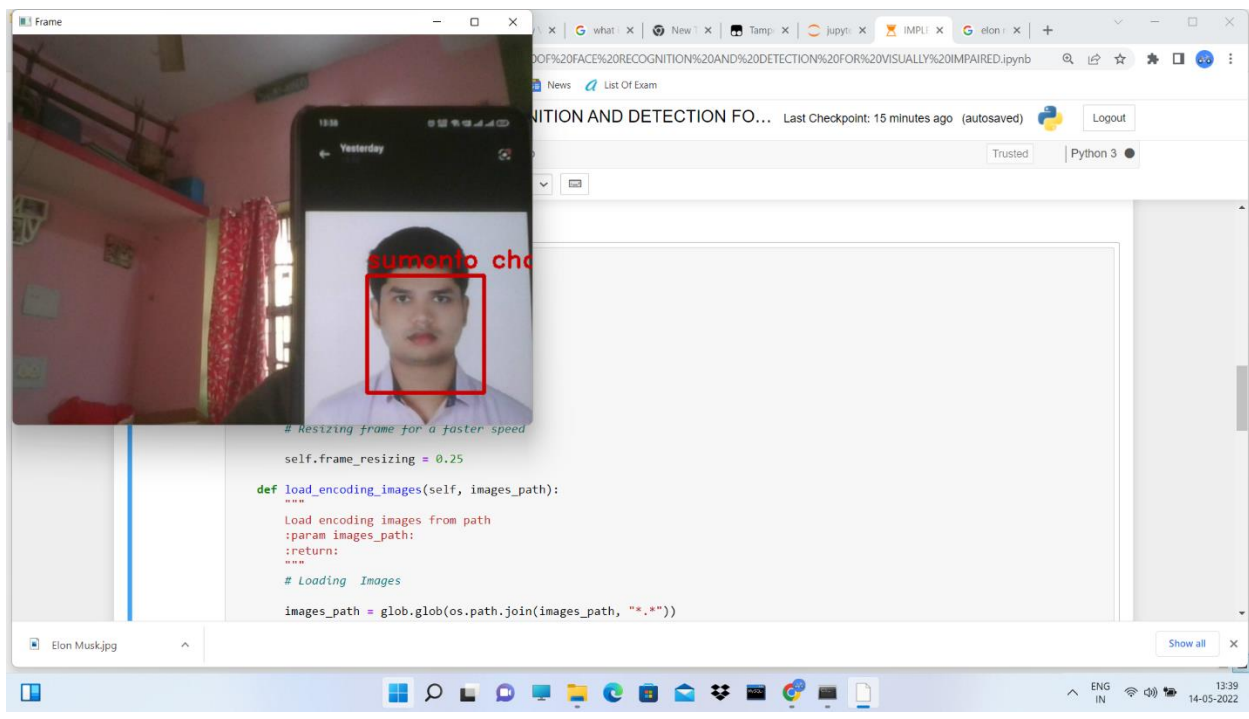
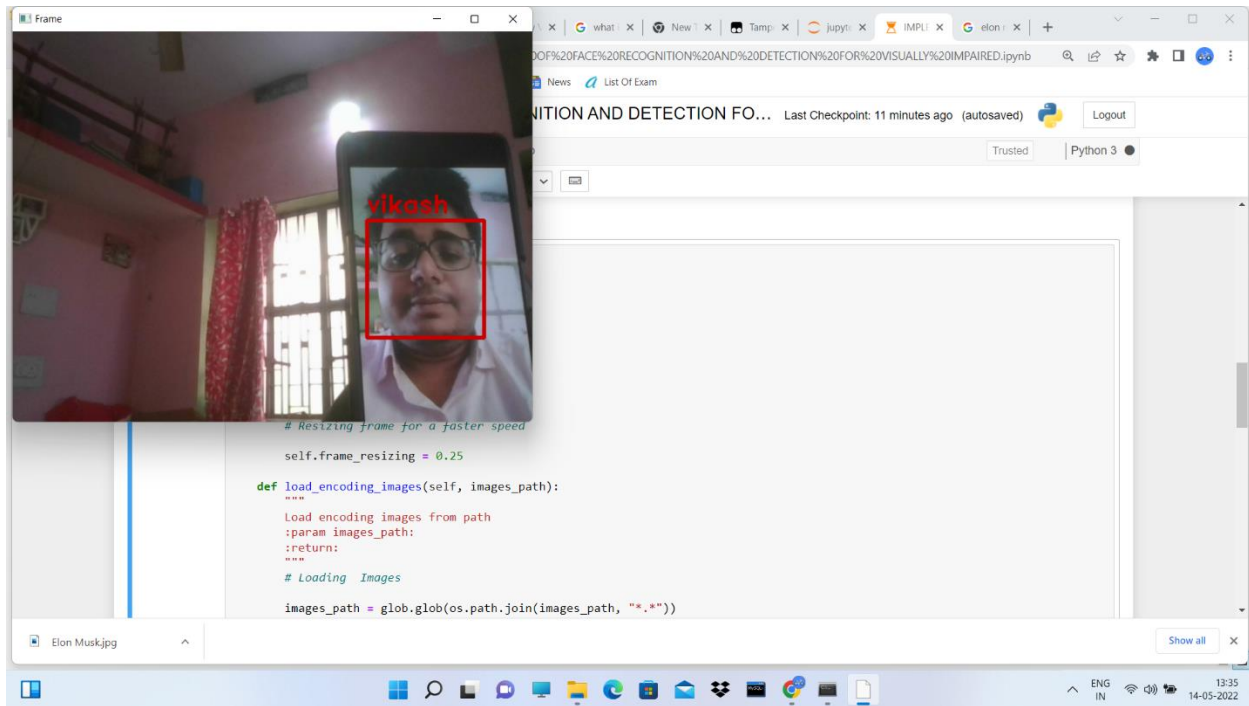
```
#converting name of detected person into audio.

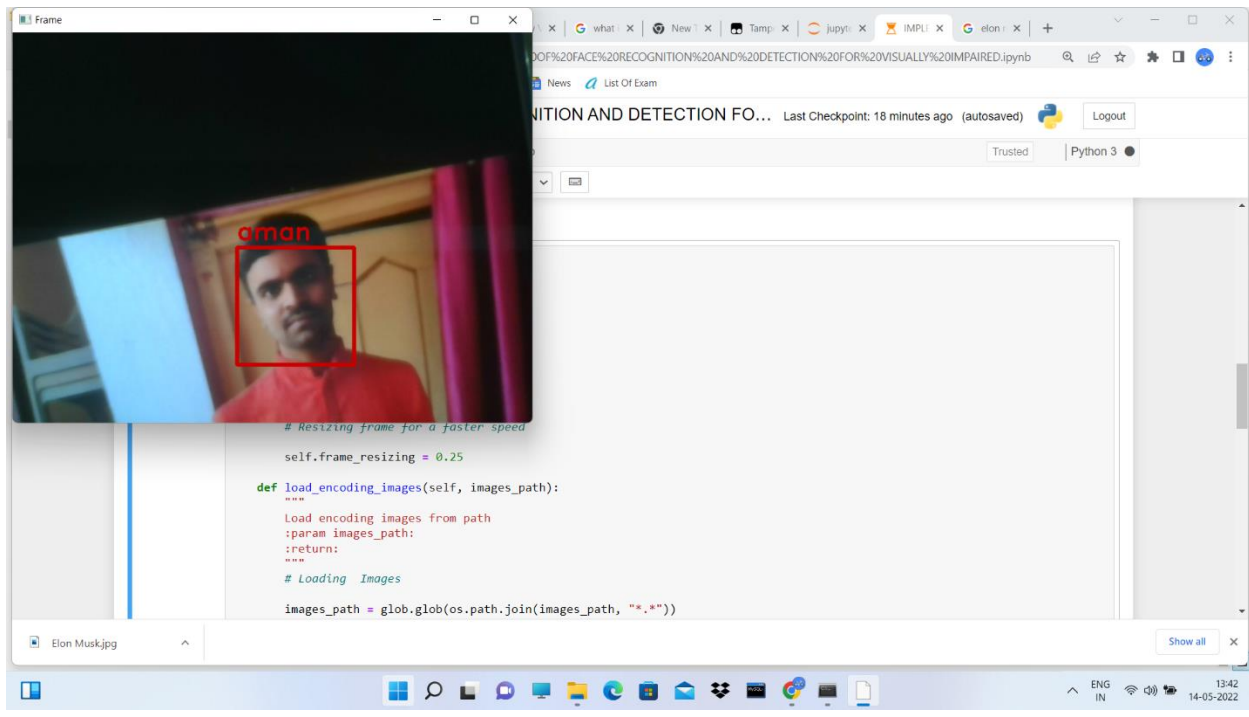
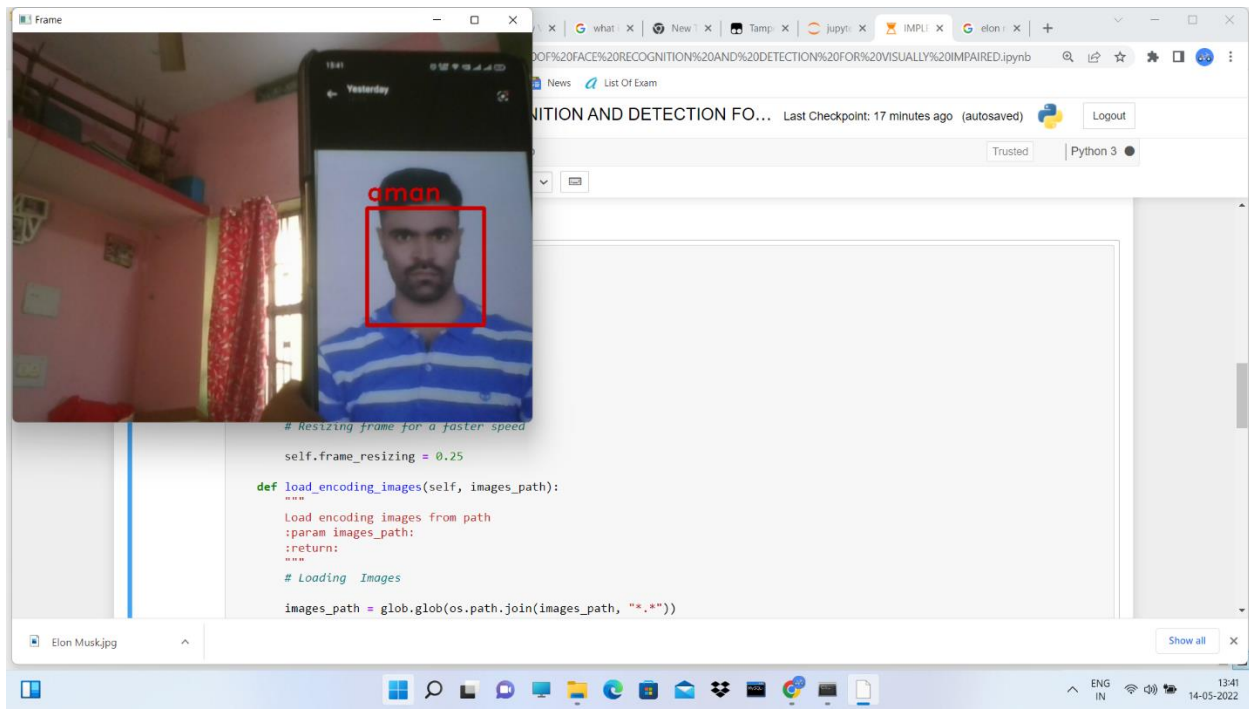
text_speech=pyttsx3.init()
text_speech.say(face_names)
text_speech.runAndWait()

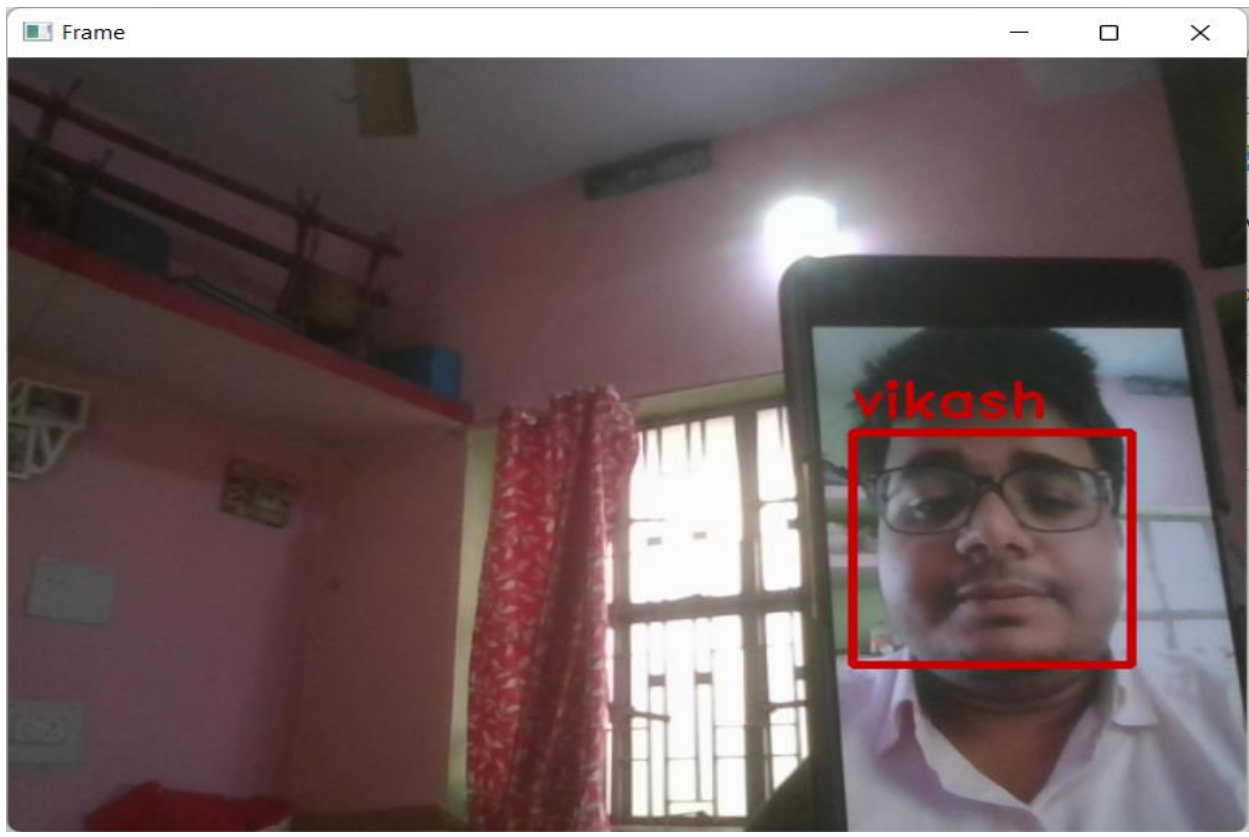
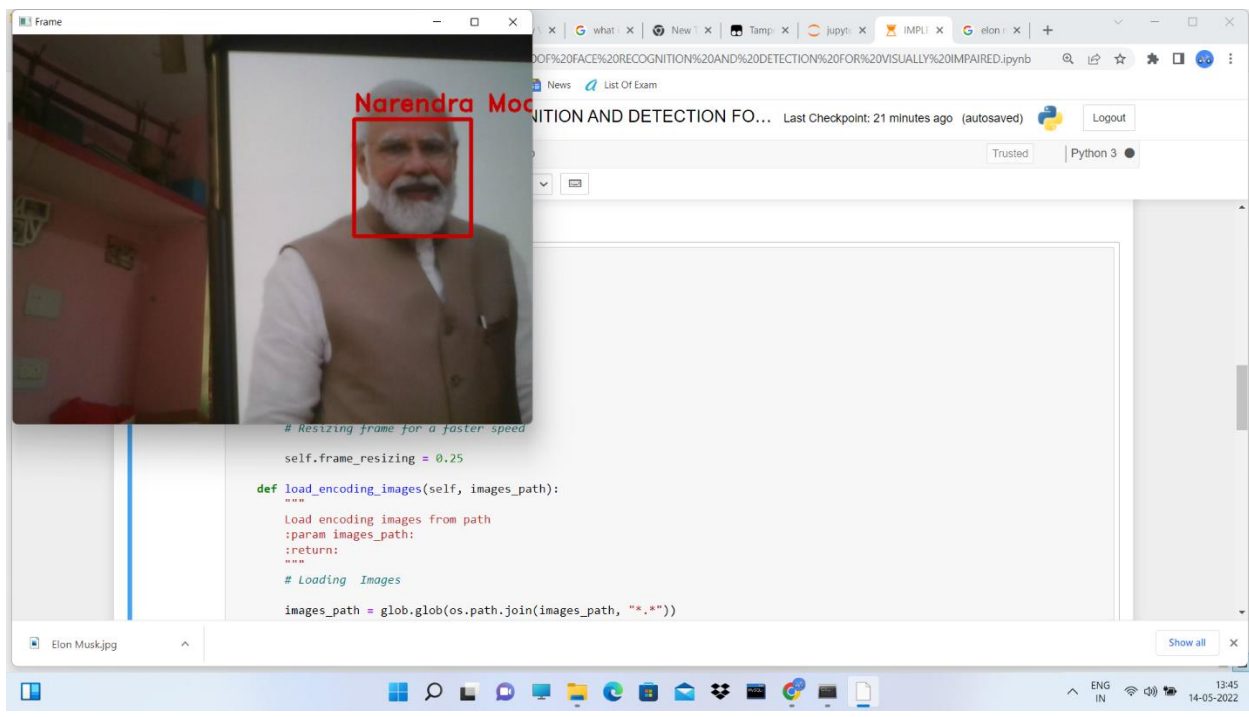
key = cv2.waitKey(1)
if key == 27:
    break
# to close all the windows and camera
cap.release()
cv2.destroyAllWindows()

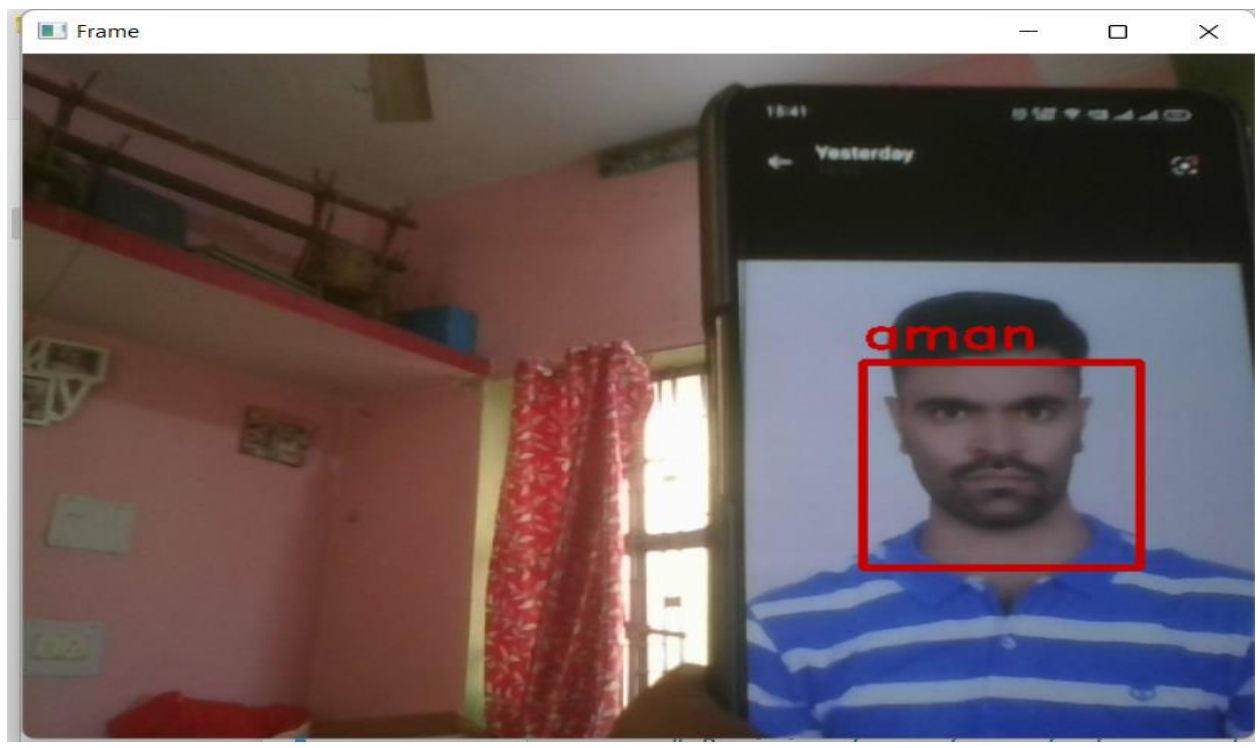
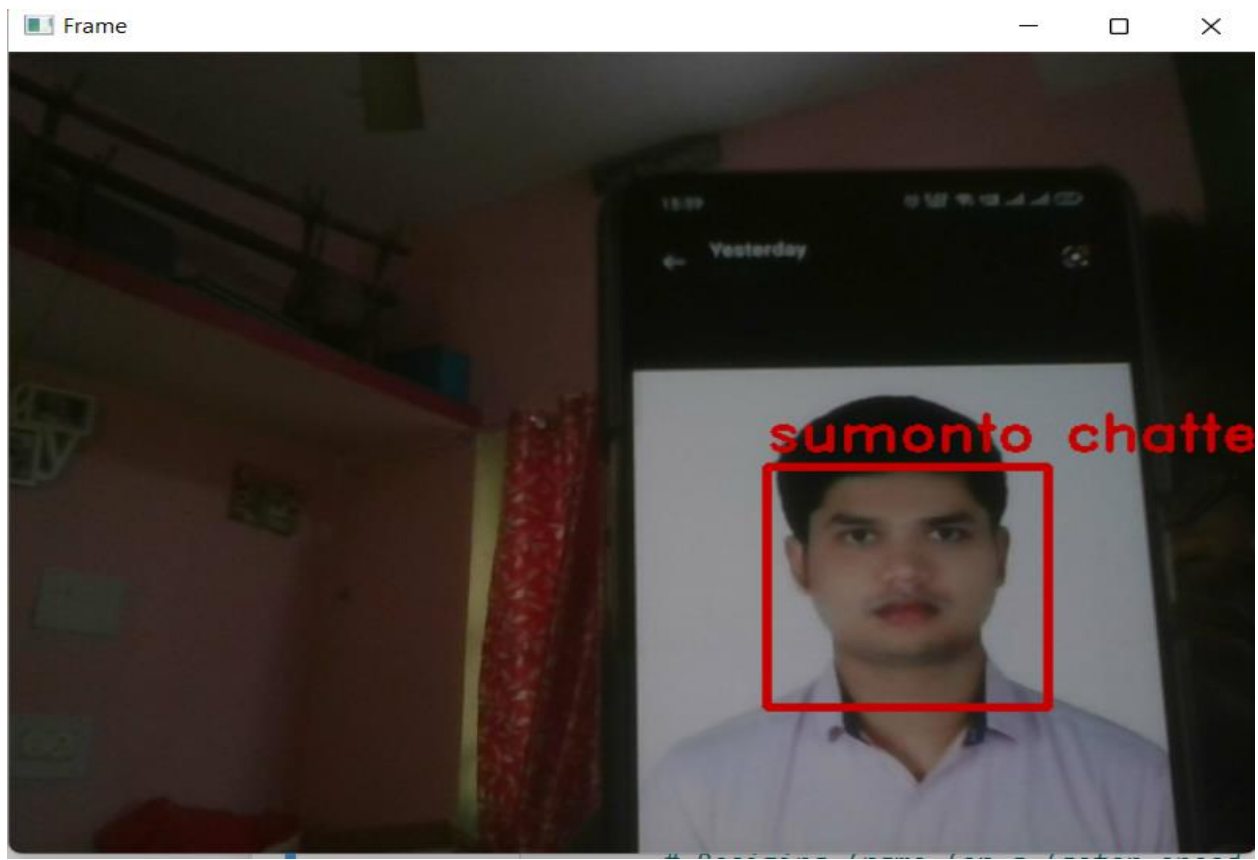
5 encoding images found.
Encoding images loaded
```

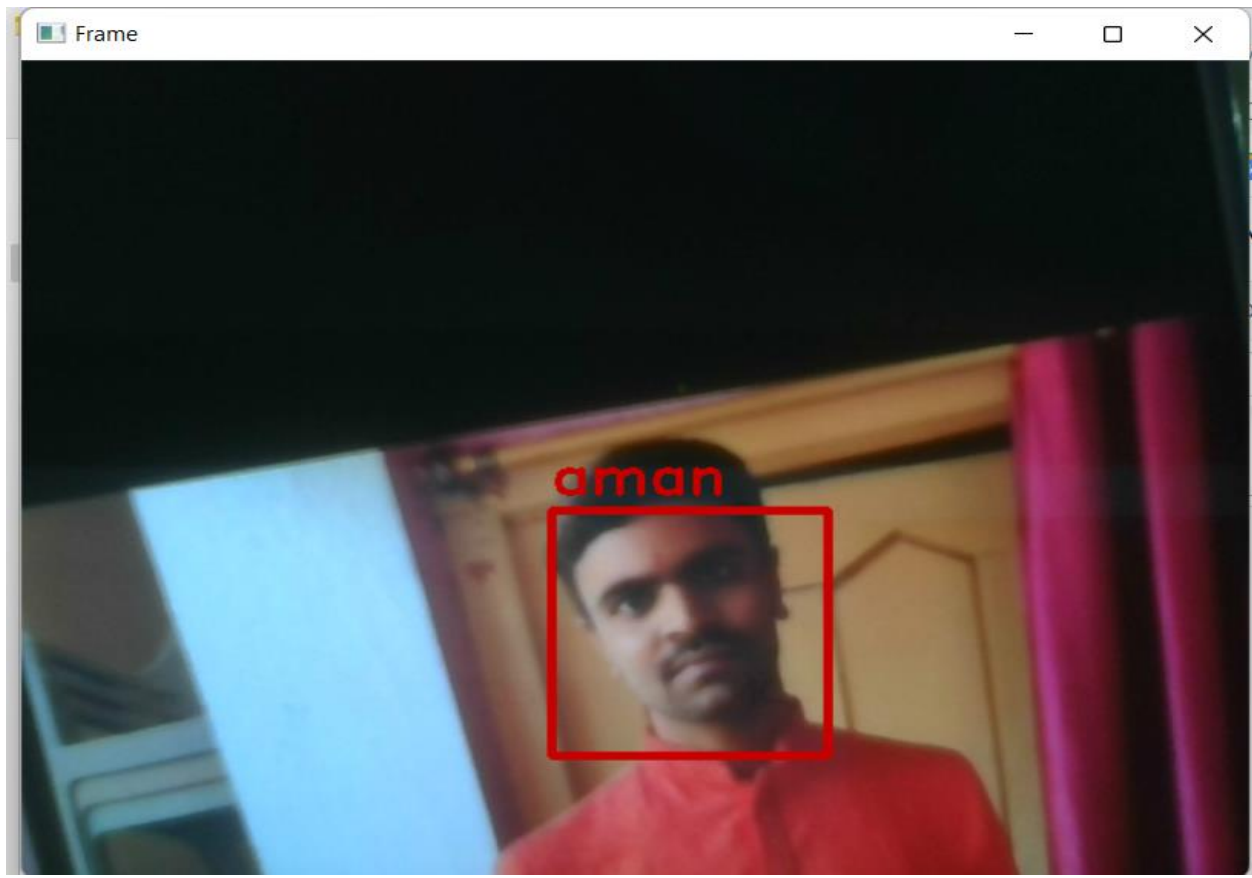
OUTPUT IMAGES OF THIS ALGORITHM:-

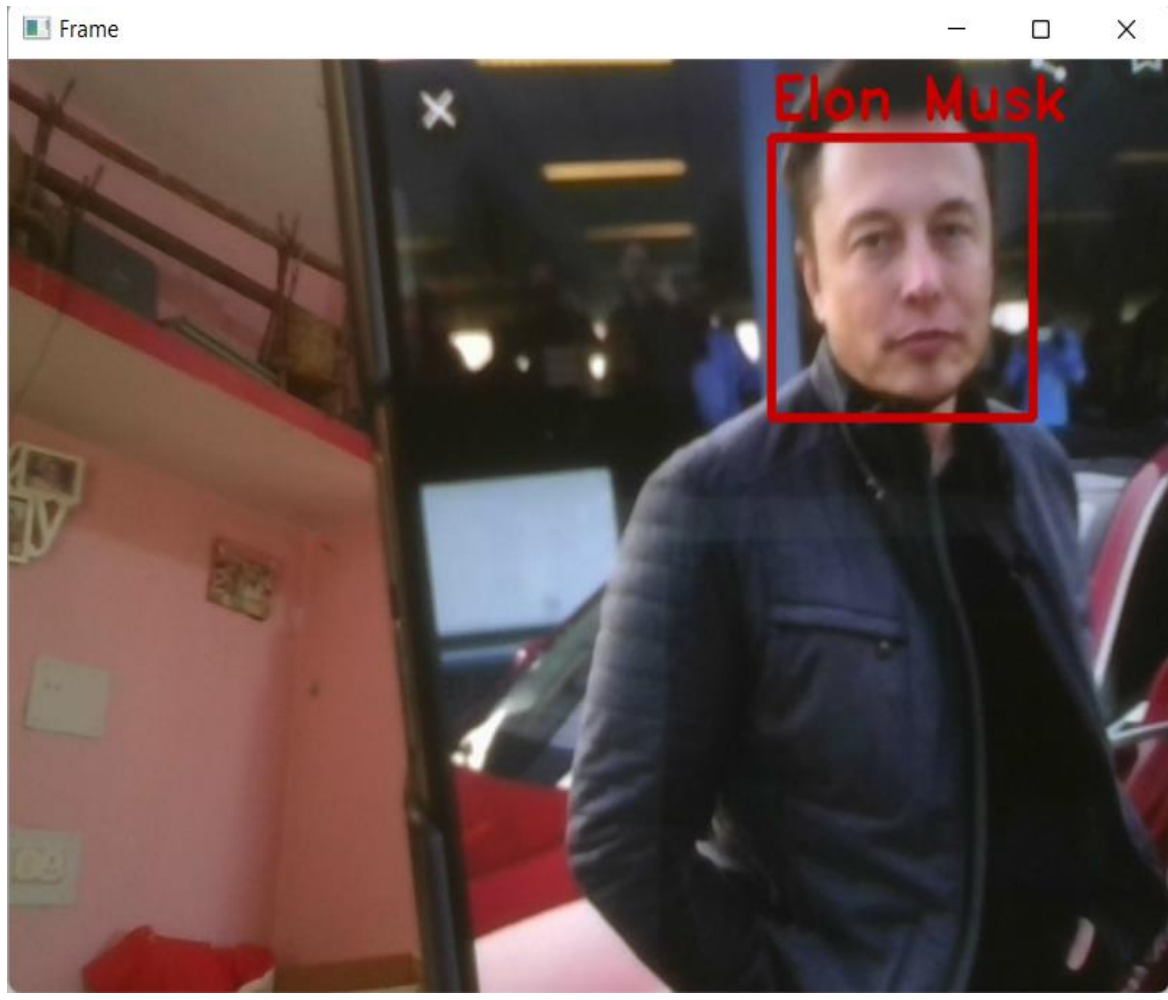












In the above output, the system not only recognizes the faces but also recites (audio feedback) the name of the person whose face is recognized.

REFERENCES:

2. Rui (Forest) Jiang, Qian Lin, Shuhui Qu,” Let Blind People See: Real-Time Visual Recognition with Results Converted to 3D Audio”, Stanford,2018.
3. N.Saranya, M.Nandinipriya, U.Priya,” Real-Time Object Detection for Blind People”, Bannari Amman Institute of Technology, Sathyamangalam, Erode. (India).
4. WHO. (2014, August) Visual impairment and blindness.
5. J. Falco, M. Idiago, A. Delgado, A. Marco, A. Asensio, and D. Cirujano, “Indoor navigation multi-agent system for the elderly and people with disabilities,” in Trends in Practical Applications of Agents and Multiagent Systems, ser. Advances in Intelligent and Soft Computing, Y. Demazeau, F. Dignum, J. Corchado, J. Bajo, R. Corchuelo, E. Corchado, F. Fernandez-Riverola, V. Julin, P. Pawlewski, and A. Campbell, Eds. Springer Berlin Heidelberg, 2010, vol. 71, pp. 437–442.
6. S. Willis and S. Helal, “Rfid information grid for blind navigation and wayfinding.” in ISWC, vol. 5, 2005, pp. 34 – 37.
7. S. K. Bahadir, V. Koncar, and F. Kalaoglu, “Wearable obstacle detection system fully integrated to textile structures for visually impaired people,” Sensors and Actuators A: Physical, vol. 179, no. 0, pp. 297 –311, 2012.
8. Google Inc. (2015) Google TalkBack. Accessed: 21-May-2015.
9. (2014) Android debug bridge. Accessed: 21-May-2015.

10. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition*, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1. IEEE, 2001, pp. 511 – 518.
11. R. Lienhart and J. May, "An extended set of haar-like features for rapid object detection," in *Image Processing*. 2002. Proceedings. 2002 International Conference on, vol. 1. IEEE, 2002, pp. I–900.
12. R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197 – 227, Jul. 1990.
13. T. Ahonen, A. Hadid, and M. Pietikainen, "Face recognition with local binary patterns," in *Computer Vision-ECCV 2004*. Springer, 2004.
14. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 4–4.
15. N. Ravi, J. Scott, L. Han, and L. Iftode, "Context-aware battery management for mobile phones," in *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, ser. PERCOM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 224 – 233.
16. X. Chen, Y. Chen, Z. Ma, and F. C. A. Fernandes, "How is the energy consumed in smartphone display applications?" in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '13. New York, NY, USA: ACM, 2013, pp. 3:1–3:6.
17. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21.

18. Y. Tian, X. Yang, and A. Arditi, "Computer vision-based door detection or accessibility of unfamiliar environments to blind persons," in Proceedings of the 12th International Conference on Computers Helping People with Special Needs, ser. ICCHP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 263 – 270.
19. J. C. Liter and H. H. Bülthoff, "An introduction to object recognition," *Zeitschrift für Naturforschung C-Journal of Biosciences*, vol. 53, no. 7, pp. 610 – 621, 1998.
20. M. A. Treiber, *An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications*, 1st ed. Springer Publishing Company, Incorporated, 2010.
21. S. Palanivel and B. Yegnanarayana, "Multimodal person authentication using speech, face and visual speech," *Computer Vision and Image Understanding*, vol. 109, no. 1, pp. 44 – 55, 2008.
22. L. Nanni and A. Lumini, "Heterogeneous bag-of-features for object/ scene recognition," *Applied Soft Computing*, vol. 13, no. 4, pp. 2171 – 2178, 2013.
23. L. Nanni, S. Brahmam, and A. Lumini, "Random interest regions for object recognition based on texture descriptors and bag of features," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 973 – 977, Jan. 2012.
24. Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, ser. CVPR'04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 506 – 513.
25. R. Jafri, S. A. Ali, and H. R. Arabnia, "Face recognition for the visually impaired," in The 2013 International Conference on Information and Knowledge (IKE13), Las Vegas, Nevada, USA, 2013, pp. 153 – 159.

Conclusion

We were able to implement and test the basic implementation of the image recognition system, we also included audio feedback alongside the recognition system.

We got satisfactory results upon testing the recognition system with various faces. The recognition was accurate most of the time.

For further detailed and more real-world implementation, a mobile device's camera can be used for image acquisition while an Android application will process the image. The application will feature face detection and recognition program. The face detection program will detect faces from the video feed using a cascade classifier. Once a face is detected, it will be recognized using a temporary image of the detected face. Google Talkback can be used to provide users with feedback on the operation of the application.

The detection system can be further improved by training a new cascade classifier for the face detector while a neural network-based program can be used to improve the recognition system. In addition, the system can be tested with actual users to determine the positive and negative aspects of the system's design. Wearable devices such as smartwatches could also improve the system by making it easier for the user to interact with the mobile application.