

BANGALORE UNIVERSITY

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

Department of Electronics and Communication Engineering

K R Circle, Bangalore – 560001



Project Report

On

**SMART HOME DOOR UNLOCK SYSTEM
USING OPEN CV**

Submitted By

SYED SAIR UR RAHMAN 17GAEE8054

VU RAGHAV KAUSHIK 17GAEE8056

VIGNESH R 17GAEE8061

Under the Guidance of

Dr. K Suresh Babu

Professor

Dept.of ECE

UVCE, Bangalore

2020-2021



BANGALORE

UNIVERSITY

CERTIFICATE

This is to certify that **VU RAGHAV KAUSHIK** Registration Number **17GAEE8056**, **SYED SAIF UR RAHMAN** Registration Number **17GAEE8054** and **VIGNESH R** Registration Number **17GAEE8061** are Bona-fide students of VIII Semester, Electronics and Communication

Engineering UVCE have completed the project work entitled

**“SMART HOME DOOR UNLOCK SYSTEM
USING OPEN CV”**

in fulfillment for the award of degree of Bachelor of Engineering in Electronics and Communication Engineering from Bangalore University, was carried out during the academic year 2020-21 under the guidance of

Dr. K Suresh Babu

Department of ECE at UVCE Bangalore-01.

GUIDE EXAMINER 1

Dr. K SURESH BABU
Dept. of ECE, UVCE,
Bangalore.

CHAIRMAN EXAMINER 2

Dr. K B RAJA
Dept. of ECE, UVCE,
Bangalore.

ACKNOWLEDGEMENT

Our sincere and grateful acknowledgement to the almighty for giving us the opportunity to pursue the bachelor degree in the Electronics and Communication Engineering and thus helping us to shape our career.

We express our gratitude to our guide **Dr.K SURESH BABU**, Professor, Dept. of Electronics and Communication Engineering for his constant encouragement, appreciation and timely advice during the implementation of this project.

We would like to thank **Dr. K B RAJA** , HOD of Electronics and Communication Engineering Dept for providing us invaluable, constructive and skilled guidance, which helped us to successfully complete this project.

We would take this opportunity to thank our principal of UVCE, Dr.HN Ramesh,for providing the necessary support.

We would like to thank our parents and friends for their kind help in bringing out this work within the stipulated time. Also, we would like to thank everyone who had directly or indirectly helped us.

VU RAGHAV KAUSHIK
17GAEE8056

SYED SAIR UR RAHMAN
17GAEE8054

VIGNESH R
17GAEE8061

TABLE OF CONTENTS

Certificate.....	(i)
Acknowledgement.....	(ii)
Table of contents.....	(iii)
Abstract.....	(iv)
List of Figures.....	(v)

Chapters

1. Introduction.....	01
2. Literature Survey.....	21
3. Proposed System.....	22
4. Implementation.....	24
5. Results.....	26
6. Objectives.....	31
7. Performance Analysis.....	32
8. Conclusion.....	33
9. References.....	34

LIST OF FIGURES

<u>1.1</u>	<u>OpenCV</u>	2
<u>1.1.1</u>	<u>API Concepts</u>	4
<u>1.1.2</u>	<u>Automatic Memory Management</u>	5
<u>1.1.3</u>	<u>Automatic Allocation of the Output Data</u>	6
<u>1.1.4</u>	<u>Saturation Arithmetic</u>	7
<u>1.1.5</u>	<u>Fixed Pixel Types. Limited Use of Templates</u>	8
<u>1.1.6</u>	<u>InputArray and OutputArray</u>	9
<u>1.1.7</u>	<u>Error Handling</u>	10
<u>1.1.8</u>	<u>Multi-threading and Re-enterability</u>	10
<u>1.1.9</u>	<u>Applications of Open CV</u>	11
<u>1.2</u>	<u>Numpy</u>	13
<u>1.2.1</u>	<u>The N-dimensional array (ndarray)</u>	13
<u>1.2.2</u>	<u>Constructing arrays</u>	13
<u>1.3</u>	<u>Arrays</u>	15
<u>1.3.1</u>	<u>Basic array operations</u>	17
<u>1.4</u>	<u>Operating System (OS)</u>	17
<u>1.4.1</u>	<u>os.urandom(size)</u>	20
<u>3.1</u>	<u>Facial Recognition</u>	22
<u>3.2</u>	<u>Emailing the Visitor's Information to the Owner</u>	22
<u>3.3</u>	<u>Software Requirements</u>	22

ABSTRACT

The proposed system helps in identification of person standing at the door for a home surveillance system . The database obtained from the captured images is properly stored in the faces folder in the desktop in the computer. We use Open CV for face detection

and identification of the person standing at the door. A prototype camera model can be installed at the door which has a high resolution camera for face detection and identification of stranger at the door. With our code and training model we have achieved a very good accuracy level. Our software gives us the name of the person if he/she is known and with a developed model the door will unlock. If the person is unknown then his/hers details is sent via a email to the admin who can take the further actions.

Chapter 1 INTRODUCTION

Over the last decade smart phones have gained popularity like no other. People enjoy using Smart Phones due to their ease of use. They enjoy doing all their work even without touching the Phone. They interact with the Virtual Assistant[3] like Siri, Cortana for all their work. They rely completely on the virtual assistants for their small to big tasks, like to call someone, to check the emails or to enquire about temperature. But till now the virtual assistants are limited to phones only, also devices like Google Home and Alexa cannot do much tasks around the house. This system revolves around the Virtual Assistant created specifically for homes and named as “OLIVIA” which can be installed anywhere inside the house as it lives on Raspberry Pi which is small computer and does not take much space and power to operate. The idea is to convert any ordinary home into a smart home by just installing devices such as microphones & speakers, PIR sensors, camera, Door/Window Sensors, Wireless Switch around the house which can be used by virtual assistant to control various things around the house [9]. Olivia, The Virtual Assistant may interact with the user solely through user’s voice and could do anything as per the user’s demand. Microphones and speakers are installed throughout the house which are used by Olivia to interact with the user. The commands from user are taken from microphone and the appropriate response is generated and then Olivia speaks the reply using speakers. Olivia may be used to provide all kinds of services in a Smart Home from entertainment to even security purposes. For instance, in a Smart Home Olivia can be connected with tube lights or table fans [4] which user may control solely through his/her voice. When the user may just say “Switch on/off the lights”, the lights may get switched on/off on their own without the user’s physical need to actually get up and on/off the switches. Olivia may be used even for entertainment purposes like the User may ask Olivia the temperature outside the smart home and Olivia will assist the User by telling the exact temperature and will even suggest what kind

of clothes one should wear for such temperature. User may ask endless queries like weather report or stock market prices in real time, calculations, meanings of different words or even a joke and Olivia will properly interact with the User with all the answers to his/her queries. In this system to demonstrate how Olivia, may be integrated into any hardware we have integrated it with the Smart Home Surveillance [6] System using Raspberry PI for the security purpose of the Smart Home. Olivia through Eigenface Algorithm [5], will detect and recognize the person present at the door as either the Owner of the house or as a Stranger.

1.1 OpenCV

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 the commercial products. Being a BSD-licensed product, 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, stitch images together to produce a 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 of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, 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, detection of 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.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.

Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.

Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.

Some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

The further chapters of the document describe functionality of each module. But first, make sure to get familiar with the common API concepts used thoroughly in the library.

1.1.1 API Concepts

cv Namespace

All the OpenCV classes and functions are placed into the `cv` namespace.

Therefore, to access this functionality from your code, use the `cv::` specifier or `using namespace cv;` directive:

```
#include "opencv2/core.hpp"
...
cv::Mat H = cv::findHomography(points1, points2, cv::RANSAC, 5);
...
```

or :

```
#include "opencv2/core.hpp"
using namespace cv;
...
Mat H = findHomography(points1, points2, RANSAC, 5 );
...
```

Some of the current or future OpenCV external names may conflict with STL or other libraries. In this case, use explicit namespace specifiers to resolve the name conflicts:

```
Mat a(100, 100, CV_32F);
randu(a, Scalar::all(1), Scalar::all(std::rand()));
cv::log(a, a);
a /= std::log(2.);
```

1.1.2 Automatic Memory Management

OpenCV handles all the memory automatically.

First of all, `std::vector`, **`cv::Mat`**, and other data structures used by the functions and methods have destructors that deallocate the underlying memory buffers when needed. This means that the destructors do not always deallocate the buffers as in case of `Mat`. They take into account possible data sharing. A destructor decrements the reference counter associated with the matrix data buffer. The buffer is deallocated if and only if the reference counter reaches zero, that is, when no other structures refer to the same buffer. Similarly, when a `Mat` instance

is copied, no actual data is really copied. Instead, the reference counter is incremented to memorize that there is another owner of the same data. There is also the `Mat::clone` method that creates a full copy of the matrix data.

1.1.3 Automatic Allocation of the Output Data

OpenCV deallocates the memory automatically, as well as automatically allocates the memory for output function parameters most of the time. So, if a function has one or more input arrays (`cv::Mat` instances) and some output arrays, the output arrays are automatically allocated or reallocated. The size and type of the output arrays are determined from the size and type of input arrays. If needed, the functions take extra parameters that help to figure out the output array properties.

The array frame is automatically allocated by the `>>` operator since the video frame resolution and the bit-depth is known to the video capturing module. The array edges is automatically allocated by the `cvtColor` function. It has the same size and the bit-depth as the input array. The number of channels is 1 because the color conversion code `cv::COLOR_BGR2GRAY` is passed, which means a color to grayscale conversion. Note that frame and edges are allocated only once during the first execution of the loop body since all the next video frames have the same resolution. If you somehow change the video resolution, the arrays are automatically reallocated.

The key component of this technology is the `Mat::create` method. It takes the desired array size and type. If the array already has the specified size and type, the method does nothing. Otherwise, it releases the previously allocated data, if any (this part involves decrementing the reference counter and comparing it with zero), and then allocates a new buffer of the required size. Most functions call the

Mat::create method for each output array, and so the automatic output data allocation is implemented.

Some notable exceptions from this scheme are **cv::mixChannels**, **cv::RNG::fill**, and a few other functions and methods. They are not able to allocate the output array, so you have to do this in advance.

1.1.4 Saturation Arithmetic

As a computer vision library, OpenCV deals a lot with image pixels that are often encoded in a compact, 8- or 16-bit per channel, form and thus have a limited value range. Furthermore, certain operations on images, like color space conversions, brightness/contrast adjustments, sharpening, complex interpolation (bi-cubic, Lanczos) can produce values out of the available range. If you just store the lowest 8 (16) bits of the result, this results in visual artifacts and may affect a further image analysis. To solve this problem, the so-called *saturation* arithmetics is used. For example, to store r , the result of an operation, to an 8-bit image, you find the nearest value within the 0..255 range:

$$I(x,y)=\min(\max(\text{round}(r),0),255)$$

Similar rules are applied to 8-bit signed, 16-bit signed and unsigned types. This semantics is used everywhere in the library. In C++ code, it is done using the **cv::saturate_cast<>** functions that resemble standard C++ cast operations. See below the implementation of the formula provided above:

$$I.at<\text{uchar}>(y, x) = \text{saturate_cast}<\text{uchar}>(r);$$

where `cv::uchar` is an OpenCV 8-bit unsigned integer type. In the optimized SIMD code, such SSE2 instructions as `paddusb`, `packuswb`, and so on are used. They help achieve exactly the same behavior as in C++ code.

1.1.5 Fixed Pixel Types. Limited Use of Templates

Templates is a great feature of C++ that enables implementation of very powerful, efficient and yet safe data structures and algorithms. However, the extensive use of templates may dramatically increase compilation time and code size. Besides, it is difficult to separate an interface and implementation when templates are used exclusively. This could be fine for basic algorithms but not good for computer vision libraries where a single algorithm may span thousands lines of code. Because of this and also to simplify development of bindings for other languages, like Python, Java, Matlab that do not have templates at all or have limited template capabilities, the current OpenCV implementation is based on polymorphism and runtime dispatching over templates. In those places where runtime dispatching would be too slow (like pixel access operators), impossible (generic `cv::Ptr<>` implementation), or just very inconvenient (`cv::saturate_cast<>()`) the current implementation introduces small template classes, methods, and functions. Anywhere else in the current OpenCV version the use of templates is limited.

Consequently, there is a limited fixed set of primitive data types the library can operate on. That is, array elements should have one of the following types:

- 8-bit unsigned integer (uchar)
- 8-bit signed integer (schar)
- 16-bit unsigned integer (ushort)
- 16-bit signed integer (short)
- 32-bit signed integer (int)
- 32-bit floating-point number (float)
- 64-bit floating-point number (double)

a tuple of several elements where all elements have the same type (one of the above). An array whose elements are such tuples, are called multi-channel arrays,

as opposite to the single-channel arrays, whose elements are scalar values. The maximum possible number of channels is defined by the **CV_CN_MAX** constant, which is currently set to 512.

For these basic types, the following enumeration is applied:

```
enum { CV_8U=0, CV_8S=1, CV_16U=2, CV_16S=3, CV_32S=4, CV_32F=5,
CV_64F=6 };
```

Multi-channel (n-channel) types can be specified using the following options:

- **CV_8UC1** ... **CV_64FC4** constants (for a number of channels from 1 to 4)
- **CV_8UC(n)** ... **CV_64FC(n)** or **CV_MAKETYPE(CV_8U,**
n) ... **CV_MAKETYPE(CV_64F, n)** macros when the number of channels is more than 4 or unknown at the compilation time.

1.1.6 InputArray and OutputArray

Many OpenCV functions process dense 2-dimensional or multi-dimensional numerical arrays. Usually, such functions take `cppMat` as parameters, but in some cases it's more convenient to use `std::vector<>` (for a point set, for example) or **cv::Matx<>** (for 3x3 homography matrix and such). To avoid many duplicates in the API, special "proxy" classes have been introduced. The base "proxy" class is **cv::InputArray**. It is used for passing read-only arrays on a function input. The derived from `InputArray` class **cv::OutputArray** is used to specify an output array for a function. Normally, you should not care of those intermediate types (and you should not declare variables of those types explicitly) - it will all just work automatically. You can assume that instead of `InputArray/OutputArray` you can always use `Mat`, `std::vector<>`, **cv::Matx<>**, **cv::Vec<>** or **cv::Scalar**. When a function has an optional input or output array, and you do not have or do not want one, pass **cv::noArray()**.

1.1.7 Error Handling

OpenCV uses exceptions to signal critical errors. When the input data has a correct format and belongs to the specified value range, but the algorithm cannot succeed for some reason (for example, the optimization algorithm did not converge), it returns a special error code (typically, just a boolean variable).

The exceptions can be instances of the **cv::Exception** class or its derivatives. In its turn, **cv::Exception** is a derivative of `std::exception`. So it can be gracefully handled in the code using other standard C++ library components.

The exception is typically thrown either using the **CV_Error(errcode, description)** macro, or its printf-like **CV_Error**(errcode, (printf-spec, printf-args)) variant, or using the **CV_Assert(condition)** macro that checks the condition and throws an exception when it is not satisfied. For performance-critical code, there is **CV_DbgAssert(condition)** that is only retained in the Debug configuration. Due to the automatic memory management, all the intermediate buffers are automatically deallocated in case of a sudden error. You only need to add a try statement to catch exceptions, if needed.

1.1.8 Multi-threading and Re-enterability

The current OpenCV implementation is fully re-enterable. That is, the same function or the same methods of different class instances can be called from different threads. Also, the same Mat can be used in different threads because the reference-counting operations use the architecture-specific atomic instructions.

1.1.9 Application of Open CV

With the help of Open CV in python, its possible to process images, videos easily and can extract useful information out of that, as there are lots of functions available. Some of the common applications are,

Image Processing:

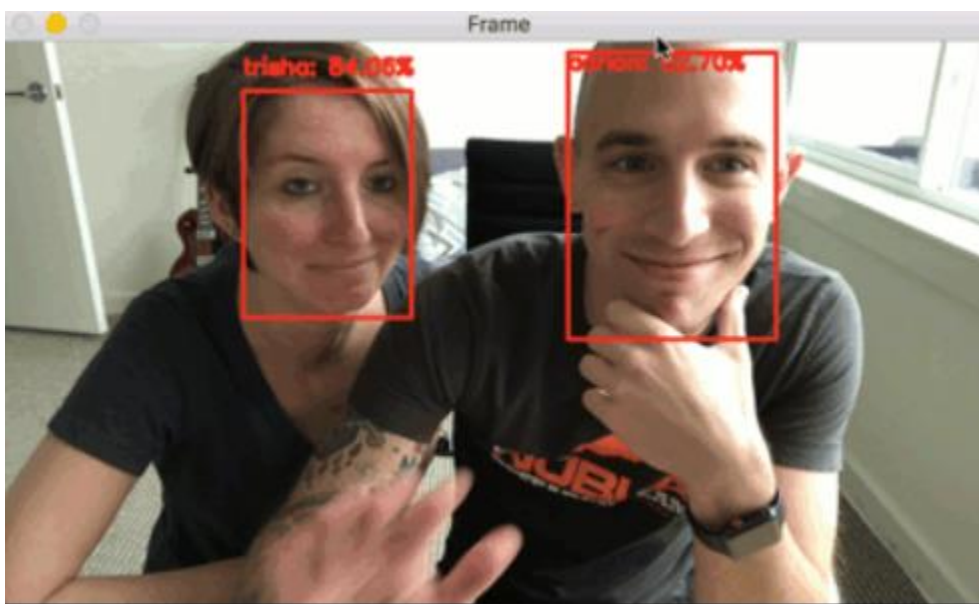
Images can be read, write, show, and processed with the OpenCV, like can generate a new image from that either by changing its shape, colour, or extract something useful from the given one and write into a new image.

Face Detection:

Either from the live streaming using web camera or from the locally stored videos/images utilizing [Haar- Cascade Classifiers](#).

Face Recognition:

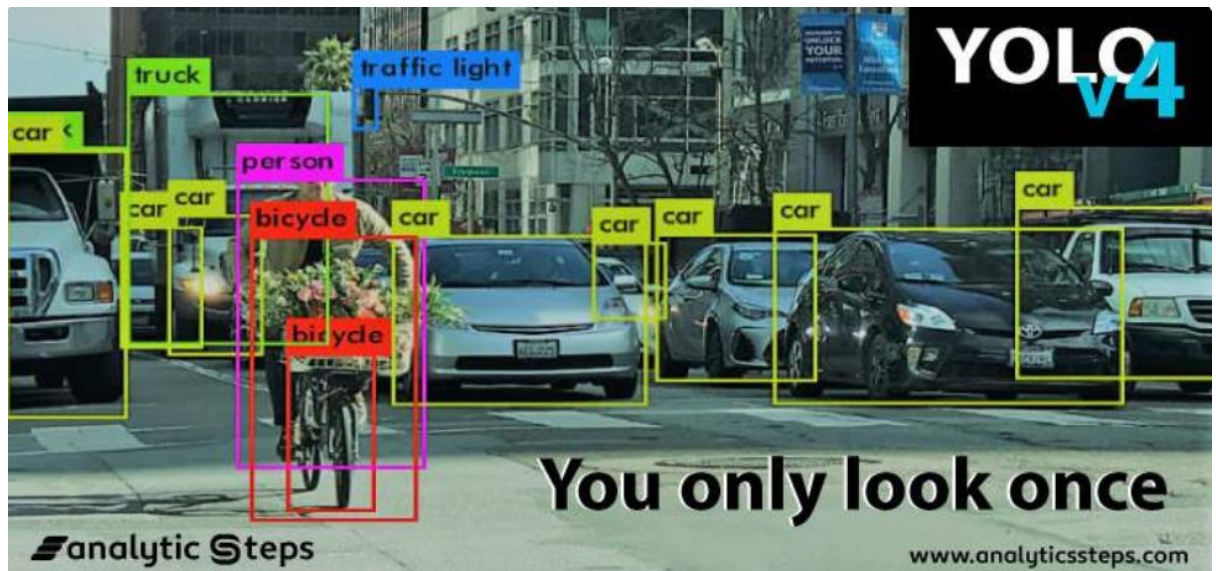
It followed by face detection from the videos using open cv by drawing bounding boxes i.e. rectangle and then model training using [ML algorithms](#) to recognize faces.



Face Recognition using Open CV, [Source](#)

Object Detection:

Open CV along with the [YOLO](#), an object detection algorithm can be used to detect objects from the image, videos either moving or stationary objects.



Object Detection using YOLO and OpenCV

1.2 Numpy

NumPy provides an N-dimensional array type, the `ndarray`, which describes a collection of “items” of the same type. The items can be indexed using for example N integers. All `ndarrays` are homogenous: every item takes up the same size block of memory, and all blocks are interpreted in exactly the same way. How each item in the array is to be interpreted is specified by a separate data-type object, one of which is associated with every array. In addition to basic types (integers, floats, etc.), the data type objects can also represent data structures. An item extracted from an array, e.g., by indexing, is represented by a Python object whose type is one of the array scalar types built in NumPy. The array scalars allow easy manipulation of also more complicated arrangements of data.

1.2.1 The N-dimensional array (`ndarray`)

An `ndarray` is a (usually fixed-size) multidimensional container of items of the same type and size. The number of dimensions and items in an array is defined by its shape, which is a tuple of N non-negative integers that specify the sizes of each dimension. The type of items in the array is specified by a separate data-type object (`dtype`), one of which is associated with each `ndarray`. As with other container objects in Python, the contents of an `ndarray` can be accessed and modified by indexing or slicing the array (using, for example, N integers), and via the methods and attributes of the `ndarray`.

Different `ndarrays` can share the same data, so that changes made in one `ndarray` may be visible in another. That is, an `ndarray` can be a “view” to another `ndarray`, and the data it is referring to is taken care of by the “base” `ndarray`. `ndarrays` can also be views to memory owned by Python strings or objects implementing the buffer or array interfaces.

1.2.2 Constructing arrays

```
class numpy.ndarray(shape, dtype=float, buffer=None, offset=0, strides=None, order=None)
```

An array object represents a multidimensional, homogeneous array of fixed-size items. An associated data-type object describes the format of each element in the array (its byte-order, how many bytes it occupies in memory, whether it is an integer, a floating point number, or something else, etc.) Arrays should be constructed using `array`, `zeros` or `empty` (refer to the See Also section below). The

parameters given here refer to a low-level method (`ndarray(...)`) for instantiating an array.

Parameters

shape

[tuple of ints] Shape of created array.

dtype

[data-type, optional] Any object that can be interpreted as a numpy data type.

buffer

[object exposing buffer interface, optional] Used to fill the array with data. **offset**

[int, optional] Offset of array data in buffer. **strides** [tuple of ints, optional]

Strides

of data in memory.

order

[{'C', 'F'}, optional] Row-major (C-style) or column-major (Fortran-style)

Also,

array

Construct an array.

zeros

Create an array, each element of which is zero.

empty

Create an array, but leave its allocated memory unchanged (i.e., it contains “garbage”).

dtype

Create a data-type.

There are two modes of creating an array using `__new__`:

1. If buffer is None, then only shape, dtype, and order are used.
2. If buffer is an object exposing the buffer interface, then all keywords are interpreted. No `__init__` method is needed because the array is fully initialized after the `__new__` method.

1.3 Arrays

The central feature of NumPy is the array object class. Arrays are similar to lists in Python, except that every element of an array must be of the same type, typically a numeric type like float or int. Arrays make operations with large amounts of numeric data very fast and are generally much more efficient than lists.

Array slicing works with multiple dimensions in the same way as usual, applying each slice specification as a filter to a specified dimension. Use of a single ":" in a dimension indicates the use of everything along that dimension.

Arrays can be reshaped using tuples that specify new dimensions. In the following example, we turn a ten-element one-dimensional array into a two-dimensional one whose first axis has five elements and whose second axis has two elements.

One can convert the raw data in an array to a binary string (i.e., not in human-readable form) using the `tostring` function. The `fromstring` function then allows an array to be created from this data later on. These routines are sometimes convenient for saving large amount of array data in files that can be read later on.

If an array has more than one dimension, it is possible to specify the axis along which multiple arrays are concatenated. By default (without specifying the axis), NumPy concatenates along the first dimension:

```
>>> a = np.array([[1, 2], [3, 4]], float)
>>> b = np.array([[5, 6], [7,8]], float)
>>> np.concatenate((a,b)) array([[ 1., 2.], [ 3., 4.], [ 5., 6.], [ 7., 8.]])
>>> np.concatenate((a,b), axis=0) array([[ 1., 2.], [ 3., 4.], [ 5., 6.], [ 7., 8.]])
>>> np.concatenate((a,b), axis=1) array([[ 1., 2., 5., 6.], [ 3., 4., 7., 8.]])
```

However, arrays that do not match in the number of dimensions will be broadcasted by Python to perform mathematical operations. This often means that the smaller array will be repeated as necessary to perform the operation indicated. Python automatically broadcasts arrays in this manner. Sometimes, however, how we should broadcast is ambiguous. In these cases, we can use the `newaxis` constant to specify how we want to broadcast.

In addition to the standard operators, NumPy offers a large library of common mathematical functions that can be applied elementwise to arrays. Among these are the functions: `abs`, `sign`, `sqrt`, `log`, `log10`, `exp`, `sin`, `cos`, `tan`, `arcsin`, `arccos`, `arctan`, `sinh`, `cosh`, `tanh`, `arcsinh`, `arccosh`, and `arctanh`.

1.3.1 Basic array operations

Many functions exist for extracting whole-array properties. The items in an array can be summed or multiplied.

For most of the routines described below, both standalone and member functions are available. A number of routines enable computation of statistical quantities in array datasets, such as the mean (average), variance, and standard deviation

1.4 Operating System (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 *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.fork`, `os.execv` and `os.spawn*p*` are not supported.

File Names, Command Line Arguments, and Environment Variables:

In Python, file names, command line arguments, and environment variables are represented using the string type. On some systems, decoding these strings to and from bytes is necessary before passing them to the operating system. Python uses the file system encoding to perform this conversion (see `sys.getfilesystemencoding()`).

Changed in version 3.1: On some systems, conversion using the file system encoding may fail. In this case, Python uses the surrogateescape encoding error handler, which means that undecodable bytes are replaced by a Unicode character U+DCxx on decoding, and these are again translated to the original byte on encoding.

The file system encoding must guarantee to successfully decode all bytes below 128. If the file system encoding fails to provide this guarantee, API functions may raise `UnicodeErrors`.

It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

1. Creating Directory

We can create a new directory using the **mkdir()** function from the OS module. A new directory corresponding to the path in the string argument of the function will be created. If we open D drive in Windows Explorer, we should notice **tempdir** folder created.

2. Changing the Current Working Directory

We must first change the current working directory to a newly created one before doing any operations in it. This is done using the **chdir()** function. There is a **getcwd()** function in the OS module using which we can confirm if the current working directory has been changed or not.

3. Removing a Directory

The **rmdir()** function in the OS module removes the specified directory either with an absolute or relative path. However, we can not remove the current working directory. Also, for a directory to be removed, it should be empty. For example, tempdir will not be removed if it is the current directory. We have to change the current working directory and then remove tempdir.

4. List Files and Sub-directories

The **listdir()** function returns the list of all files and directories in the

specified directory. If we don't specify any directory, then list of files and directories in the current working directory will be returned.

1.4.1 os.urandom(*size*)

Return a string of *size* random bytes suitable for cryptographic use.

This function returns random bytes from an OS-specific randomness source. The returned data should be unpredictable enough for cryptographic applications, though its exact quality depends on the OS implementation.

On Linux, if the `getrandom()` syscall is available, it is used in blocking mode: block until the system urandom entropy pool is initialized (128 bits of entropy are collected by the kernel). See the **PEP 524** for the rationale. On Linux, the `getrandom()` function can be used to get random bytes in non-blocking mode (using the `GRND_NONBLOCK` flag) or to poll until the system urandom entropy pool is initialized.

On a Unix-like system, random bytes are read from the `/dev/urandom` device. If the `/dev/urandom` device is not available or not readable, the `NotImplementedError` exception is raised.

On Windows, it will use `CryptGenRandom()`.

Chapter 2 Literature Survey

The most available home automation systems in the literature uses different wireless communication standards like IR, RF, Bluetooth, ZigBee, Wi-Fi, and Global System for Mobile Communication (GSM) to exchange data and signaling between their components. Wireless home automation systems decrease installation cost and effort, and enhance system flexibility and scalability. [4] A few examples of home automation system in literature are GSM based Home Automation using SMS [2,3,6,7], IR Remote based Home Automation System [10], Bluetooth Based Home Automation System [8], RF and Zigbee based Home Automation System [12], Wi-Fi based Home Automation System, Java-(Web) Based Home Automation System. [11] Even if many varieties of home automation systems are available, current systems have number of limitations. Currently home automation systems are implemented with a large amount of hardware. The installation and maintenance of the systems is a difficult task. It also imposes a huge installation cost on the user or consumer. Current home automation systems are inefficient in security. In Infrared (IR) remote based Home automation requires line of sight communication. [10] Radio Frequency (RF) based home automation required high power consumption. They are also very poor in bandwidth utilization. In case of ZigBee, the bandwidth is too low and in case of GSM, it is too high. [12] The java web based home automation is very poor in security as it uses web pages to access and control the appliances. [11] Bluetooth have limited communication range. [8] MS based and GSM based home automation is costly for the consumer as it becomes expensive to communicate via SMS. [2, 3] To overcome all this limitation system based on Raspberry Pi and Android operating system has been proposed. Using concept of android-based home automation systems, users can be provided with simple, secure and easily configurable home automation system. Raspberry Pi is a credit card sized computer. It's basically a small PC which provides all the basic functions that are provided by a desktop PC. For example, it provides functions like word processing, gaming and playing audio/video. [9]

Chapter 3 Proposed Model

3.1 Facial Recognition

A web camera is attached at the main door of the house. Whenever someone rings the doorbell, Olivia gets activated, clicks a picture of the person and identifies the person either as the Owner of the house or as Stranger/Visitor using Eigenface [5] Algorithm. If the person is recognized as the Owner then Olivia unlocks the door and assists the Owner inside the house with various queries like temperature outside, stock market price etc. Else if the Owner isn't present inside the house and Olivia recognizes the person at door as a Stranger/Visitor then Olivia doesn't unlock the door, but clicks a picture of the Stranger, takes his/her name and message (if any) and informs the Owner simultaneously via Email/SMS [10]. The main features of the system include: 1. Capturing photos and recognizing faces in real time. 2. Olivia, the Virtual Assistant interacting with the User with human like Voice to give a sense of belongingness. 3. If a person is recognized as the Owner, then the door gets unlocked automatically. 4. In the home, Olivia may assist the Owner by performing Arithmetic calculations, searching Wikipedia, checking emails, weather or stock price based on voice commands provided by the owner and the generated computed response via voice. 5. If a person is recognized as a stranger in absence of the Owner in the home, then Olivia will interact with him/her and ask for name and message, if any and will inform the Owner with the picture, name and message of the Stranger via Email/SMS. 6. Olivia can control various devices around the house on command of the user. 7. Olivia can set on/off timer for devices around the house [9]. 8. Olivia can detect the presence of human inside a room. 9. Entertaining the Owner/User like playing music or telling jokes as per voice command.

3.2 Emailing the Visitor's Information to the Owner

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides smtplib module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

3.3 Software Requirements

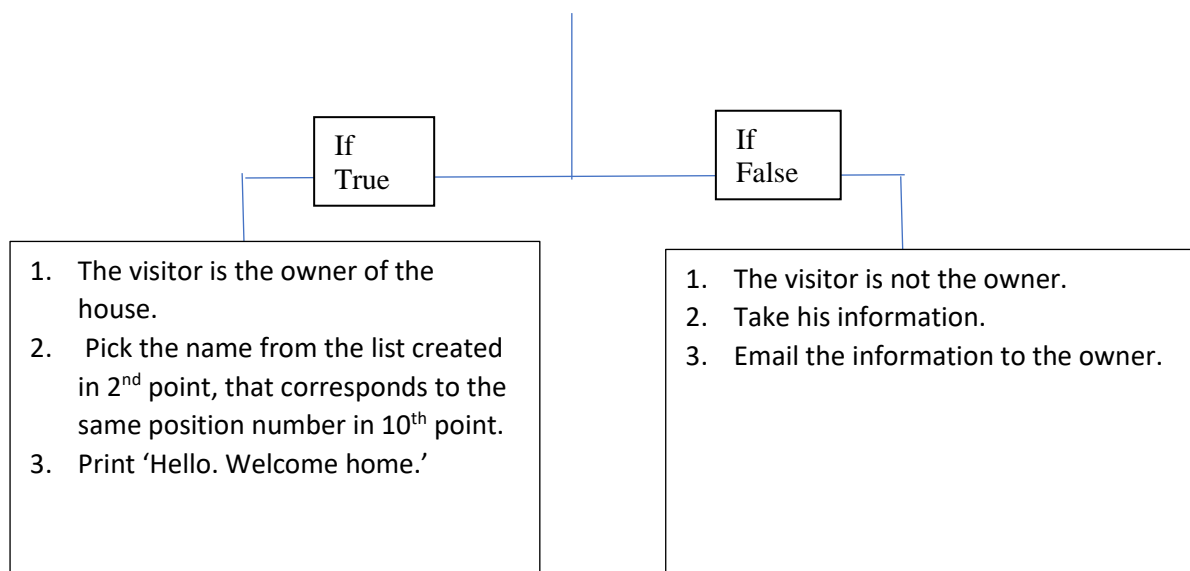
Various software libraries and applications were used in developing this system. Major software resources used include:

1. EIGENFACE [5] ALGORITHM FOR FACE DETECTION— It is a facial recognition algorithm that uses eigen vector to detect and classify faces. This approach of using eigenfaces to recognize and classify faces was developed by Sirovich and Kirby.
2. In this algorithm faces are transformed into a small set of crucial or key characteristics known as eigenfaces, which form the main component of the initial training set images. Facial Recognition is done by casting a new image in the subspace of eigenface, after which the comparison in the position of eigenface space and position of known person is done and the person may be classified.
3. The main advantage of using this algorithm over the other face recognition algorithms is its inconsideration towards small or gradual changes on the face, Speed and Simplicity.
4. OPENCV—This library gives a infrastructure for computer vision applications. It offers algorithms to detect and recognize classify human actions, face, identify objects, track object movement and much more. OpenCV works with various computing languages such as C++, Java, Python, MATLAB interfaces and supports Windows, Linux, Mac OS and Android also.
5. We used it for face recognition and face detection (EigenFaces [5]) in our system. It is easy to implement in python.
6. PYTHON 2[2] — Due to the simplicity and availability of wide range of libraries we chose python as our primary language. Removal of errors is much easier with python interpreter thus making the process of development easy.

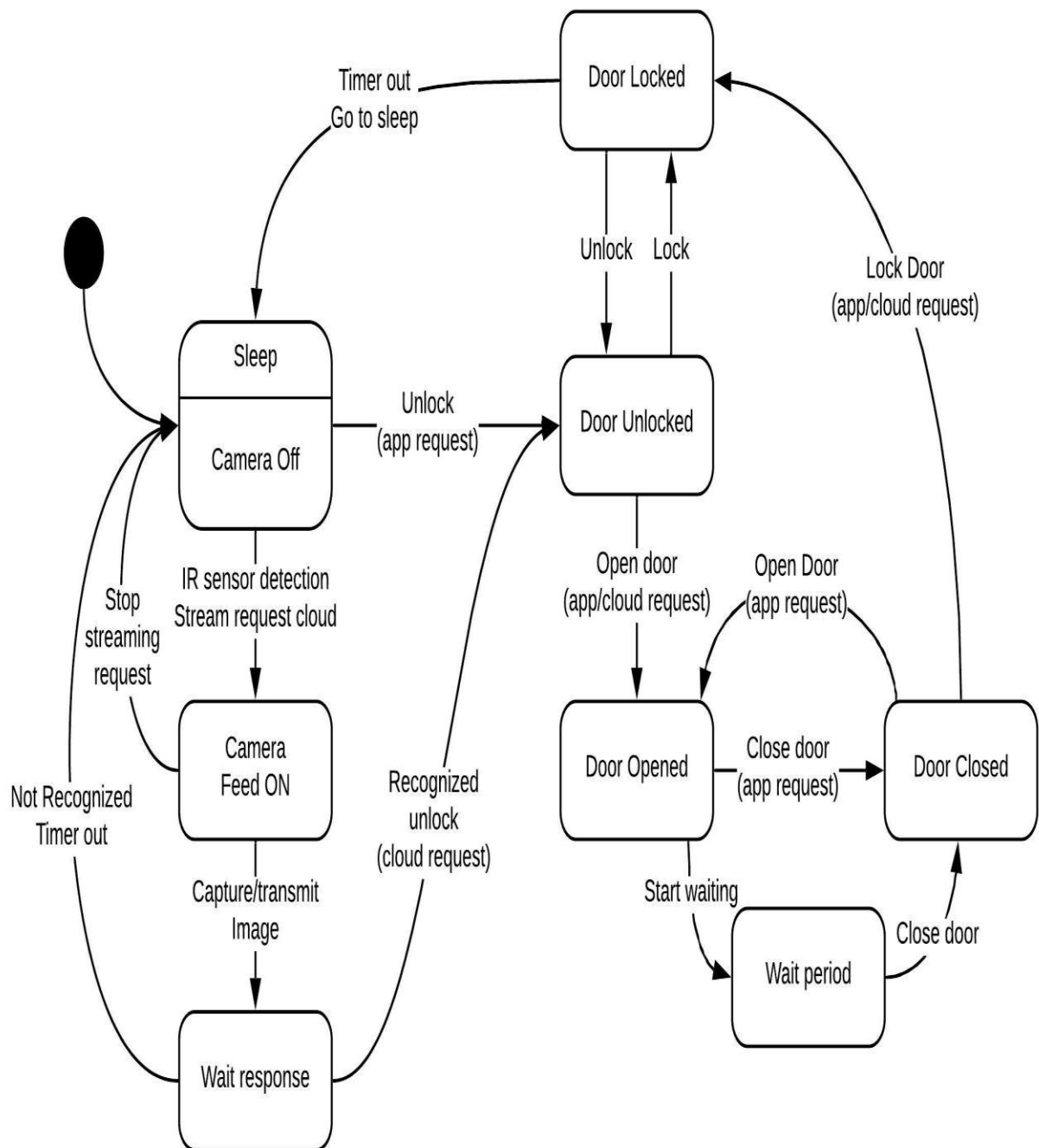
Chapter 4 IMPLEMENTATION

Proposed Algorithm

1. Create a database of owner/owners of the house.
2. Create a list of names of the owners in the same order the pictures are stored in the database.
3. Extract encodings of every image in the database, and store them in a list in the same order the corresponding images are stored in the database.
4. Turn on the camera.
5. Detect the face of the visitor in the camera.
6. Extract the encoding of the visitor's face.
7. Compare the visitor's encoding with all the encodings of the owners, stored in the list earlier. (Returns True if the encoding matches with either of the saved encoding. Returns false otherwise). Store them in a list.
8. Also find the Euclidian distance between the visitor's encoding and all the encodings of the owners, stored in the list earlier. (Returns values which are Euclidian distances).
9. Store the Euclidian distances in a list. Find the position of the least Euclidian distance in the list.
10. Check if the value (in the list got in the 7th point) corresponding to the position got from the above 9th point, is true or false.



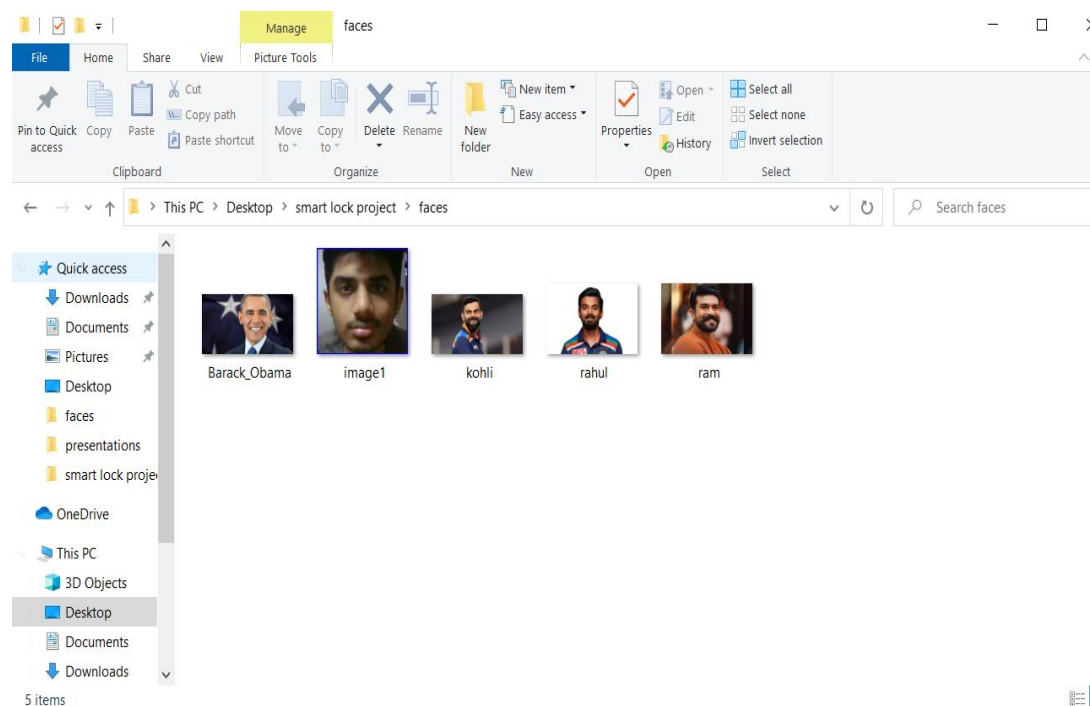
FLOWCHART



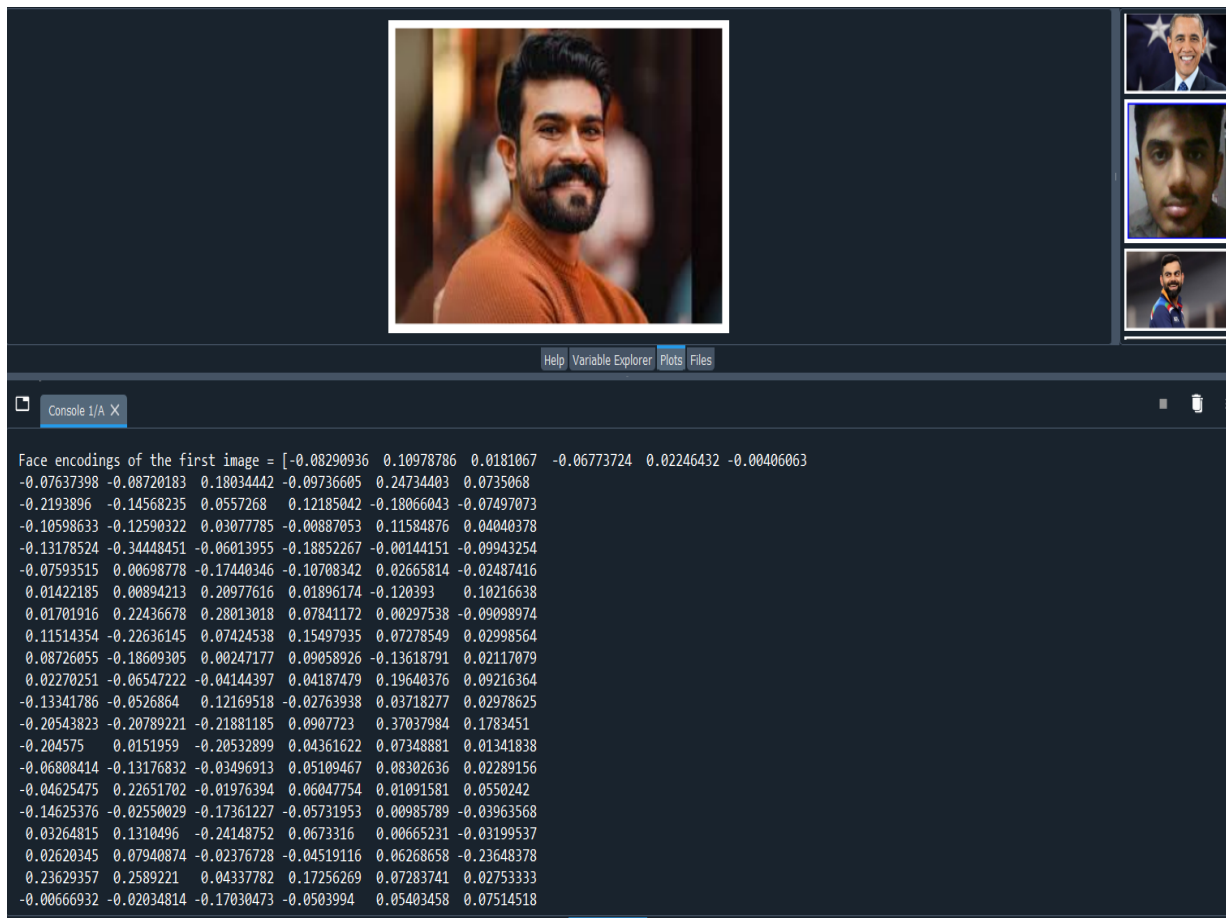
Chapter 5 RESULTS

The below pictures show how the output of the program is from the beginning of taking the input. The tasks which happen are shown in the below steps.

Images stored in database in folder faces:



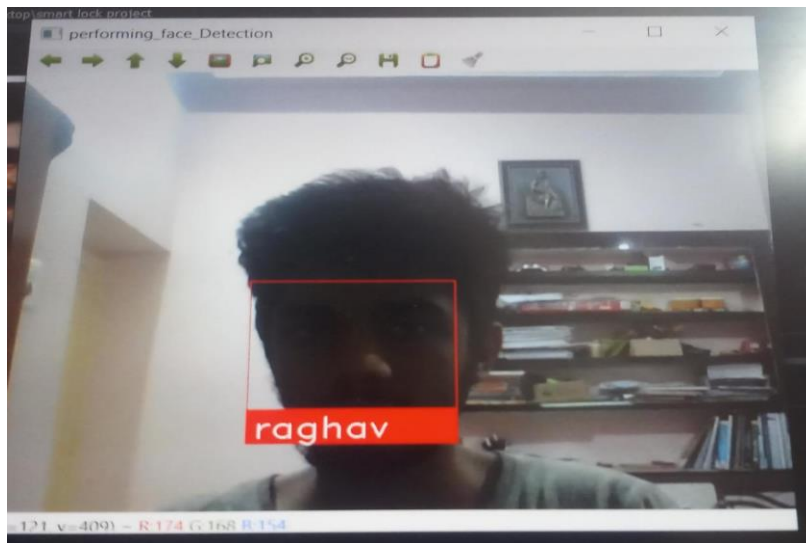
Encoding of images in console:



Face encodings of the first image = [-0.08290936 0.10978786 0.0181067 -0.06773724 0.02246432 -0.00406063

-0.07637398	-0.08720183	0.18034442	-0.09736605	0.24734403	0.07350668
-0.2193896	-0.14568235	0.0557268	0.12185042	-0.18066043	-0.07497073
-0.10598633	-0.12590322	0.03077785	-0.00887053	0.11584876	0.04040378
-0.13178524	-0.34448451	-0.06013955	-0.18852267	-0.00144151	-0.09943254
-0.07593515	0.00698778	-0.17440346	-0.10708342	0.02665814	-0.02487416
0.01422185	0.00894213	0.20977616	0.01896174	-0.120393	0.10216638
0.01701916	0.22436678	0.28013018	0.07841172	0.00297538	-0.09098974
0.11514354	-0.22636145	0.07424538	0.15497935	0.07278549	0.02998564
0.08726055	-0.18609305	0.00247177	0.09058926	-0.13618791	0.02117079
0.02270251	-0.06547222	-0.04144397	0.04187479	0.19640376	0.09216364
-0.13341786	-0.0526864	0.12169518	-0.02763938	0.03718277	0.02978625
-0.20543823	-0.20789221	-0.21881185	0.0907723	0.37037984	0.1783451
-0.204575	0.0151959	-0.20532899	0.04361622	0.07348881	0.01341838
-0.06808414	-0.13176832	-0.03496913	0.05109467	0.08302636	0.02289156
-0.04625475	0.22651702	-0.01976394	0.06047754	0.01091581	0.0550242
-0.14625376	-0.02550029	-0.17361227	-0.05731953	0.00985789	-0.03963568
0.03264815	0.1310496	-0.24148752	0.0673316	0.00665231	-0.03199537
0.02620345	0.07940874	-0.02376728	-0.04519116	0.06268658	-0.23648378
0.23629357	0.2589221	0.04337782	0.17256269	0.07283741	0.02753333
-0.00666932	-0.02034814	-0.17030473	-0.0503994	0.05403458	0.07514518

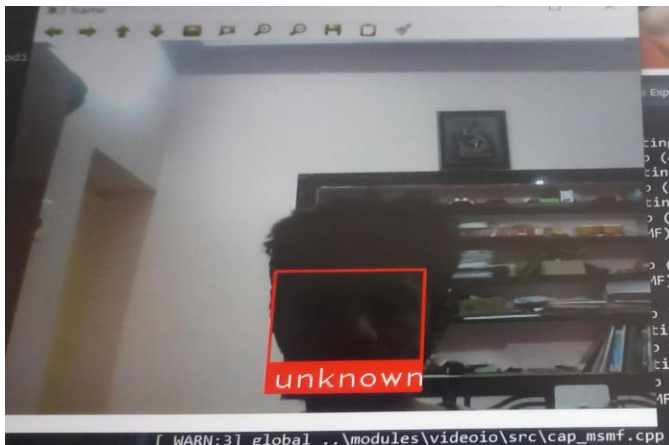
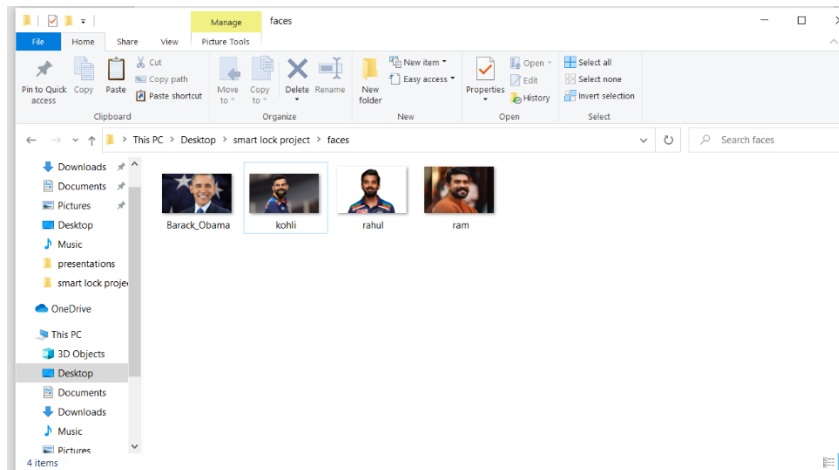
Known Face detection:



Unlocking of the door message:

```
Help Variable Explorer Plots F
Console 1/A X
1.23738363e-01, -1.50310516e-01, -2.01995336e-02, 5.60201369e-02,
-8.26061741e-02, -1.63321078e-01, -3.30574542e-01, 1.23433851e-01,
3.97384226e-01, 1.03702776e-01, -1.75617069e-01, 4.79167476e-02,
-1.64299741e-01, -1.06176473e-01, 7.35460892e-02, 1.15173729e-03,
-6.74451143e-02, 3.76886837e-02, -9.56813246e-02, 6.70061111e-02,
1.72060922e-01, 6.09361231e-02, -2.14769617e-02, 2.08052427e-01,
-5.95448911e-02, -4.17200960e-02, 1.18824942e-02, 3.42960260e-03,
-1.00873053e-01, 2.72486359e-02, -8.87462869e-02, 1.69301517e-02,
4.07163352e-02, -2.30818000e-02, -1.85815506e-02, 5.99173605e-02,
-1.69522077e-01, 1.05742320e-01, 7.67558292e-02, -5.39103858e-02,
-3.22042778e-03, 6.04315624e-02, -1.40287817e-01, -9.14570689e-02,
1.52560413e-01, -2.64202565e-01, 1.35477692e-01, 1.32871717e-01,
2.03345157e-02, 1.68758273e-01, 5.54443412e-02, 5.63696623e-02,
-3.57069410e-02, 8.76915175e-03, -7.35535473e-02, -3.21829915e-02,
2.78046839e-02, -2.89496444e-02, 3.94306481e-02, 1.08971484e-02]])
Matches = [False, True, False, True, False]
Euclidian Distances are
[0.68145388 0.32880903 0.6179517 0.51384732 0.72687631]
Hello, raghav
Welcome home.
Unlocking the door.
```

Removal of image in database for unknown Face detection:



Entering details to send email:

```
-1072873821
[ WARN:2] global ..\modules\videoio\src\cap_msmf.cpp (388) `anonymous-
namespace':::SourceReaderCB::OnReadSample videoio(MSMF): async ReadSample() call is failed with error
status: -1072873821
[ WARN:0] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback
[ WARN:0] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback
[ WARN:3] global ..\modules\videoio\src\cap_msmf.cpp (376) `anonymous-
namespace':::SourceReaderCB::OnReadSample videoio(MSMF): OnReadSample() is called with error status:
-1072873821
[ WARN:3] global ..\modules\videoio\src\cap_msmf.cpp (388) `anonymous-
namespace':::SourceReaderCB::OnReadSample videoio(MSMF): async ReadSample() call is failed with error
status: -1072873821
[ WARN:0] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback

Please enter your name: raghav

Please enter your contact number: axy
Thank you for providing the information. Goodbye!
I

Python console History
Line 93, Col 62 ASCII LF RW
```

Email Sent With Unknown Person Entered Details And Video Recorded:

```
Console 1/1 X
[ WARN:3] global ..\modules\videoio\src\cap_msmf.cpp (376) `anonymous-
namespace':::SourceReaderCB::OnReadSample videoio(MSMF): OnReadSample() is called with error status:
-1072873821
[ WARN:3] global ..\modules\videoio\src\cap_msmf.cpp (388) `anonymous-
namespace':::SourceReaderCB::OnReadSample videoio(MSMF): async ReadSample() call is failed with error
status: -1072873821
[ WARN:0] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback

Please enter your name: raghav

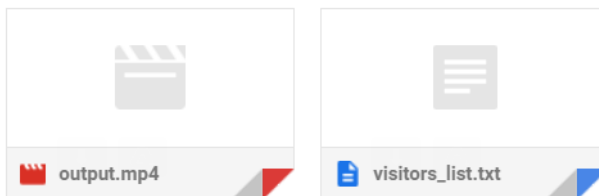
Please enter your contact number: axy
Thank you for providing the information. Goodbye!
The email is sent!

[ WARN:1] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback
[ WARN:0] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback
[ WARN:0] global ..\modules\videoio\src\cap_msmf.cpp (438) `anonymous-
namespace':::SourceReaderCB::~SourceReaderCB terminating async callback
[ WARN:2] global ..\modules\videoio\src\cap_msmf.cpp (376) `anonymous-
```

Email Notification:

Here are a video of the visitor recorded, and the contact information.

2 Attachments



Reply Forward

Chapter 6 OBJECTIVES

The objective of the project is to have an affordable and efficient product of Smart-home that detects whether the visitor is the owner or an outsider, unlocks the door if it's the owner, but only interacts with the outsider by asking his information, and emailing the same to the owner.

Chapter 7 PERFORMANCE ANALYSIS

The project does the job with 100% accuracy and no delay in turning on the camera, capturing the face of the person, concluding whether or not the person is the owner, and emails the visitor's information to the owner immediately.

Chapter 8 CONCLUSION

The virtual assistant with home surveillance system was developed after thoroughly understanding the Python Language- its modules and libraries, the working of existing Virtual Assistants and their shortcomings, the working of the existing face detection and recognition algorithms. The best suited of these algorithms, modules and libraries were selected and implemented in the system. All the resources were chosen keeping in mind the financial and resource constraints.

Chapter 9 References

- [1] R. Vinay Chand¹, M.Veda Chary², "Wireless Home Automation System with Acoustic Controlling", International Journal of Engineering Trends and Technology (IJETT) – Volume 4 Issue 9- Sep 2013
- [2] B. Yukesekkaya, A. A. Kayalar, M. B. Tosun, M. K. Ozcan, and A. Z. Alkar, "A GSM, Internet and Speech Controlled Wireless Interactive Home Automation System," IEEE Transactions on Consumer Electronics, vol. 52, pp. 837-843, August 2006.
- [3] R. Teymourzadeh, S. A. Ahmed, Kok W. Chan and M. Hoong, "Smart GSM Based Home Automation System", 2013, IEEE Conference on Systems, Process & Control, Kuala Lumpur, Malaysia.
- [4] A. Alheraish, "Design and Implementation of Home Automation System", 2004, IEEE Transactions on Consumer Electronics, Vol. 50(4), pp. 1087- 1092.
- [5] V. Bhaumik, P. Niteen, M. Vardhman, "Home Automation and Monitoring System Using Raspberry Pi and Android" 2015, International Journal of Engineering Development and Research, Volume 3, Issue 4, ISSN: 2321-9939, pp.885-892.
- [6] M..N..Jivani, "GSM Based Home Automation System Using App-Inventor for Android Mobile Phone", 2014, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3(9), pp. 12121-12128.
- [7] Y. Baris, A. Alper Kayalar, M. Bilgehan Tosun, M. Kaan Ozcan, and Ali Ziya Alkar. "A GSM, internet and speech controlled wireless interactive home automation system." IEEE Transactions on Consumer Electronics 52, no. 3 (2006): 837-843.
- [8] R. Piyare, M. Tazil, "Bluetooth Based Home Automation System Using Cell Phone", 2011, IEEE 15th International Symposium on Consumer Electronics, Singapore, pp. 192 - 195
- [9] About Raspberry Pi: www.raspberrypi.org
- [10] T. Starner, J. Auxier, D. Ashbrook and M. Gandy, "The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring", In Wearable computers, the fourth international symposium on (pp. 87-94). IEEE, October, 2000.
- [11] A. R. Al-Ali, M. Al-Rousan, "Java-based home automation system", IEEE Transactions on Consumer Electronics, 50(2), 498-504, 2004
- [12] F. Baig, S. Beg and M. Khan, "ZigBee Based Home Appliances Controlling Through Spoken Commands Using Handheld Devices", 2013, International Journal of Smart Home, Vol. 7(1), pp 19 -26.
- [13] A. R. Delgado, R. Picking, V. Grout, "Remote-Controlled Home Automation Systems with Different Network Technologies", Centre for Applied Internet Research, University of Wales, UK.

- [14] P. Kalaivani and D. S. Vimala. "Human Action Recognition Using Background Subtraction Method." International Research Journal of Engineering and Technology (IRJET) 2.3 (2015): 1032-1035.
- [15] P.I. Wilson, J. Fernandez, " Facial feature detection using Haar classifiers", Journal of Computing Sciences in Colleges, 21(4), 127-133, 2006
- [16] J. Yang, D. Zhang, A.F. Frangi & J.Y. Yang, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition", IEEE transactions on pattern analysis and machine intelligence, 26(1), 131- 137, 2004
- [17] Yale database for face Recognition, "http://vision.ucsd.edu/datasets/yale_face_dataset_original/yalefaces.zip"