**VIDEO BASED DYNAMIC HUMAN AUTHENTICATION SYSTEM FOR ACCESS CONTROL**

A MINI PROJECT REPORT

submitted

*in the partial fulfillment of the requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

by

**PITTALA VENKAT RAMANA (17B81A05U8)**

**NUNNA VENKAT VARSHITH (17B81A05U6)**

**BADRI VARUN RAJ (17B81A05U4)**

Under the guidance of

R. SRISHYLAM

ASSISTANT PROFESSOR

A picture containing food

Description automatically generated

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CVR COLLEGE OF ENGINEERING**

**(*An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)**

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),  
Rangareddy (D), Telangana- 501 510

June 2020

**CERTIFICATE**

This is to certify that the project entitled “VIDEO BASED DYNAMIC HUMAN AUTHENTICATION SYSTEM FOR ACCESS CONTROL” that is being submitted by Pittala Venkat Ramana(17B81A05U8), Nunna Venkat Varshith(17B81A05U6), Badri Varun Raj(17B81A05U4) in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the CVR College of Engineering, is a record of bonafide work carried out by them under my guidance and supervision during the year 2019-2020.

The results embodied in this project work has not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the project guide Signature of the HOD

Mr. R Srishylam DR. K VENKATESHWARA RAO

Assistant Professor

Dept. of CSE

CVR College of Engineering

External Examiner

**ACKNOWLEDGEMENT**

The satisfaction and happiness that accompanies the completion of any project would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance have been a source of inspiration throughout the course of the project work. We express our sincere gratitude to all of them.

We thank Dr. K Venkateshwara Rao, Professor and Head of the Department of Computer Science and Engineering, CVR College of Engineering for his constant support and guidance throughout this project.

We thank our guide Mr. R. Srishylam, Assistant Professor, Computer Science and Engineering, CVR College of Engineering for his constant support and guidance throughout the project.

Pittala Venkat Ramana

Nunna Venkat Varshith

Badri Varun Raj

**ABSTRACT**

Technology grows rapidly and keeps on growing exponentially with time. As we are introduced to new technologies we are also faced with a number of problems related to those technologies. Some problems are easy to identify but take a lot of effort to be rectified. One such problem is related to security and surveillance, real time video based monitoring. There have been many attempts to identify people accessing a particular service and decide to whether grant them access or deny but are not so efficient under some conditions such as proper lighting, shady pictures etc., Our project is to overcome the above problem and create a dynamic console application which validates who can enter the premises. When a person who is authorized to enters the premises, he will be given access. If an unauthorized person enters the premises, the app will alert the user, if the user verifies him as authorized, the app allows the person to enter.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table of Contents** | | | |
|  | | | Page No. |
|  |  | List of Figures | 6 |
|  |  |  |  |
| 1 |  | **Introduction** | 7 |
|  | 1.1 | Motivation |  |
|  | 1.2 | Problem statement |  |
|  | 1.3 | Project Objectives |  |
|  |  |  |  |
| 2 |  | **Video Based Dynamic Human Authentication System for Access Control** |  |
|  | 2.1 | Introduction to the characteristics of the Problem |  |
|  | 2.2 | Design challenges |  |
|  | 2.3 | Proposed Solution: Dynamic Face Recognition System |  |
|  |  |  |  |
| 3 |  | **Requirements and Specifications** |  |
|  | 3.1 | Software Requirements |  |
|  | 3.1.1 | Functional Requirements |  |
|  | 3.1.2 | Non-Functional Requirements |  |
|  | 3.2 | System Specifications |  |
|  | 3.2.1 | Software Specs |  |
|  | 3.2.2 | Hardware Specs |  |
|  |  |  |  |
| 4 |  | **Analysis and Design** |  |
|  | 4.1 | Use case Diagram |  |
|  | 4.2 | Class Diagrams |  |
|  | 4.3 | Activity Diagrams |  |
|  | 4.4 | Sequence Diagrams |  |
|  | 4.5 | System Architecture |  |
|  | 4.6 | Technology Description |  |
|  |  |  |  |
| 5 |  | **Implementation & Testing** |  |
|  | 5.1 | Front page Screenshot |  |
|  | 5.2 | --- |  |
|  | 5.3 | --- |  |
|  |  |  |  |
| 6 |  | **Conclusion & Future scope** |  |
|  |  | **References:** |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Figure** | **Page No.** |
| 1. |  |  |
| 2. |  |  |
| 3. |  |  |

1. **INTRODUCTION**
   1. **Motivation**

Over the past few years, face recognition has become one of the most successful applications in computer vision and pattern recognition. These systems are the most promising for personal identification. Examples of modes of biometric systems include face recognition, fingerprints, iris scanning, and others.

Security is one of the most important features to look at as the technology grows. A normal smartphone is more powerful than the processing power used to send an astronaut to moon. The data circulation in today’s life is enormous. Hence Data Security is of utmost importance.

There are many different security methods that can be used to secure data such as PIN, Password, Pattern and Biometric methods such as Fingerprint sensors, Iris scanners etc., These methods do not provide the best security as someone can steal your PIN, Password , Pattern by many different methods such as Social Engineering, Shoulder surfing etc.,

Our project is to overcome the above problem and create a dynamic console application which validates who can enter the premises. When a person who is authorized to enters the premises, he will be given access. If an unauthorized person enters the premises, the app will alert the user, if the user verifies him as authorized, the app allows the person to enter.

* 1. **Problem Statement**

Face recognition systems include a good level of complexity. Some of the problems with the existing face recognition systems are:

* Image acquisition
* Age of the person
* Illumination
* Pose variations
  1. **Project Objectives**
* Develop a Desktop Application for Face Recognition System for Access Control.
* The Application should be user-friendly.
* Registration of users does not take a lot of time.
* Deleting images after training the model reduces space complexity by a huge margin.
* The System should work good in medium to good lighting conditions.

**2. VIDEO BASED DYNAMIC HUMAN AUTHENTICATION SYSTEM FOR ACCESS CONTROL**

**2.1 Introduction to the characteristics of the Problem**

Developing a face recognition project includes a fair amount of difficulty. There are different problems to tackle at each individual step. They are

* acquiring the image of a person from the program
* proper lighting conditions
* Distance of person from camera

Acquiring image of a person is the foremost difficulty in this project. This project works well if there is a clear background behind the user at the time of image acquisition. If there is a disturbance in the background the face detector fails to identify the face of the user and captures other objects which are like a human face.

Lighting conditions greatly influences face detection and recognition. Images in different lighting conditions tend to look like different people for a computer. To avoid such a condition, we use an artificial light to enhance precision.

A group of people posing for the camera

Description automatically generated

Fig 2.1.1: faces in different lighting conditions

We can overcome this problem by using an LBPH face recognizer.

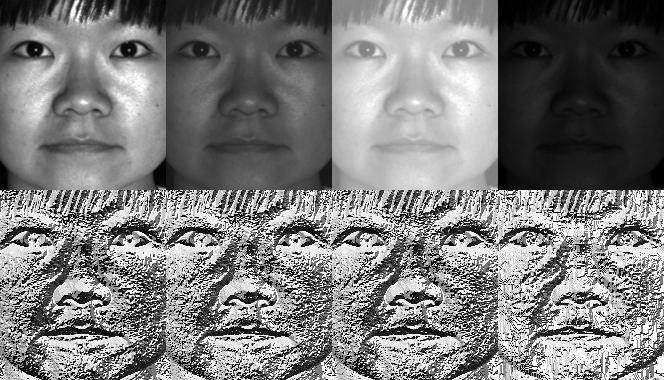


Fig 2.1.2 Images recognized by LBPH

Distance of a person from the camera is important to detect faces. While detecting faces we move a box on the entire frame captured and check whether the box contains the image or not using the cascade classifier haarcascade\_frontalface\_default.xml. The size of the box is important as person nearer to the camera has a larger face than a person who is far from the camera. Giving a proper minimum size to the box helps in detecting faces. The size (30,24) is used in this project to detect faces of person who is at 1 -1.2 m from the camera.

**2.2 Design challenges**

Selection of a suitable face recognizer:

OpenCV provides three types of face recognizers

* Eigenfaces
* Fisherfaces
* Local Binary Patterns Histograms (LBPH)

Eigenfaces and Fisherfaces find a mathematical description of the most dominant features of the training set as a whole.

In [LBPH](http://en.wikipedia.org/wiki/Local_binary_patterns) each images is analysed independently, while the eigenfaces method looks at the dataset as a whole. The LBPH method is somewhat simpler, in the sense that we characterize each image in the dataset locally; and when a new unknown image is provided, we perform the same analysis on it and compare the result to each of the images in the dataset. The way which we analyse the images is by characterizing the local patterns in each location in the image.

LBPH method for face recognition works better in different environments and light conditions, however, it will depend on our training and testing data sets. We will need around 10 different images of this person's face in order to be able to recognize him/her.

In our project we are taking 60 images of each person.

**2.3 Proposed Solution**

* Develop a Desktop Application for dynamic human authentication system for access control.
* The proposed solution consists of the creation of a face verification (1:1 match comparison) system using open-source face recognition and detection algorithms in order to implement it in events with access control, such as research facilities.
* An artificial light is added which helps to compensate the lack or the excess of light in the scene.

**3. REQUIREMENTS AND SPECIFICATIONS**

**3.1 Software Requirements**

* Windows 8/8.1/10
* Coding Language: Python 3.8.3
* GUI: PyQt5
* OpenCV 4.2.0
* numpy 1.18.4

**3.1.1 Functional Requirements**

* The app should be able to capture video while the user is registering.
* The app should be able to match the face captured by the camera with the data present in trainer.
* The app should be able to verify whether the person is authorized to enter the premises or not.

**3.1.2 Non-Functional Requirements**

Performance and Scalability:

* The app uses Histogram of gradients algorithm for face detection whose performance is very good.
* The app uses Local Binary Patterns Histogram (LBPH) face recognizer to improve precision in face recognition.
* The images of users are deleted once the model is trained. This greatly optimizes the space required.

Portability and Compatibility:

* The app can run on any operating system with latest version of required software installed.

Maintainability:

* The application is easy to maintain.
* The application does not consume much space.

Usability:

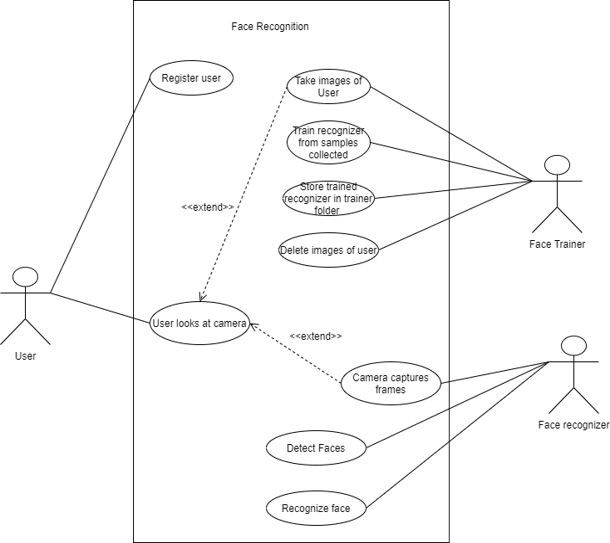
* The app is user friendly and has a good user interface.

**3.2 Hardware Requirements**

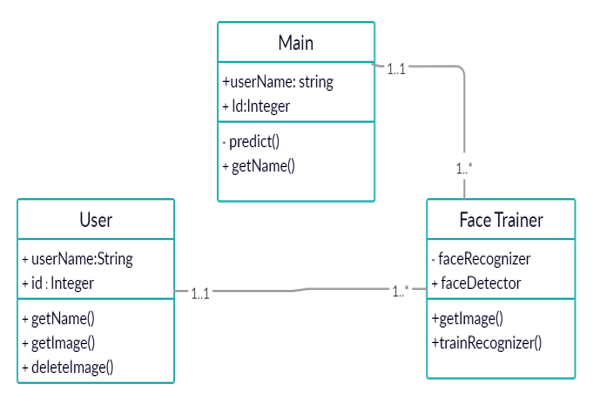
* System with Windows 10 to analyze the video captured by the camera.
* Hard Disk:100MB
* Ram:4GB
* Camera: (0.92 Megapixels, FrameRate-30 Fps)
* Artificial light

**4. Analysis and Design**

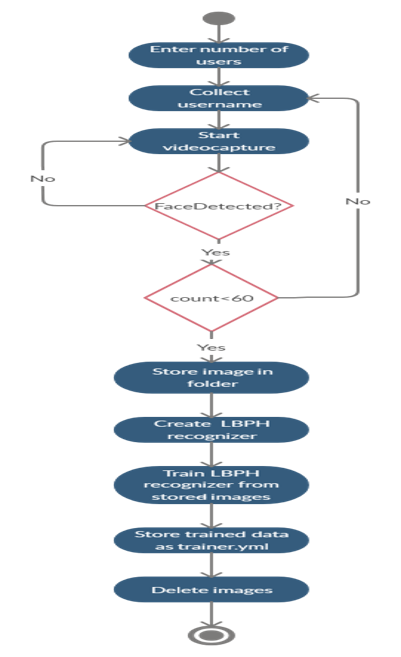
**4.1 Use Case Diagram**



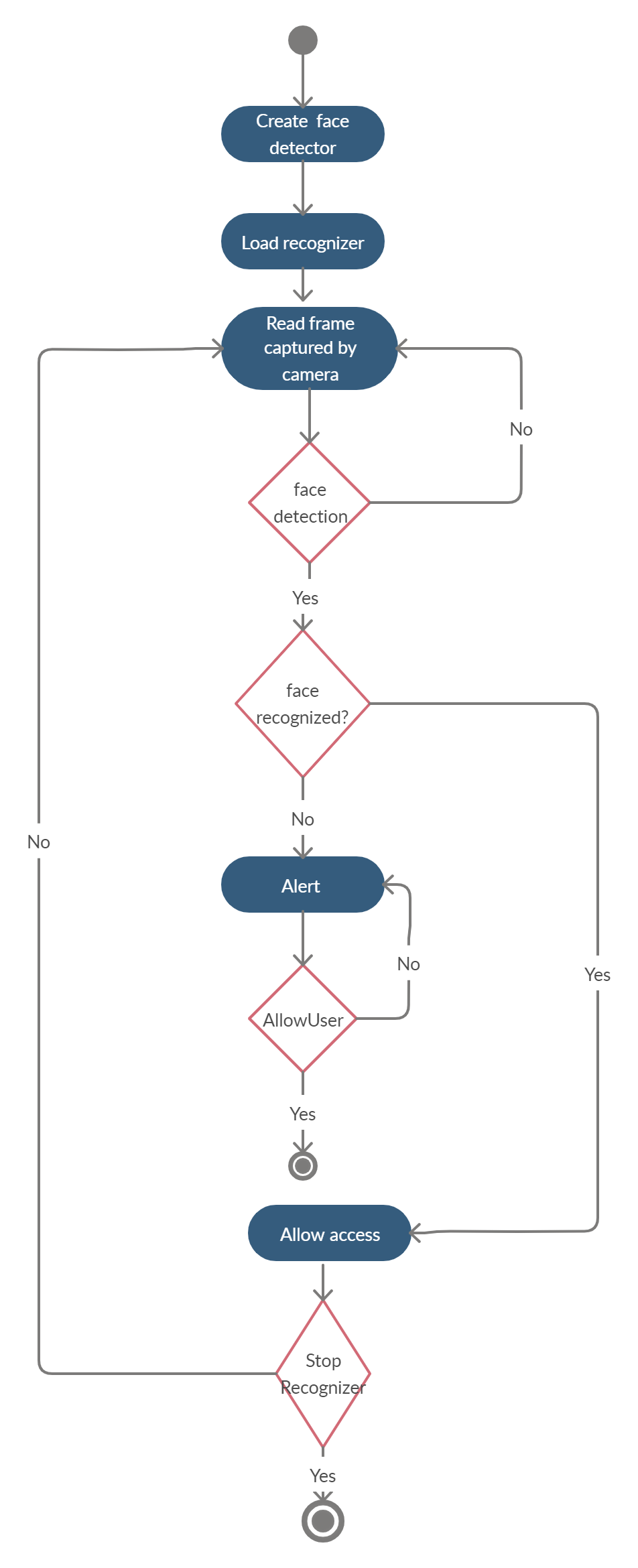
**4.2 Class Diagram**



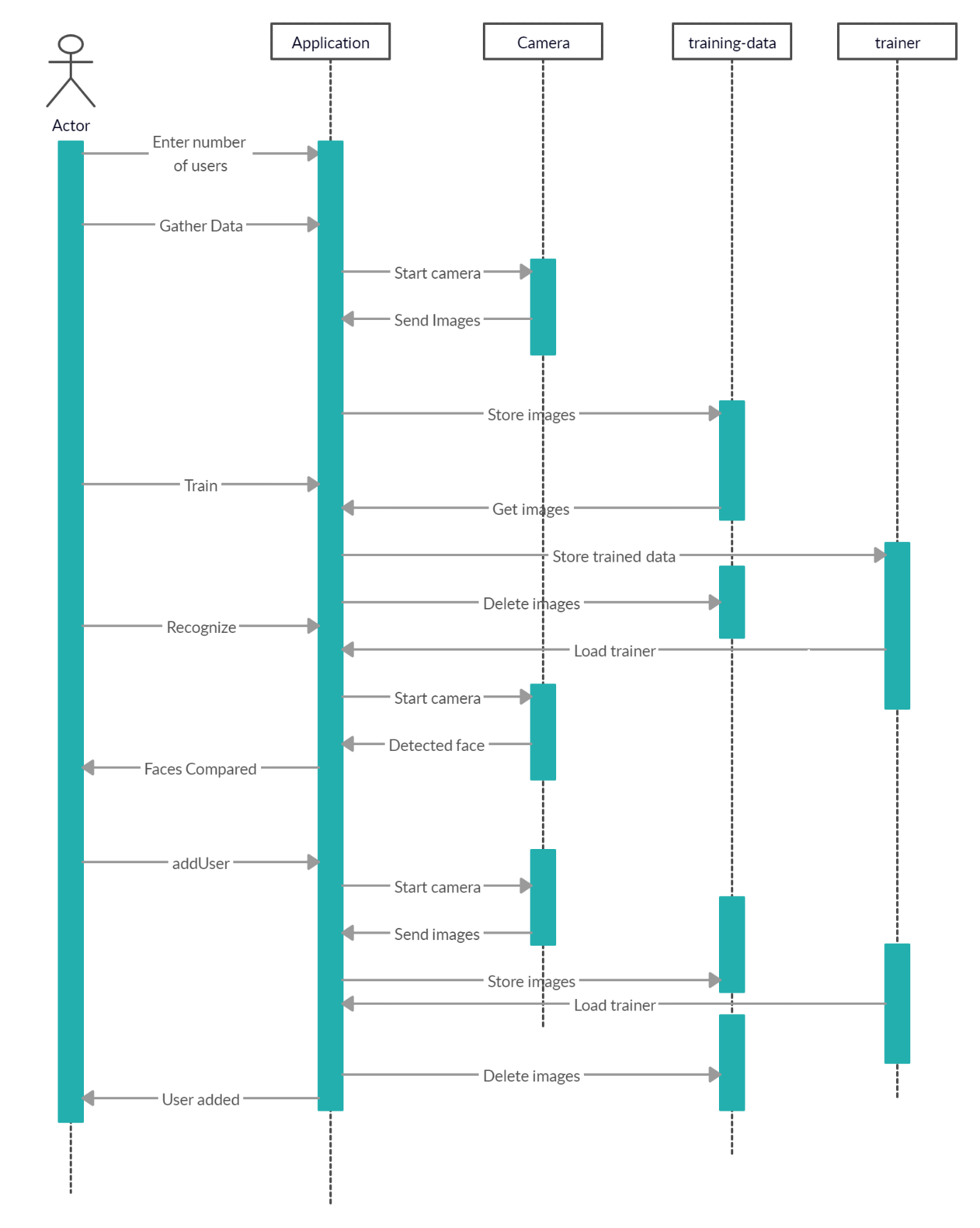
**4.3.1 Activity Diagrams**



**4.3.2 Activity Diagrams**



**4.4 Sequence Diagram**

****

**4.5 System Architecture**

A close up of a map

Description automatically generated

**4.6 Technology Description**

**Python**

* Python is an interpreted, high-level, general-purpose programming language.
* Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including structured object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

**Syntax and semantics**

* Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional.

**library**

* Python's large Standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as HTTP are supported. It includes modules for creating graphical interfaces, connecting to relational databases, generating pseudo numbers arithmetic with arbitrary-precision decimals, manipulating regular expression.

**Python Features**

* Python is a dynamic, high level, free open source and interpreted programming language.
* Easy to code.
* Free and Open Source.
* Object-Oriented Language.
* GUI Programming Support.
* High-Level Language.
* Extensible feature.
* Python is Portable language.

**Python Speciality**

* Python is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools and libraries. professionally,  is great for backend web development, data analysis, artificial intelligence, and scientific computing.

**Advantages of python**

* Python is an open-source language.
* It needs less coding. Python by nature has a very simple syntax.
* Python has a vibrant community.
* It can be used for any type of application such as web development, game development etc.,
* It has a wide range of libraries for any kind of software development.
* It is one of the most trending languages.
* Python is for everyone involved in software development.

**Facial Recognition library**

Import libraries

* Import os.
* Import cv2.
* Import numpy.

**Open cv**

* OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays.

**Installing OpenCV**

* Pip install opencv-python.
* Pip install opencv-contrib-python.

**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 NumP**y,** mathematical and logical operations on arrays can be performed.

**Installing numpy**

* Pip install numpy

**Facial algorithms**

* Haar Cascade.
* LBPH (Local Binary Pattern Histogram).

**Cascade Classifier Training.**

* Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face.
* Those XML files are stored in opencv/data/haarcascades/ folder to create face.
* face\_cascade =cv2.cascadeClassifier(‘face.xml’).

**Haar cascade Algorithm**

* Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of ​​ features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" .

A group of men posing for a photo

Description automatically generated

Fig 4.5.1 haarcascade classification

**LBPH (local Binary Pattern Histogram)**

Face Recognition using LBPH face recognition happens in two phases:

* Training
* Testing

**Training the data**

* First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID.
* We first enter the number of users in the text box provided
* In our application after the button gatherData is pressed the application asks for username and waits for ten seconds. Then it starts collecting images of users and stores them in the training data directory. The images are stored in the format User.id.count.jpeg. Here, id goes from 0 to number of users, count goes from 1 to 60 for each user.
* Now, we create an lbph face recognizer object using the constructor cv2.face.LBPHFaceRecognizer\_create()

recognizer=cv2.face.LBPHFaceRecognizer\_create()

* The above generated recognizer is trained with the images present in the directory training-data and the trained model is stored in the directory trainer as trainer.yml.
* After training, the images are deleted from the training-data directory.

**Testing**

* Once the model is trained, we write the trained data into another recognizer of LBPH face recognizer and use this to recognize faces.

**PyQt5**

PyQt5 is a cross platform graphical user interface toolkit. We can develop an interactive desktop application using PyQt5. It has many in-built tools for development of applications. QtDesigner is one of the widely used Graphical user interface designers available in PyQt5. The graphical user interface in this project is developed using PyQt5.

**Installation**

PyQt5 library – pip install pyqt5

Pyqt5 tools – pip install PyQt5-tools

**5.Impletmenting and Testing**

**5.1 front Page Screenshot**

A screenshot of a cell phone

Description automatically generated

Enter number of users

A screenshot of a cell phone

Description automatically generated

Click gatherDataA screenshot of a cell phone

Description automatically generated

Enter username

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

After 10 seconds of clicking gatherData the camera starts to capture images of user

A screenshot of a social media post

Description automatically generated

The application collects 60 images of each user

A screenshot of a social media post

Description automatically generated

After collecting images Click on Train

A screenshot of a video screen

Description automatically generated

The recognizer starts training

A screenshot of a computer screen

Description automatically generated

After training is completed

A screenshot of a social media post

Description automatically generated

To start Recognition, Click on Recognize

A screenshot of a cell phone

Description automatically generated

If an unknown person’s face is captured by the camera, that frame is displayed, and an alarm starts ringing.

A screenshot of a cell phone

Description automatically generated

If we click on allow User, the alarm will be stopped.

A screenshot of a cell phone

Description automatically generated

To add Another user into the existing users, we can click on add User button and enter his username into the textbox. The user must provide his images by staying in front of the camera. Then the previously trained model is updated using the new user images and a new id is given to that person.

To stop face recognition, we must click on Stop Recognizer button.

A screenshot of a cell phone

Description automatically generated

Example recognition images

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Code:

videoCap2.py

import sys

import cv2

import numpy as np

import os

import faceDetect as fd

from PyQt5 import QtCore

from PyQt5.QtCore import pyqtSlot

from PyQt5.QtGui import QImage,QPixmap

from PyQt5.QtWidgets import QDialog, QApplication, QMainWindow

from PyQt5.uic import loadUi

import datetime

import winsound

class Gather(QMainWindow):

def \_\_init\_\_(self):

super(Gather,self).\_\_init\_\_()

loadUi('videoCap1.ui',self)

self.gather.clicked.connect(self.onClicked)

self.train.clicked.connect(self.onClickedTrain)

self.recognize.clicked.connect(self.onClickedRecognize)

self.stopRecognizer.clicked.connect(self.onClickedStopRecognizer)

self.allowUser.clicked.connect(self.onClickedAllowUser)

self.addUser.clicked.connect(self.onClickedAddUser)

self.f=False

self.f1=False

@pyqtSlot()

def onClicked(self):

self.textbox1.setText('Collecting Images...')

cam=cv2.VideoCapture(0)

names=[]

face\_detector = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

n=(int)(self.num.text())

i=0

if not os.path.exists('training-data'):

os.makedirs('training-data')

while(i<n):

if(self.name.text()==""):

self.textBox.setText("Enter username "+str(i+1)+":")

cv2.waitKey(10000)

name1=self.name.text()

names.append(name1)

count=0

while(True):

ret, img=cam.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_detector.detectMultiScale(gray,scaleFactor=1.2,minNeighbors=5,minSize=(30,24))

for (x,y,w,h) in faces:

#cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)

count += 1

# Save the captured image into the datasets folder

cv2.imwrite("training-data/User." + str(i) + '.' + str(count) + ".jpg", img)

#cv2.imshow('image', img)

self.displayImage(img,1)

cv2.waitKey(65)

if(count>=60):

break

self.name.setText("")

i=i+1

self.textbox1.setText('Images Collected')

if not os.path.exists('usernames'):

os.makedirs('usernames')

path='usernames/users.txt'

#fw=open(path,w)

fp=open(path,'a')

for na in names:

#print(na)

fp.write(na+'/')

cam.release()

cv2.destroyAllWindows()

@pyqtSlot()

def onClickedTrain(self):

path = 'training-data'

recognizer = cv2.face.LBPHFaceRecognizer\_create()

#self.textbox1.setText('Traini')

faces=[]

ids=[]

faces,ids = self.getImagesAndLabels(path)

recognizer.train(faces, np.array(ids))

if not os.path.exists('trainer'):

os.makedirs('trainer')

recognizer.write('trainer/trainer.yml')

self.textbox1.setText('Model Trained')

images=os.listdir(path)

for image in images:

os.remove(path+'/'+image)

@pyqtSlot()

def onClickedRecognize(self):

self.f1=False

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read('trainer/trainer.yml')

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath)

font = cv2.FONT\_HERSHEY\_SIMPLEX

id = 0

names=[]

path='usernames/users.txt'

fp=open(path,'r')

s=fp.read().split('/')

for name in s:

#print(name)

names.append(name)

cam = cv2.VideoCapture(0)

cam.set(3, 431)

cam.set(4, 311)

#minW = 0.1\*cam.get(3)

#minH = 0.1\*cam.get(4)

#fl=False

while True:

ret, img =cam.read()

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(

gray,

scaleFactor = 1.05,

minNeighbors = 5,

minSize = (30,24),#before int(minW), int(minH)

)

for(x,y,w,h) in faces:

cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

if (confidence < 100):

name = names[id]

confidence = " {0}%".format(round(100 - confidence))

#self.f=False

else:

name = "unknown"

confidence = " {0}%".format(round(confidence-100))

self.f=True

cv2.putText(img, str(name), (x+5,y-5), font, 1, (255,255,255), 2)

cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

if self.f==True:

break

if self.f1==True:

break

#cv2.imshow('camera',img)

self.displayImage(img,1)

# if(self.f==True):

# duration = 300 # milliseconds

# freq = 440 # Hz

# winsound.Beep(freq, duration)

# cv2.waitKey(10)

if(self.f==True):

break

if(self.f1==True):

break

cv2.waitKey(10)

while(self.f==True):

duration = 300 # milliseconds

freq = 440 # Hz

winsound.Beep(freq, duration)

cv2.waitKey(10)

cam.release()

#window.close()

@pyqtSlot()

def onClickedStopRecognizer(self):

self.f1=True

#cam=cv2.VideoCapture(0)

#cam.release()

#window.close()

@pyqtSlot()

def onClickedAddUser(self):

face\_detector = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

data\_path='training-data'

path='usernames/users.txt'

fr=open(path,'r')

str1=fr.read().split('/')

id=int(len(str1))

id=id-1

cam=cv2.VideoCapture(0)

if(self.name.text()==""):

self.textBox.setText("Enter username :")

cv2.waitKey(10000)

name1=self.name.text()

count=0

while(True):

ret, img=cam.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_detector.detectMultiScale(gray,scaleFactor=1.2,minNeighbors=5,minSize=(30,24))

for (x,y,w,h) in faces:

#cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)

count += 1

# Save the captured image into the datasets folder

cv2.imwrite("training-data/User." + str(id) + '.' + str(count) + ".jpg", img)

#cv2.imshow('image', img)

self.displayImage(img,1)

cv2.waitKey(65)

if(count>=60):

break

self.name.setText("")

self.textbox1.setText('Images Collected')

path='usernames/users.txt'

fp=open(path,'a')

fp.write(name1+'/')

cam.release()

cv2.destroyAllWindows()

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read('trainer/trainer.yml')

faces,ids = self.getImagesAndLabels(data\_path)

recognizer.update(faces, np.array(ids))

recognizer.write('trainer/trainer.yml')

#print('new user face trained')

self.textbox1.setText('New User added')

images=os.listdir(data\_path)

for image in images:

os.remove(data\_path+'/'+image)

#print('new user added')

@pyqtSlot()

def onClickedAllowUser(self):

self.f=False

def displayImage(self,img,window=1):

qformat=QImage.Format\_Indexed8

if(len(img.shape)==3):

if(img.shape[2]==4):

qformat=QImage.Format\_RGBA888

else:

qformat=QImage.Format\_RGB888

img=QImage(img,img.shape[1],img.shape[0],qformat)

img=img.rgbSwapped()

self.imgLabel.setPixmap(QPixmap.fromImage(img))

self.imgLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)

def getImagesAndLabels(self,data\_path):

faces=[]

ids=[]

self.textbox1.setText('Training Image...')

img\_names=os.listdir(data\_path)

for img\_name in img\_names:

str=img\_name.split('.')

id=int(str[1])

img\_path=data\_path+"/"+img\_name

image=cv2.imread(img\_path)

#cv2.imshow("Training image...",cv2.resize(image,(320,240)))

self.displayImage(image,1)

cv2.waitKey(100)

face,rect =fd.detectFaces(image)

#print(len(faces))

if face is not None:

faces.append(face)

ids.append(id)

return faces,ids

app=QApplication(sys.argv)

window=Gather()

window.show()

try:

sys.exit(app.exec\_())

except:

print('exit')

cv2.destroyAllWindows()

**6. Conclusion and Future Scope**

**Conclusion:**

* Faces well recognized in optimal lighting conditions.
* The Application is User-Friendly
* Faces detected up to 1.5 meters.

**Future Enhancements:**

* Ability to capture faces from different angles.
* Enhance face detection in bad lighting conditions.
* Discard blurred images to improve precision.
* Send a notification of unknown person on premises to admin using cloud computing.

**References**

* [Train your haarcascade - https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html](https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html)
* [www.pyimagesearch.com](http://www.pyimagesearch.com/)
* <https://www.superdatascience.com/blogs/opencv-face-recognition>
* <https://www.youtube.com/channel/UC4JX40jDee_tINbkjycV4Sg>
* <https://www.youtube.com/watch?v=VGxsue9ZNww&t=543s>

**Appendices**