MINI PROJECT ON SMART CCTV

RED SURVEILLANCE SYSTEM

PRESENTED BY: **KARTIK SOLANKI**
UNIV ROLL NO: **2118685**
STUDENT ID: **21011414**
CLASS ROLL NO: **39**

PRESENTED TO: **DR. DIVESH KUMAR**

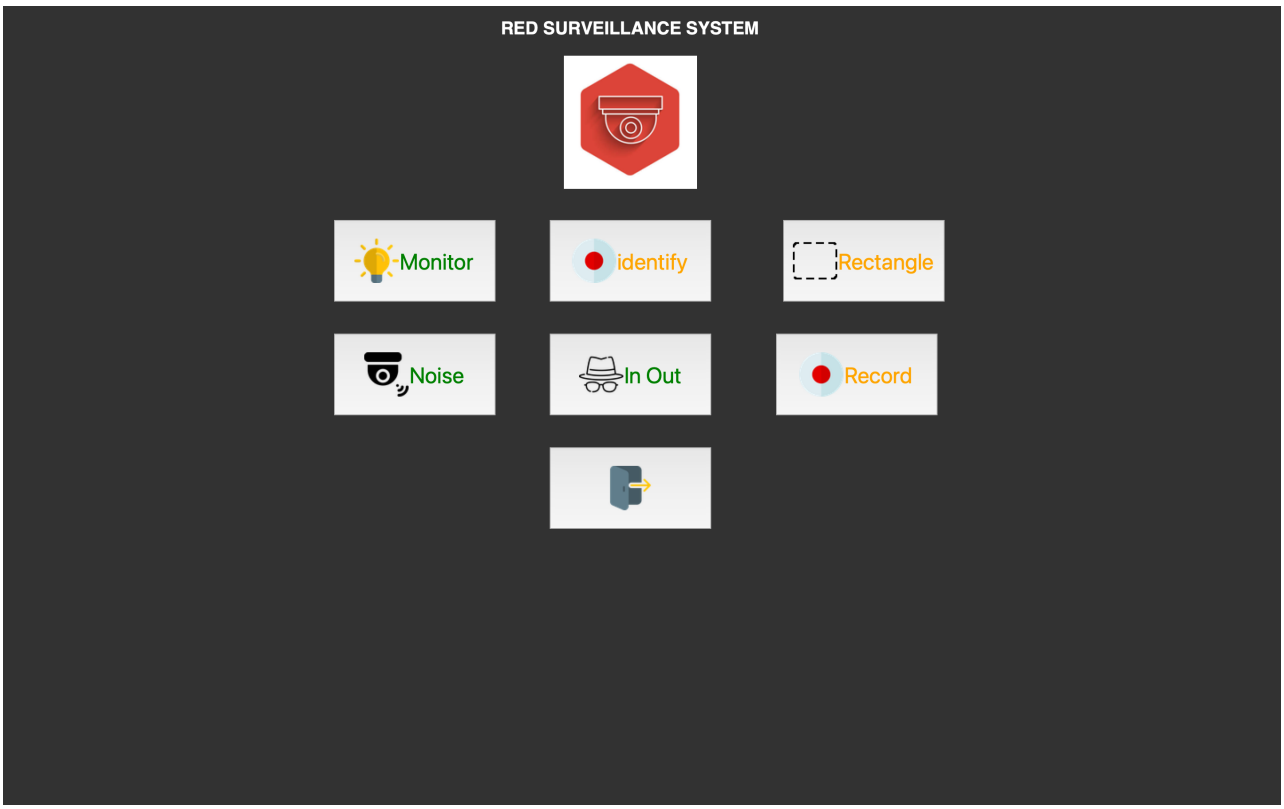
About Project

- **Project Name** – RED SURVEILLANCE SYSTEM

- **Short Description** – this is a python GUI application which can run on any operating system, uses webcam and has number of features which are not in normal cctv, discussed in detail below pages.
- **Programming Language –** Python
- **Features -**
  - Anti-thief
  - Noise Detection
  - Visitors Counting
  - Normal Recording
  - Face Identification

This is a Project built using latest Programming Language and highly evolving Computer Science field which is “Computer Vision”. Which means this project allow computer to watch or in other words it gives vision capability to computers.

Project Sneak Peak



It got nice **GUI** and every button is supported by using beautiful icon.

Monitor – allow to detect what thing is stolen from frame

Identify – Finds the family members(it has to be trained first)

Noise – Finds the motions in the frame

In Out – Finds who entered and Gone from room.

Index

chapter 1 – Technical Details
1.1 Technology and Algorithms used
1.2 Process model
1.3 Software Requirements
1.4 Hardware Requirements

chapter 2 – Coding
2.1 Source Code

chapter 3 – In use
3.1 Showing the work

chapter 4 - future scopes

chapter 5 - Refrences

CHAPTER I
Technical Details

For this project we have used various latest technologies which will be evaluated in this chapter with every details of why it is used.
We'll divide this section of explaination of technolgy based on modules/features in project.
But first lets see the language used in this project.

Language Used,

We have used **Python language** as it is very new and also comes with so many features like we can do Machine Learning, Computer Vision and Also make GUI application with ease.

Reasons for Selecting this language :

1 – Short and Concise Language.
2 – Easy to Learn and use.
3 – Good Technical support over Internet
4 – Many Package for different tasks.
5 – Run on Any Platform.
6 – Modern and OOP language

Well these are just the minor points from our sides Python is just a lot more than this.

Some more notes about python,

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.There are two major Python versions: **Python 2 and Python 3**

Some specific features of Python are as follows:

- an *interpreted* (as opposed to *compiled*) language. Contrary to e.g. C or Fortran, one does not compile Python code before executing it. In addition, Python can be used **interactively**: many Python interpreters are available, from which commands and scripts can be executed.
- a free software released under an **open-source** license: Python can be used and distributed free of charge, even for building commercial software.
- **multi-platform**: Python is available for all major operating systems, Windows, Linux/Unix, MacOS X, most likely your mobile phone OS, etc.
- a very readable language with clear non-verbose syntax
- a language for which a large variety of high-quality packages are available for various applications, from web frameworks to scientific computing.
- a language very easy to interface with other languages, in particular C and C++.

- Some other features of the language are illustrated just below. For example, Python is an object-oriented language, with dynamic typing (the same variable can contain objects of different types during the course of a program).

Below are the different features which can performed by using this minor project:

1. **Monitor**
2. **Identify the family member**
3. **Detect for Noises**
4. **Visitors in room detection**

So lets see each features one by one.

Monitor Feature :

This feature is used to find what is the thing which is stolen from the frame which is visible to webcam. Meaning It constantly monitors the frames and checks which object or thing from the frame has been taken away by the thief.

This uses **Structural Similarity** to find the differences in the two frames. The two frames are captured first when noise was not happened and second when noise stopped happening in the frame.

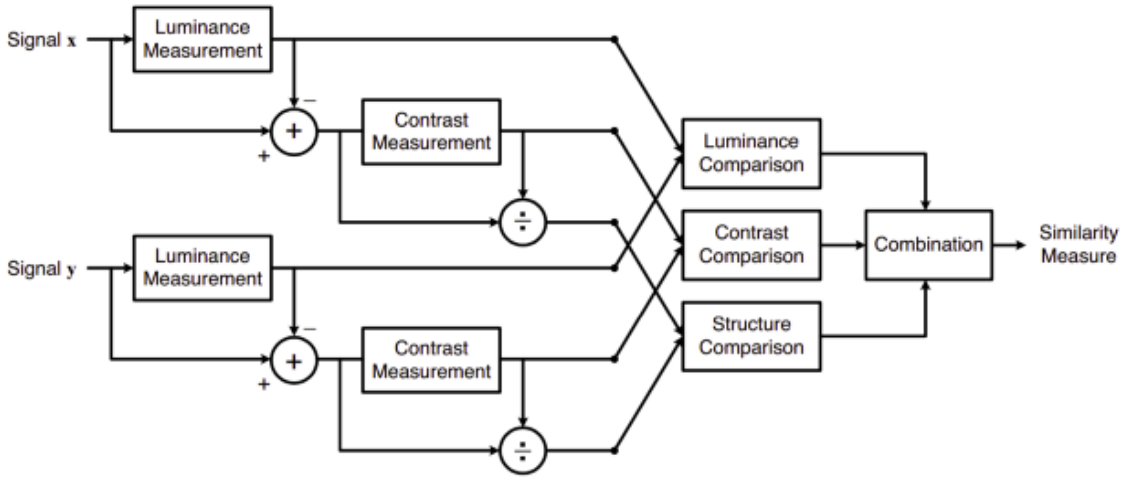
SSIM is used as a metric to measure the *similarity* between two given images. As this technique has been around since 2004, a lot of material exists explaining the theory behind SSIM but very few resources go deep into the details, that too specifically for a

gradient-based implementation as SSIM is often used as a loss function.

The Structural Similarity Index (SSIM) metric extracts 3 key *features* from an image:

- **Luminance**
- **Contrast**
- **Structure**

The comparison between the two images is performed on the basis of these 3 features.



This system calculates the *Structural Similarity Index* between 2 given images which is a value between -1 and +1. *A value of +1* indicates that the 2 given images are **very similar or the same** while a *value of -1* indicates the 2 given images are **very**

different. Often these values are adjusted to be in the range [0, 1], where the extremes hold the same meaning.

Luminance: Luminance is measured by *averaging* over all the pixel values. Its denoted by μ (Mu) and the formula is given below,

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{2}$$

The luminance comparison function $l(\mathbf{x}, \mathbf{y})$ is then a function of μ_x and μ_y.

Luminance: Luminance is measured by *averaging* over all the pixel values. Its denoted by μ (Mu) and the formula is given below,

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}. \tag{4}$$

The contrast comparison $c(\mathbf{x}, \mathbf{y})$ is then the comparison of σ_x and σ_y.

Structure: The structural comparison is done by using a consolidated formula (more on that later) but in essence, we divide the input signal with its *standard deviation* so that the result has unit standard deviation which allows for a more robust comparison.

$$(\mathbf{x} - \mu_x)/\sigma_x$$

Luckily , thanks to skimage package in python we dont have to replicate all this mathematical calculation in python since

skimage has pre build feature that d o all of these tasks for us with just calling its in-built function.

We just have to feed in two images/frames which we have captured earlier, so we just feed them in and its gives us out the masked image with score.

Identify the Family Member feature

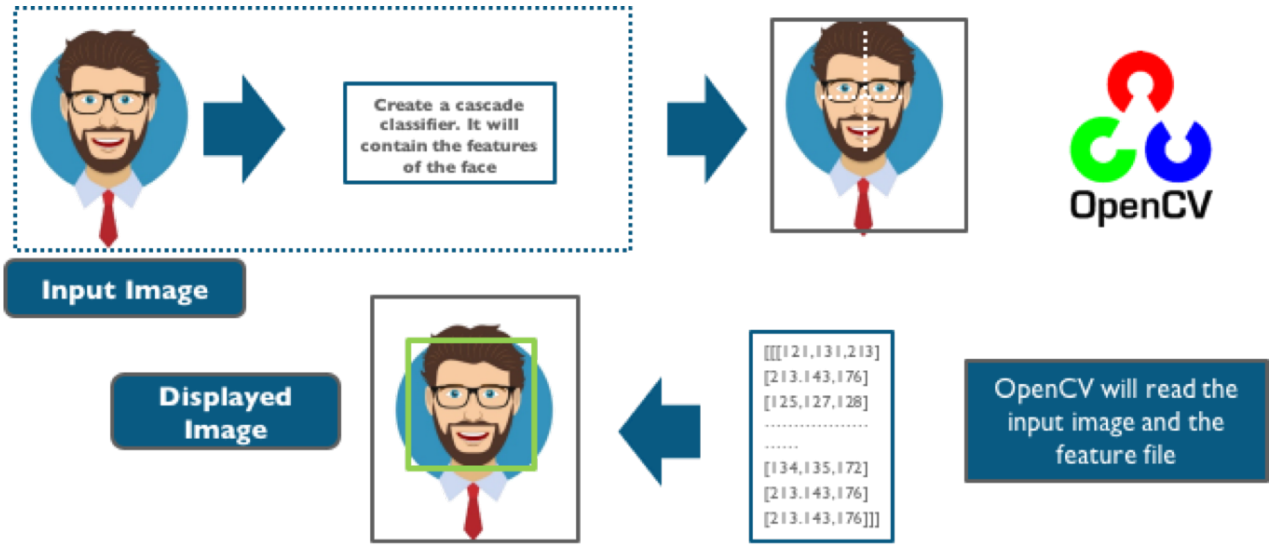
This feature is very useful feature of our minor project, It is used to find if the person the frame is known or not. It do this in two steps :

1 – Find the faces in the frames
2 – Use LBPH face recognizer algorithm to predict the person from already trained model.

So lets divide this in following categories,

1 – Detecting faces in the frames

This is done via **Haarcascade** classifiers which are again in-built in openCV module of python.



Cascade classifier, or namely *cascade of boosted classifiers working with haar-like features*, is a special case of **ensemble learning**, called **boosting**. It typically relies on **Adaboost** classifiers (and other models such as Real Adaboost, Gentle Adaboost or Logitboost).

Cascade classifiers are trained on a few hundred sample images of image that contain the object we want to detect, and other images that do not contain those images.

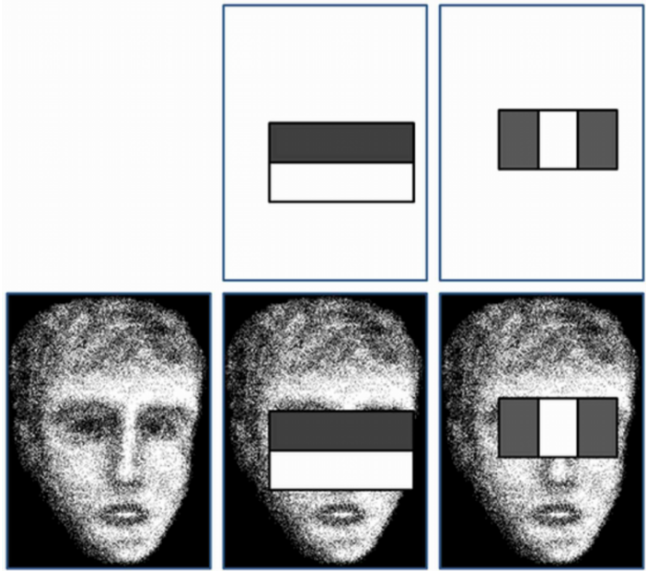
There are some common features that we find on most common human faces :

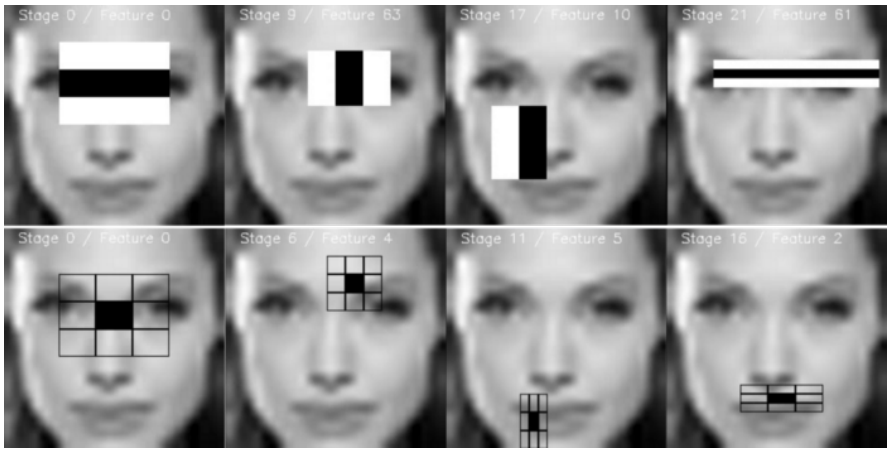
- *a dark eye region compared to upper-cheeks*
- *a bright nose bridge region compared to the eyes*
- *some specific location of eyes, mouth, nose…*

The characteristics are called **Haar Features**. The feature extraction process will look like this :

Haar features are similar to these **convolution kernels** which are used to detect the presence of that feature in the given image.

For doing all this stuff openCV module in python language has inbuild function called cascadeclassifier which we have used in order to detect for faces in the frame



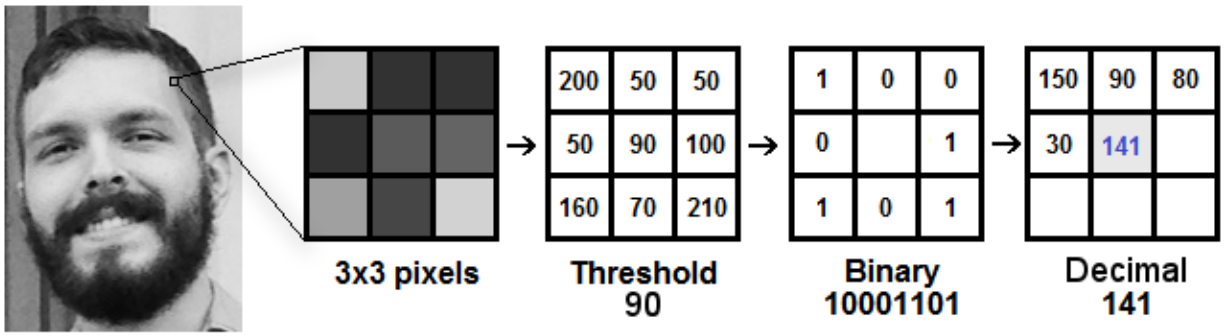


2 – Using LBPH for face recognition

So now we have detected for faces in the frame and this is the time to identify it and check if it is in the dataset which we've used to train our lbph model.

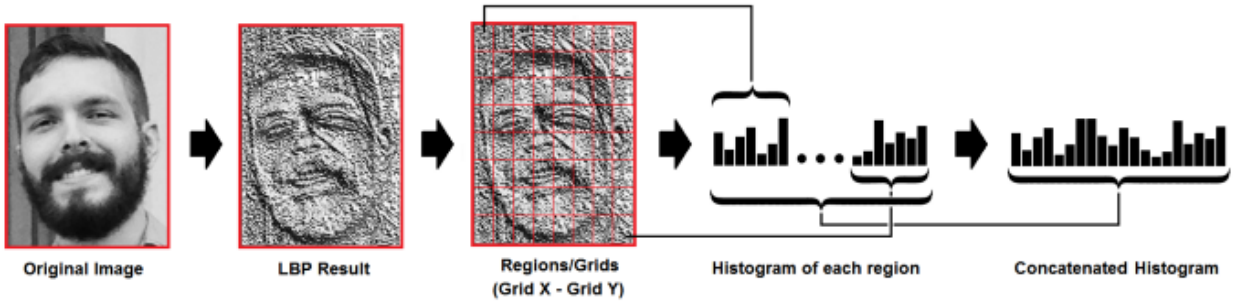
The LBPH uses 4 parameters:

- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8



The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**. Which is shown perfectly via the above image.

Extracting the Histograms: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can



be seen in the following image:

And after all this the model is trained and later onwhen we want to make predictions the same steps are applied to the make and its histograms are compared with already trained model and in such way this feature works.

3 – Detect for Noises in the frame

This feature is used to find the noises in the frames well this is something you would find in most of the cctv's but in this module we'll see how it works.

Talking in simple way all the frames are continously analyzed and checked for noises. Noise in checked in the consecutive frames. Simply we do the absolute difference between two frames and in this way the difference of two images are analyzed and Contours(boundaries of the motion are detected) and if there are no boundries then no motion and if there is any there is motion.

frame1

10	90	16	16
0	11	11	11
18	30	33	33
18	18	18	18

frame2

10	90	16	16
0	13	17	11
18	34	31	33
18	17	19	18

frame2 - frame1

0	0	0	0
0	2	6	0
0	4	-2	0
0	-1	1	0

abs (frame2 - frame1)

0	0	0	0
0	2	6	0
0	4	2	0
0	1	1	0

As you would know all images are just integer/ float values of pixels which tells the brightness of pixel and similarly every pixel has that valules of brigthness.

So we just do simply absolute difference because negative will make no sense at all.

4 – Visitors in room detection

This is the feature which can detect if someone has entered in the room or gone out.

So it works using following steps:

1 – It first detect for noises in the frame.
2 – Then if any moiton happen it find from which side does that happen either left or right.
3 – Last if checks if motion from left ended to right then its will detect it as entered and capture the frame.
Or vise-versa.

So there is not complex mathematics going on around in this specific feature.

So basically to know from which side does the motion happened we first detect for motion and later on we draw rectangle over noise and last step is we check the co-ordinates if those points lie on left side then it is classified as left motion.

Process model used

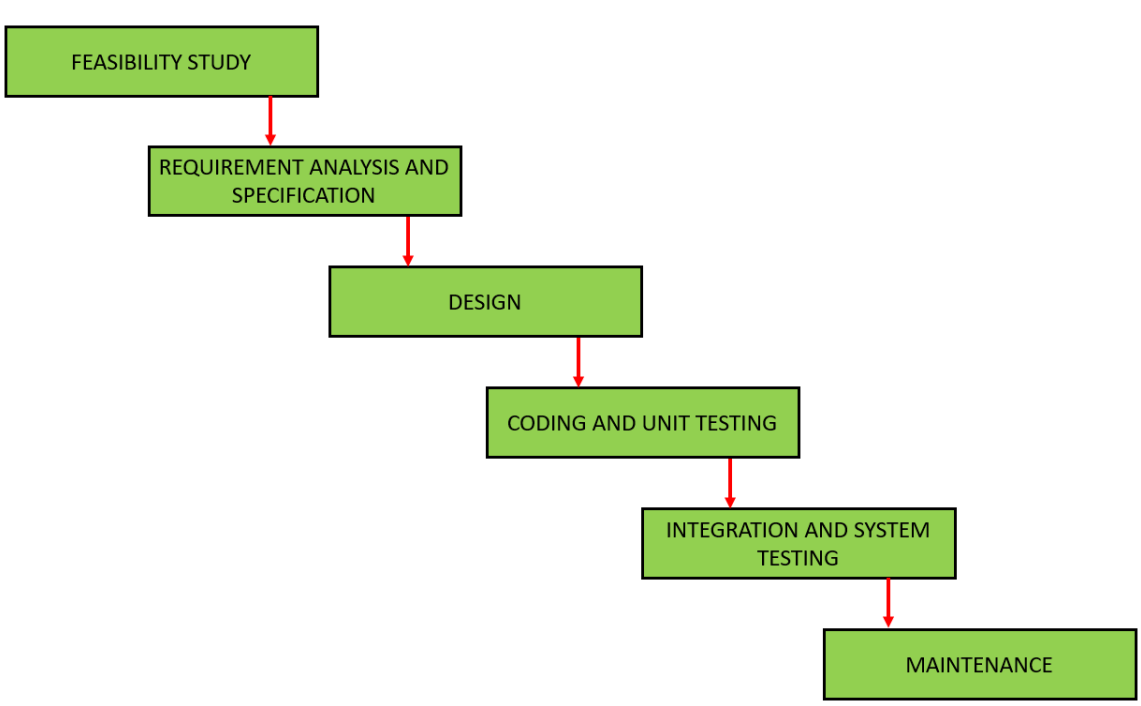
For this model we have used **waterfall model,** since it was not huge project at all.

Reasons behind choosing waterfall model -

1. Good for minor projects.
2. Easy to follow.
3. Well tracking for small projects.
4. Well time managed.

Waterfall model,

Classical waterfall model is the basic **software development life cycle** model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.
Classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of



Software Requirements

Since this is a software hence it will have to run on some hardware and operating system obvously, so below are the requirements to run this software :

- Windows/Linux/Mac OS any version, hence it can run on any platform.
- Python3, it need python to be installed in your system to run this successfully.
- Packages in python -
  - openCV
  - skimage
  - numpy
  - tkinter

Hardware Requirements

In terms of hardware requirements there is not much required at all but still below requirements are must :

- Working PC or Laptop
- Webcam with drivers installed
- Flashlight/ LED if using this at night.

Technology used to make this project :

- as mentioned earlier **Python** language is used.
- Sublime Text Editor is used to write the code.
- Linux Mint os is used to run and create this minor project.
- HP-ay503tx laptop is used.
  - Core i5 dual core
  - 240GB ssd
  - 8GB RAM
  - In-built webcam hp-truevision
- Terminal to run the code

Platforms already tested,

It is tested on Linux Mint, Linux Ubunutu, Windows 7, Windows 10.

CHAPTER II
Coding

The coding part in below with specific modules discussed above.

Main.py

```
import tkinter as tk
import tkinter.font as font
from in_out import in_out
from motion import noise
from rect_noise import rect_noise
from record import record
from PIL import Image, ImageTk
from find_motion import find_motion
from identify import maincall

window = tk.Tk()
window.title("Smart cctv")
window.iconphoto(False, tk.PhotoImage(file='mn.png'))
window.geometry('1080x700')



frame1 = tk.Frame(window)

label_title = tk.Label(frame1, text="Smart cctv Camera")
label_font = font.Font(size=35, weight='bold',family='Helvetica')
label_title['font'] = label_font
label_title.grid(pady=(10,10), column=2)



icon = Image.open('icons/spy.png')
icon = icon.resize((150,150), Image.ANTIALIAS)
icon = ImageTk.PhotoImage(icon)
label_icon = tk.Label(frame1, image=icon)
label_icon.grid(row=1, pady=(5,10), column=2)

btn1_image = Image.open('icons/lamp.png')
btn1_image = btn1_image.resize((50,50), Image.ANTIALIAS)
btn1_image = ImageTk.PhotoImage(btn1_image)

btn2_image = Image.open('icons/rectangle-of-cutted-line-geometrical-shape.png')
btn2_image = btn2_image.resize((50,50), Image.ANTIALIAS)
btn2_image = ImageTk.PhotoImage(btn2_image)

btn5_image = Image.open('icons/exit.png')
btn5_image = btn5_image.resize((50,50), Image.ANTIALIAS)
btn5_image = ImageTk.PhotoImage(btn5_image)

btn3_image = Image.open('icons/security-camera.png')
btn3_image = btn3_image.resize((50,50), Image.ANTIALIAS)
btn3_image = ImageTk.PhotoImage(btn3_image)

btn6_image = Image.open('icons/incognito.png')
btn6_image = btn6_image.resize((50,50), Image.ANTIALIAS)
btn6_image = ImageTk.PhotoImage(btn6_image)
```

```
btn4_image = Image.open('icons/recording.png')
btn4_image = btn4_image.resize((50,50), Image.ANTIALIAS)
btn4_image = ImageTk.PhotoImage(btn4_image)

btn7_image = Image.open('icons/recording.png')
btn7_image = btn7_image.resize((50,50), Image.ANTIALIAS)
btn7_image = ImageTk.PhotoImage(btn7_image)



# ---------- Button ----------#
btn_font = font.Font(size=25)
btn1 = tk.Button(frame1, text='Monitor', height=90, width=180, fg='green',command =
find_motion, image=btn1_image, compound='left')
btn1['font'] = btn_font
btn1.grid(row=3, pady=(20,10))

btn2 = tk.Button(frame1, text='Rectangle', height=90, width=180, fg='orange',
command=rect_noise, compound='left', image=btn2_image)
btn2['font'] = btn_font
btn2.grid(row=3, pady=(20,10), column=3, padx=(20,5))

btn_font = font.Font(size=25)
btn3 = tk.Button(frame1, text='Noise', height=90, width=180, fg='green', command=noise,
image=btn3_image, compound='left')
btn3['font'] = btn_font
btn3.grid(row=5, pady=(20,10))

btn4 = tk.Button(frame1, text='Record', height=90, width=180, fg='orange', command=record,
image=btn4_image, compound='left')
btn4['font'] = btn_font
btn4.grid(row=5, pady=(20,10), column=3)



btn6 = tk.Button(frame1, text='In Out', height=90, width=180, fg='green', command=in_out,
image=btn6_image, compound='left')
btn6['font'] = btn_font
btn6.grid(row=5, pady=(20,10), column=2)

btn5 = tk.Button(frame1, height=90, width=180, fg='red', command=window.quit,
image=btn5_image)
btn5['font'] = btn_font
btn5.grid(row=6, pady=(20,10), column=2)

btn7 = tk.Button(frame1, text="identify", fg="orange",command=maincall, compound='left',
image=btn7_image, height=90, width=180)
btn7['font'] = btn_font
btn7.grid(row=3, column=2, pady=(20,10))

frame1.pack()
window.mainloop()
```

Monitors is divided into two modules.

1. find_noise.py
2. spot_diff.py

```
import cv2
from spot_diff import spot_diff
import time
import numpy as np


def find_motion():

motion_detected = False
is_start_done = False

cap = cv2.VideoCapture(0)

check = []

print("waiting for 2 seconds")
time.sleep(2)
frame1 = cap.read()

_, frm1 = cap.read()
frm1 = cv2.cvtColor(frm1, cv2.COLOR_BGR2GRAY)



while True:
_, frm2 = cap.read()
frm2 = cv2.cvtColor(frm2, cv2.COLOR_BGR2GRAY)

diff = cv2.absdiff(frm1, frm2)

_, thresh = cv2.threshold(diff, 30, 255, cv2.THRESH_BINARY)

contors = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]

#look at it
contors = [c for c in contors if cv2.contourArea(c) > 25]



if len(contors) > 5:
cv2.putText(thresh, "motion detected", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 255)
motion_detected = True
is_start_done = False
```

```
elif motion_detected and len(contors) < 3:
if (is_start_done) == False:
start = time.time()
is_start_done = True
end = time.time()

end = time.time()

print(end-start)
if (end - start) > 4:
frame2 = cap.read()
cap.release()
cv2.destroyAllWindows()
x = spot_diff(frame1, frame2)
if x == 0:
print("runnig again")
return

else:
print("found motion sending mail")
return

else:
cv2.putText(thresh, "no motion detected", (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 2, 255)

cv2.imshow("winname", thresh)

_, frm1 = cap.read()
frm1 = cv2.cvtColor(frm1, cv2.COLOR_BGR2GRAY)

if cv2.waitKey(1) == 27:

break

return
```

spot_diff.py

```
import cv2
import time
from skimage.metrics import structural_similarity
from datetime import datetime

def spot_diff(frame1, frame2):

frame1 = frame1[1]
frame2 = frame2[1]

g1 = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
g2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)

g1 = cv2.blur(g1, (2,2))
g2 = cv2.blur(g2, (2,2))

(score, diff) = structural_similarity(g2, g1, full=True)

print("Image similarity", score)

diff = (diff * 255).astype("uint8")
thresh = cv2.threshold(diff, 100, 255, cv2.THRESH_BINARY_INV)[1]

contors = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[0]
contors = [c for c in contors if cv2.contourArea(c) > 50]

if len(contors):
for c in contors:

x,y,w,h = cv2.boundingRect(c)

cv2.rectangle(frame1, (x,y), (x+w, y+h), (0,255,0), 2)

else:
print("nothing stolen")
return 0

cv2.imshow("diff", thresh)
cv2.imshow("win1", frame1)
cv2.imwrite("stolen/"+datetime.now().strftime('%-y-%-m-%-d-%H:%M:%S')
+".jpg", frame1)
cv2.waitKey(0)
cv2.destroyAllWindows()

return 1
```

identify.py

```
import cv2
import os
import numpy as np
import tkinter as tk
import tkinter.font as font

def collect_data():
name = input("Enter name of person : ")

count = 1
ids = input("Enter ID: ")

cap = cv2.VideoCapture(0)

filename = "haarcascade_frontalface_default.xml"

cascade = cv2.CascadeClassifier(filename)

while True:
_, frm = cap.read()

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

faces = cascade.detectMultiScale(gray, 1.4, 1)

for x,y,w,h in faces:
cv2.rectangle(frm, (x,y), (x+w, y+h), (0,255,0), 2)
roi = gray[y:y+h, x:x+w]

cv2.imwrite(f"persons/{name}-{count}-{ids}.jpg", roi)
count = count + 1
cv2.putText(frm, f"{count}", (20,20), cv2.FONT_HERSHEY_PLAIN, 2,
(0,255,0), 3)
cv2.imshow("new", roi)


cv2.imshow("identify", frm)

if cv2.waitKey(1) == 27 or count > 200:
cv2.destroyAllWindows()
cap.release()
train()
break
```

```
def train():
print("training part initiated !")

recog = cv2.face.LBPHFaceRecognizer_create()

dataset = 'persons'

paths = [os.path.join(dataset, im) for im in os.listdir(dataset)]

faces = []
ids = []
labels = []
for path in paths:
labels.append(path.split('/')[-1].split('-')[0])

ids.append(int(path.split('/')[-1].split('-')[2].split('.')[0]))

faces.append(cv2.imread(path, 0))

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

recog.save('model.yml')

return

def identify():
cap = cv2.VideoCapture(0)

filename = "haarcascade_frontalface_default.xml"

paths = [os.path.join("persons", im) for im in os.listdir("persons")]
labelslist = []
for path in paths:
if path.split('/')[-1].split('-')[0] not in labelslist:
labelslist.append(path.split('/')[-1].split('-')[0])

print(labelslist)
recog = cv2.face.LBPHFaceRecognizer_create()

recog.read('model.yml')

cascade = cv2.CascadeClassifier(filename)

while True:
_, frm = cap.read()

gray = cv2.cvtColor(frm, cv2.COLOR_BGR2GRAY)
```

```
faces = cascade.detectMultiScale(gray, 1.4, 1)

for x,y,w,h in faces:
cv2.rectangle(frm, (x,y), (x+w, y+h), (0,255,0), 2)
roi = gray[y:y+h, x:x+w]

label = recog.predict(roi)

cv2.putText(frm, f"{labelslist[label[0]]}", (x,y),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 3)


cv2.imshow("identify", frm)

if cv2.waitKey(1) == 27:
cv2.destroyAllWindows()
cap.release()
break

def maincall():


root = tk.Tk()

root.geometry("480x100")
root.title("identify")

label = tk.Label(root, text="Select below buttons ")
label.grid(row=0, columnspan=2)
label_font = font.Font(size=35, weight='bold',family='Helvetica')
label['font'] = label_font

btn_font = font.Font(size=25)

button1 = tk.Button(root, text="Add Member ", command=collect_data,
height=2, width=20)
button1.grid(row=1, column=0, pady=(10,10), padx=(5,5))
button1['font'] = btn_font

button2 = tk.Button(root, text="Start with known ", command=identify, height=2,
width=20)
button2.grid(row=1, column=1,pady=(10,10), padx=(5,5))
button2['font'] = btn_font
root.mainloop()

return
```

in_out.py

```
import cv2
from datetime import datetime
def in_out():
    cap = cv2.VideoCapture(0)


    right, left = "", ""

    while True:
        _, frame1 = cap.read()
        frame1 = cv2.flip(frame1, 1)
        _, frame2 = cap.read()
        frame2 = cv2.flip(frame2, 1)

        diff = cv2.absdiff(frame2, frame1)

        diff = cv2.blur(diff, (5,5))

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

        _, threshd = cv2.threshold(gray, 40, 255, cv2.THRESH_BINARY)

        contr, _ = cv2.findContours(threshd, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        x = 300
        if len(contr) > 0:
            max_cnt = max(contr, key=cv2.contourArea)
            x,y,w,h = cv2.boundingRect(max_cnt)
            cv2.rectangle(frame1, (x, y), (x+w, y+h), (0,255,0), 2)
            cv2.putText(frame1, "MOTION", (10,80),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0), 2)


        if right == "" and left == "":
            if x > 500:
                right = True

            elif x < 200:
                left = True

        elif right:
                if x < 200:
                    print("to left")
                    x = 300
                    right, left = "", ""
                    cv2.imwrite(f"visitors/in/{datetime.now().strftime('%-y-%-m-%-
```

```
    elif left:
            if x > 500:
                print("to right")
                x = 300
                right, left = "", ""
                cv2.imwrite(f"visitors/out/{datetime.now().strftime('%-y-
%-m-%-d-%H:%M:%S')}.jpg", frame1)



    cv2.imshow("", frame1)

    k = cv2.waitKey(1)

    if k == 27:
        cap.release()
        cv2.destroyAllWindows()
        break
```

motion.py

```
import cv2

def noise():
    cap = cv2.VideoCapture(0)

    while True:
        _, frame1 = cap.read()
        _, frame2 = cap.read()

        diff = cv2.absdiff(frame2, frame1)
        diff = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)

        diff = cv2.blur(diff, (5,5))
        _, thresh = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)

        contr, _ = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        if len(contr) > 0:
            max_cnt = max(contr, key=cv2.contourArea)
            x,y,w,h = cv2.boundingRect(max_cnt)
            cv2.rectangle(frame1, (x, y), (x+w, y+h), (0,255,0), 2)
            cv2.putText(frame1, "MOTION", (10,80),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0), 2)

        else:
            cv2.putText(frame1, "NO-MOTION", (10,80),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,0,255), 2)

        cv2.imshow("esc. to exit", frame1)

        if cv2.waitKey(1) == 27:
            cap.release()
            cv2.destroyAllWindows()
            break
```

finding noises in the rectangle.

```
import cv2

donel = False
doner = False
x1,y1,x2,y2 = 0,0,0,0


def select(event, x, y, flag, param):
    global x1,x2,y1,y2,donel, doner
    if event == cv2.EVENT_LBUTTONDOWN:
        x1,y1 = x,y
        donel = True
    elif event == cv2.EVENT_RBUTTONDOWN:
        x2,y2 = x,y
        doner = True
        print(doner, donel)

def rect_noise():

    global x1,x2,y1,y2, donel, doner
    cap = cv2.VideoCapture(0)



    cv2.namedWindow("select_region")
    cv2.setMouseCallback("select_region", select)

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

        cv2.imshow("select_region", frame)

        if cv2.waitKey(1) == 27 or doner == True:
            cv2.destroyAllWindows()
            print("gone--")
            break

    while True:
        _, frame1 = cap.read()
        _, frame2 = cap.read()

        frame1only = frame1[y1:y2, x1:x2]
        frame2only = frame2[y1:y2, x1:x2]

        diff = cv2.absdiff(frame2only, frame1only)
        diff = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
```

```
        diff = cv2.blur(diff, (5,5))
        _, thresh = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)

        contr, _ = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        if len(contr) > 0:
            max_cnt = max(contr, key=cv2.contourArea)
            x,y,w,h = cv2.boundingRect(max_cnt)
            cv2.rectangle(frame1, (x+x1, y+y1), (x+w+x1, y+h+y1), (0,255,0), 2)
            cv2.putText(frame1, "MOTION", (10,80),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0), 2)

        else:
            cv2.putText(frame1, "NO-MOTION", (10,80),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,0,255), 2)

        cv2.rectangle(frame1, (x1,y1), (x2, y2), (0,0,255), 1)
        cv2.imshow("esc. to exit", frame1)

        if cv2.waitKey(1) == 27:
            cap.release()
            cv2.destroyAllWindows()
            break
```

At-last this is most required feature which is recording.

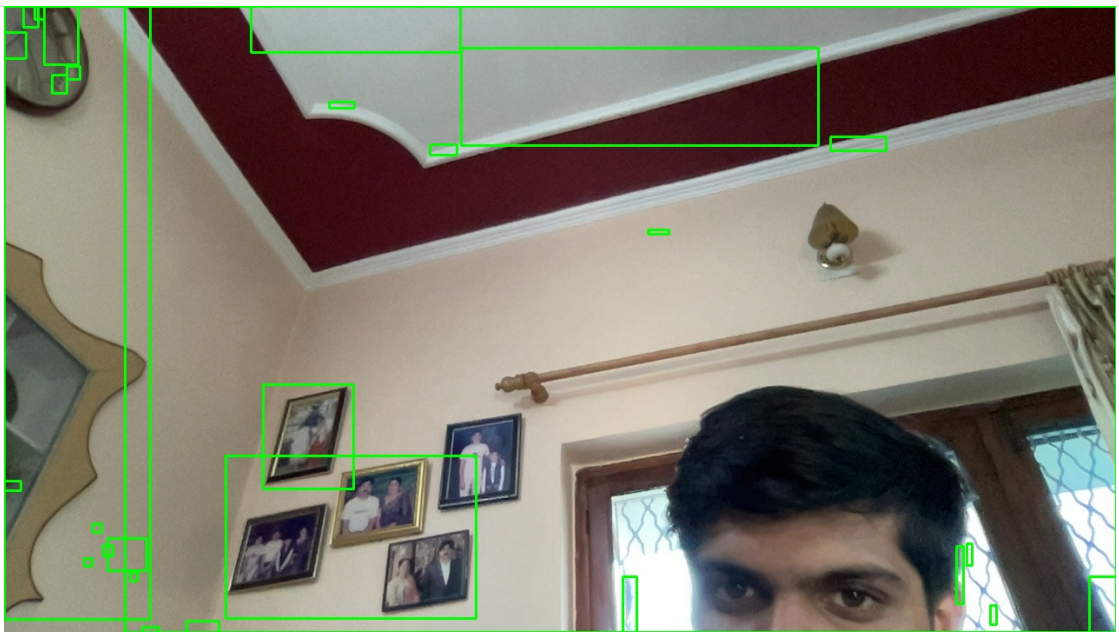
```
import cv2
from datetime import datetime

def record():
    cap = cv2.VideoCapture(0)

    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(f'recordings/{datetime.now().strftime("%H-
%M-%S")}.avi', fourcc,20.0,(640,480))

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

        cv2.putText(frame, f'{datetime.now().strftime("%D-%H-%M-
%S")}', (50,50), cv2.FONT_HERSHEY_COMPLEX,
                    0.6, (255,255,255), 2)

        out.write(frame)


        cv2.imshow("esc. to stop", frame)

        if cv2.waitKey(1) == 27:
            cap.release()
            cv2.destroyAllWindows()
            break
```

Chapter III
In Use

feature 1 - Monitor

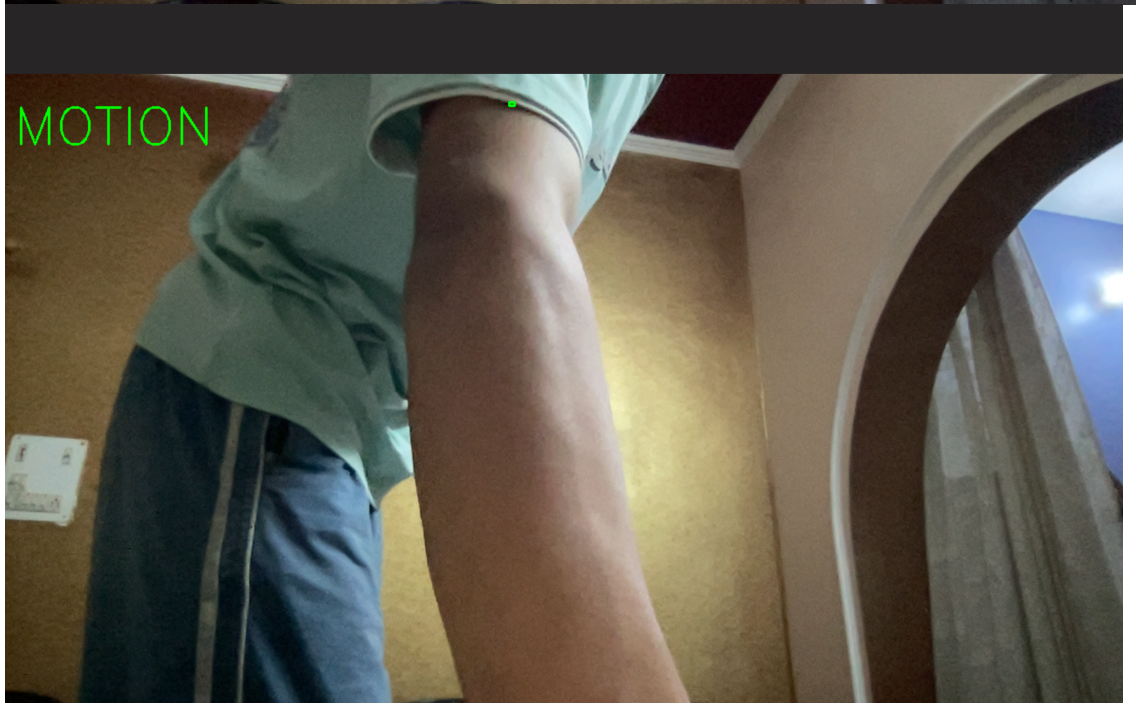
Showing use of first feature or you can consider it as output from feature 1.

as you can see that it is detecting the objects in the frame which is tr



#feature 2 – Noise Detection

This is working captured output for NO-Motion and Motion being detected by this application.

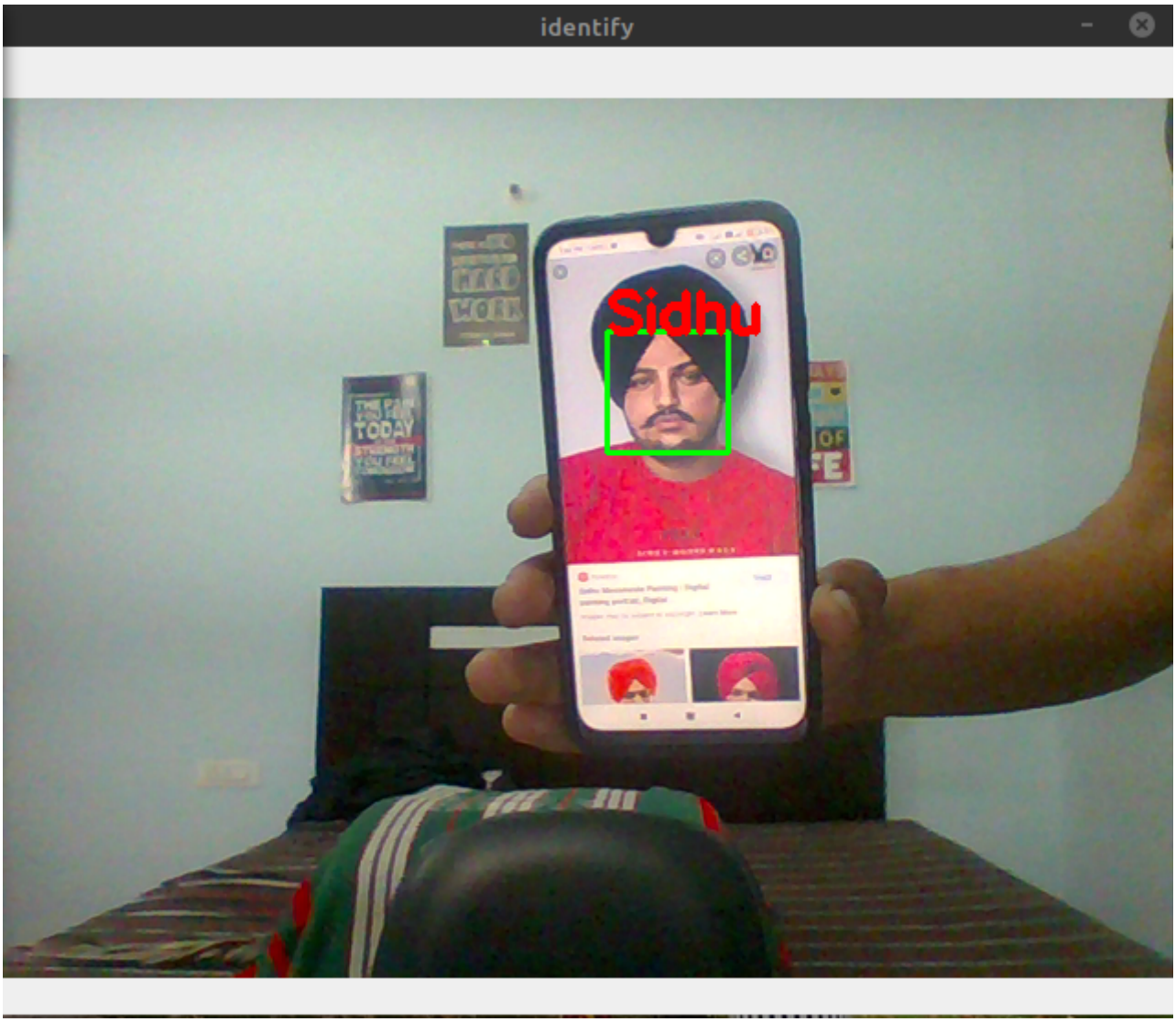


 # feature 3 – In out Detection

It has detected me entering in the room and being detected as entered and saving the image locally.

feature 4 – Face Identification

Since I have trained my model for Sidhu and it is predicting Sidhu correctly.
Now its not always predict right sometime it makes bad predictions also.



Refrences

For making this project we have used so many websites and papers and youtube tutorials all are below specified.

- waterfall model geeksforgeeks
- Structural Similarity from medium
- face detection
- LBPH algorithm
- openCV
- tech with tim

Also we have used so many other youtube channels and google and stack overflow to solve our errors.
Also we used Official python documentations to know basics about python.

Future Scopes

Based On the technology improvements such being having the capability of small size but high processing power this project can be broadly used. Below are some future workout on this project.

- Creating Portable cctv.
- Adding in-built night vision capability.
- Adding deep learning if having high power device.
- More feature such as
  - Deadly weapon detection
  - Accindent detection
  - Fire Detection
  - much more..
- Making a stand alone application with no requirements such as python, etc.
- Making standalone device.

Adding DL support would create broad scope in this project such as with DL we would be able to add up much more functionality.