

PROFESSIONAL TRAINING REPORT

At

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award

of Bachelor of Technology Degree in

Information Technology

By

RITHWIK.E

(Reg. No. 39120096)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119**

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI CHENNAI– 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **RITHWIK.E (Reg. No: 39120096)** who carried out the project entitled **“Smart CCTV Application”** under my supervision from June 2021 to November 2021.

Internal Guide

Dr. S. Gowri, M.E., Ph.D.

Head of the Department

Dr. R. Subashini, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **RITHWIK.E** hereby declare that the project report entitled **Smart CCTV Application** done by me under the guidance of **Dr. S. Gowri, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE: 13/11/2021

PLACE: Chennai

A handwritten signature in blue ink, appearing to be 'Rithwik.E', is written over a faint, circular official stamp.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

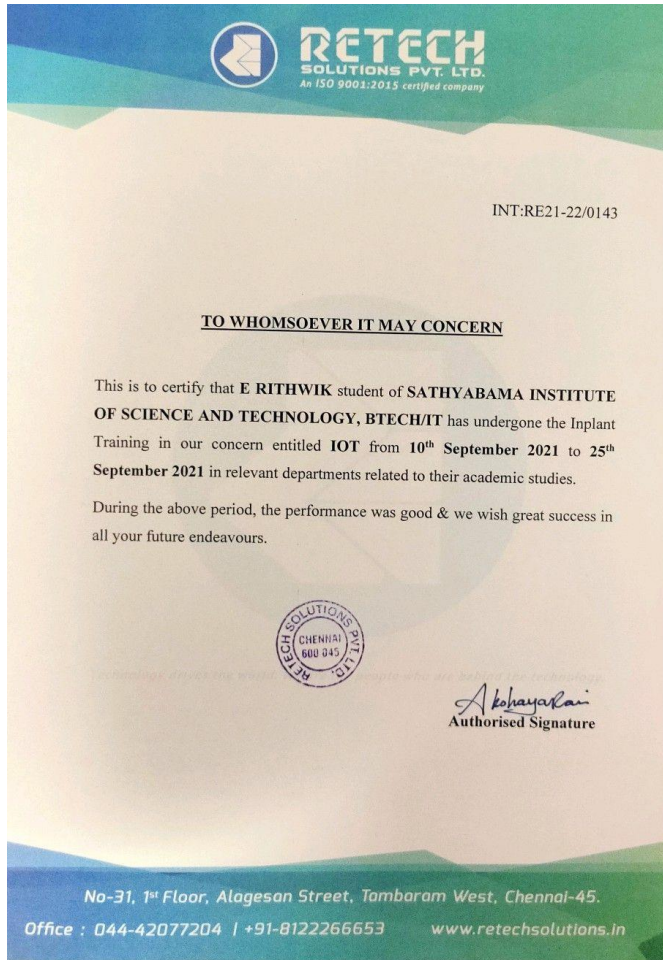
I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D. Dean**, School of Computing, **Dr. Dr. R. Subhashini, M.E., Ph.D. Heads of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. Gowri, M.E., Ph.D.** for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

Training Certification:



ABSTRACT

The field of CCTV surveillance is topical and widely used in many different applications. The fundamental part of the CCTV system is a reliable image evaluation by a human observer, whose effectiveness is influenced by many variables. Optimization of the effectiveness is a multidimensional problem related to both technical and human characteristics. In many applied systems, the overall performance is affected by a real performance of technical system (image compression, channel transmission, etc.). On the other hand, these technical systems have different optimization criteria than a typical video system. TNO Defense, Security and Safety (formerly the TNO Physics and Electronics Laboratory) initiated a project in which the influence of several technical variables on the operator's effectiveness was studied. The whole project was carried out in close co-operation with the Czech Technical University in Prague (CTU). The paper demonstrates and summarizes selected experimental results and evaluated dependencies.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
1	ABSTRACT	6
	LIST OF FIGURES	9
	LIST OF ABBREVIATIONS	10
1	INTRODUCTION	11
	1.1 Brief Intro	11
2	AIM AND SCOPE	12
	2.1 Aim	12
	2.2 Scope	12
	2.3 Advantage of this project	12
3	SYSTEM ANALYSIS AND DESIGN	13-14
	3.1 General	13
	3.2 User Interface (UI)	13
	3.3 Software's Used	13
	3.4 System Requirement	14
4	PROJECT DESCRIPTION	15-20
	4.1 General	15
	4.2 Modules Involved	15-20

5	RESULT AND DICUSSION, PERFORMANCE ANALYSIS	19
	5.1. Results	19
	5.2. Discussion	19
	5.3. Performance Analysis	19
6	SUMMARY AND CONCLUSION	20
	6.1. Summary	20
		20
	6.2. Conclusion	
	APPENDEX	21-39
	A. Screenshots	21-27
	B. Source Code	28-39

LIST OFFIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
Fig.4.1	Performance Analysis	19
Fig.A.1	Sneak peak	23
Fig.A.2	Structure	24
Fig.A.3	OpenCV	24
Fig.A.4	LBPH	24
Fig.A.5	Noise frame	24
Fig.A.6	Waterfall model	25
Fig.A.7	Monitoring	26
Fig.A.8	Noise detection	27
Fig.A.9	In-Out Detection	28

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CCTV	Closed-Circuit Television
OpenCV	Computer Vision
IoT	Internet of Things
IDLE & Learning Environment	Integrated Development

1. INTRODUCTION

1.1. BRIEF INTRO:

The field of CCTV surveillance is topical and widely used in many different applications such as a town center surveillance. The fundamental part of the CCTV system is a reliable image evaluation by a human observer, whose effectiveness (i.e., incident detection rate) is influenced by many variables. Optimization of the effectiveness is a multidimensional problem related to both technical and human characteristics. On the technical side, it is important that the modern technology is applied properly. Of the human characteristics, operator skill and motivation are decisive. Even if the system technical performance is excellent, the crucial element in the chain is the human observer. The quality and pre-processing of video information, manner of presentation and working environment strongly influence the operator's effectiveness.

CHAPTER 2

AIM AND SCOPE

2.1 AIM

The main aim of this project is movement detection and surveillance. This smart camera works when any movement is detected in a person

2.2 SCOPE

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

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

2.3. ADVANTAGES OVER THIS PROJECT

- Avoid internal and external theft.
- Crime prevention
- Evidence collection
- Staff security
- Private and sensitive area surveillance.
- Customer satisfaction endurance.
- Protection against false liability lawsuits.
- Curb sexual harassment.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 GENERAL

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.

3.2 USER INTERFACE (UI)

The user interface (UI) is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse, camera and the appearance of a desktop. It is also the way through which a user interacts with an application

3.3 SOFTWARES USED

3.3.1 CODE EDITORS USED

VISUAL STUDIO CODE

Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and MacOS. It includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.

3.3.2 PROGRAMMING LANGUAGES USED

SOFTWARE REQUIREMENTS

PYTHON 3

- Since this is a software hence it will have to run on some hardware and operating system obviously, 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.

3.4 SYSTEM REQUIREMENTS

Operating System: Windows / MacOS / Linux

RAM: 1GB (minimum)

Processor: 1GHz processor (minimum)

ROM: 20GB (minimum)

CHAPTER 4

PROJECT DESCRIPTION

4.1 GENERAL

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

Hosted Link:

4.2 MODULES INVOLVED

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, let's see each feature 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,

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

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.

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 it gives us out the masked image with score.

Identify the Family Member feature

This feature is a very useful feature of our minor project, it is used to find if the person in the frame is known or not. It does 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, let's divide this in following categories,

1 – Detecting faces in the frames

This is done via **Haar cascade** 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 Logit boost).

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.

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 parameter's **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 on when 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 continuously analyzed and checked for noises. Noise is checked in the consecutive frames. Simply we do the absolute difference between two frames and in this way the difference of two images is analyzed and Contours (boundaries of the motion are detected) and if there are no boundaries then no motion and if there is any there is motion.

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 values of brightness.

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 motion 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.

CHAPTER – 5.

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

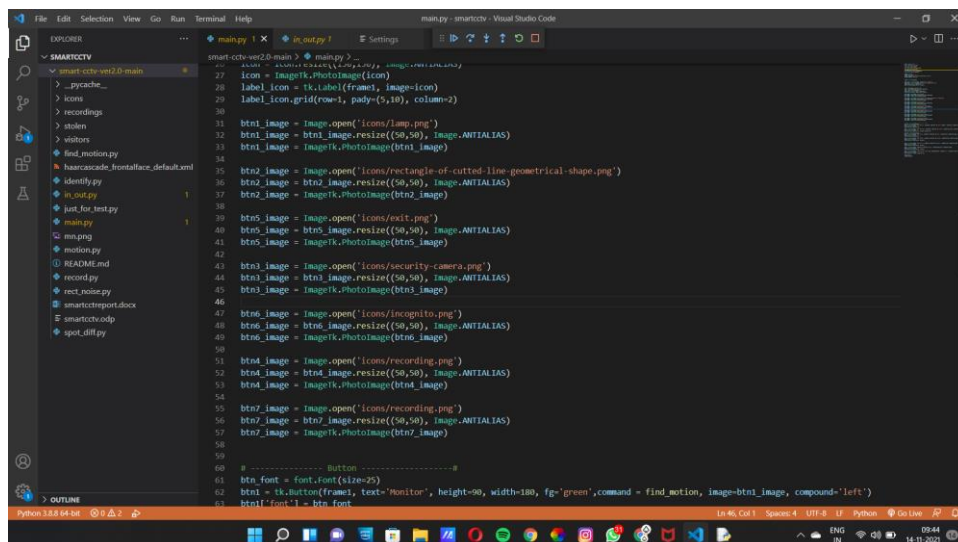
4.1. RESULTS: Using this Smart CCTV we Monitored face detection, motion detection, Noise detection, etc.

4.2. DISCUSSION:

We have discussed with our guide regarding the requirements for the project and when it comes to the work me and my friend who worked along with me. We both discussed together deciding the colors, icons, and images for the application

4.3. PERFORMANCE ANALYSIS:

Fig.4.1. Performance Analysis



When it comes to the Performance, runs perfectly

CHAPTER-6 SUMMARY AND CONCLUSIONS

5.1 SUMMARY:

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.

5.2 CONCLUSION:

The Project Smart CCTV camera Surveillance system is designed using wireless technology. This project is basically design for providing a security in different areas like military, banks, and industries etc and also save the power, memory required for the CCTV and CCTV footage respectively.

REFERENCES:

[1] <https://developer.mozilla.org/en-US/docs/Learn>

[2] <https://www.w3schools.com/whatis/>

APPENDIX

A. SCREENSHOTS:

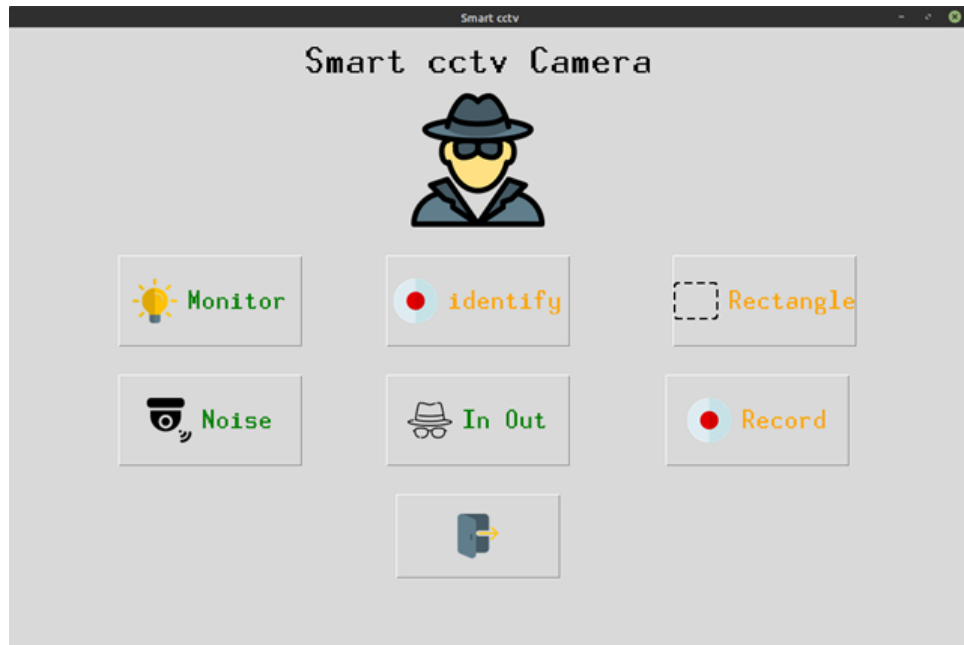


Fig A.1. Sneak peak

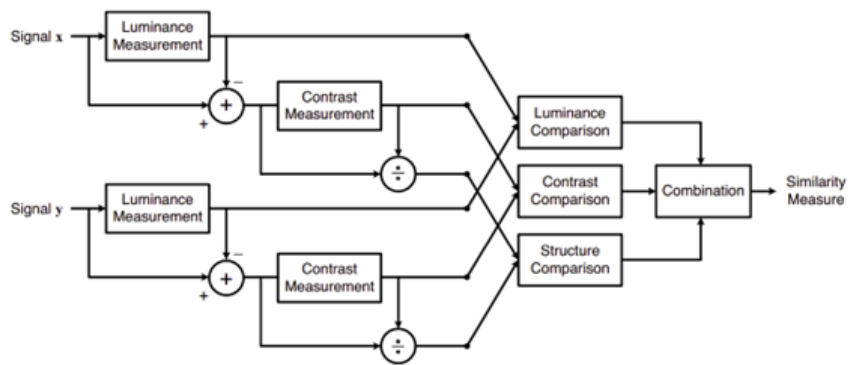


Fig A.2. Structure

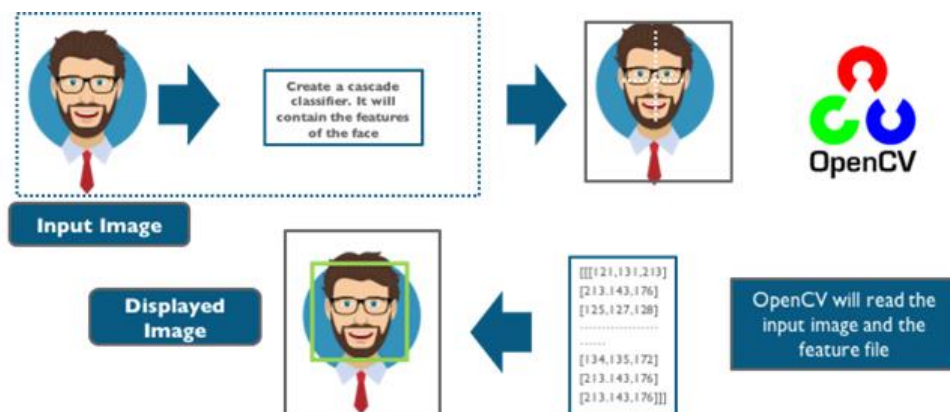


Fig A.3. OpenCV

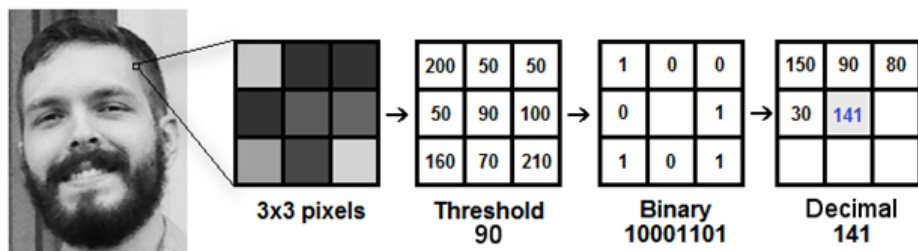


Fig A.4. LBPH

frame1	frame2	frame2 - frame1	abs (frame2 - frame1)
10 90 16 16	10 90 16 16	0 0 0 0	0 0 0 0
0 11 11 11	0 13 17 11	0 2 6 0	0 2 6 0
18 30 33 33	18 34 31 33	0 4 -2 0	0 4 2 0
18 18 18 18	18 17 19 18	0 -1 1 0	0 1 1 0

Fig. A.5. Noise Frame

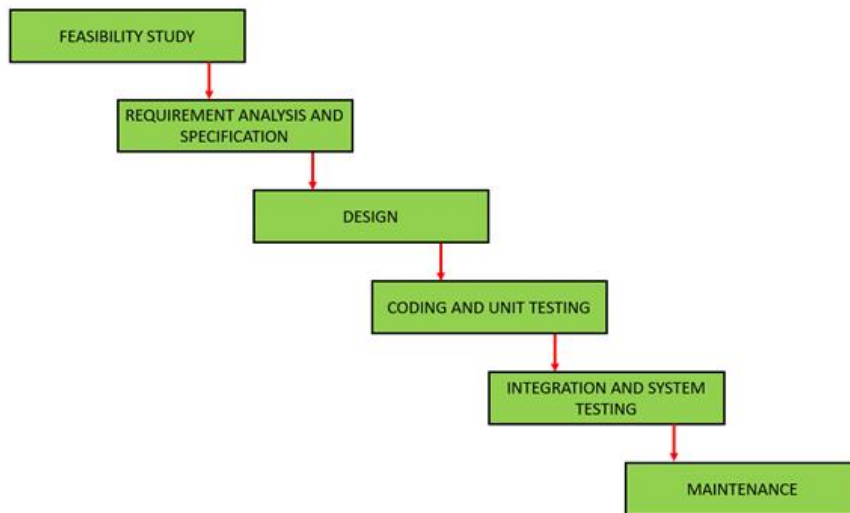


Fig. A.6. Waterfall model

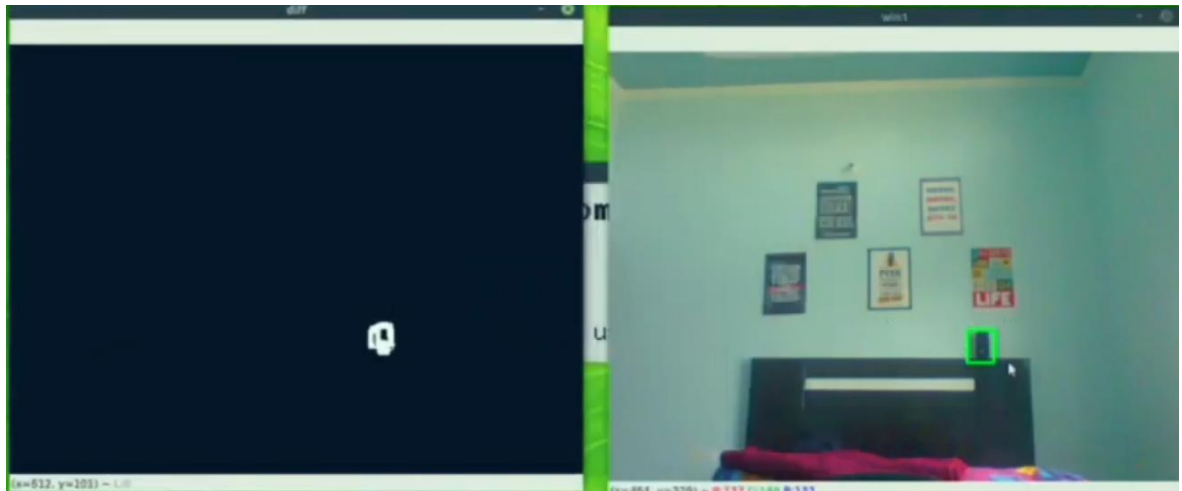


Fig. A.7. Monitoring

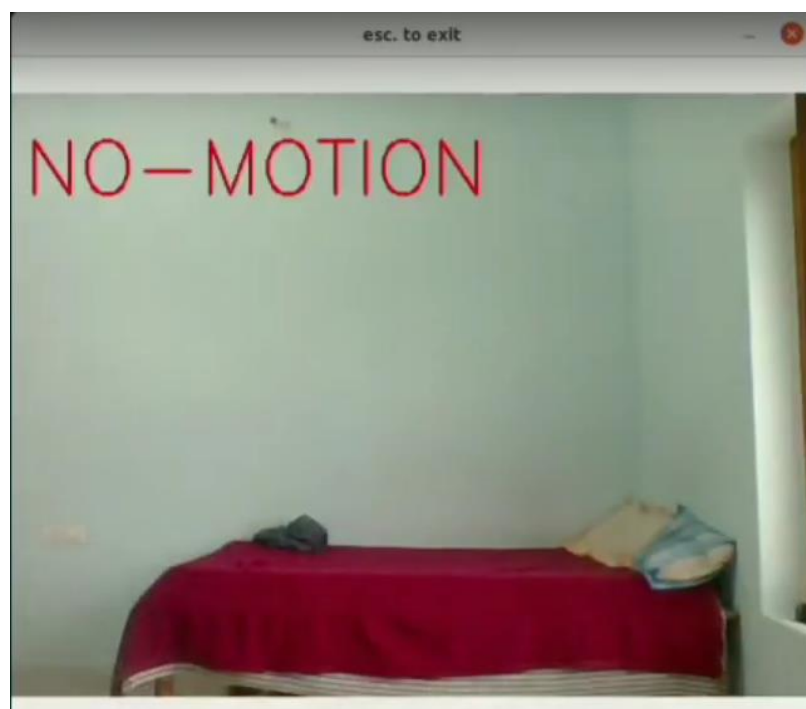
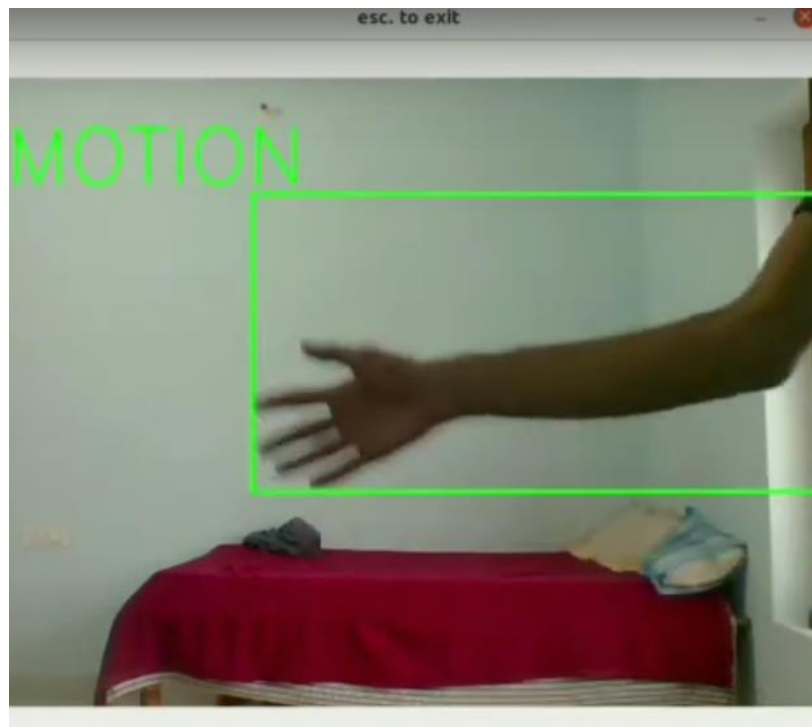


Fig A.8. Noise detection

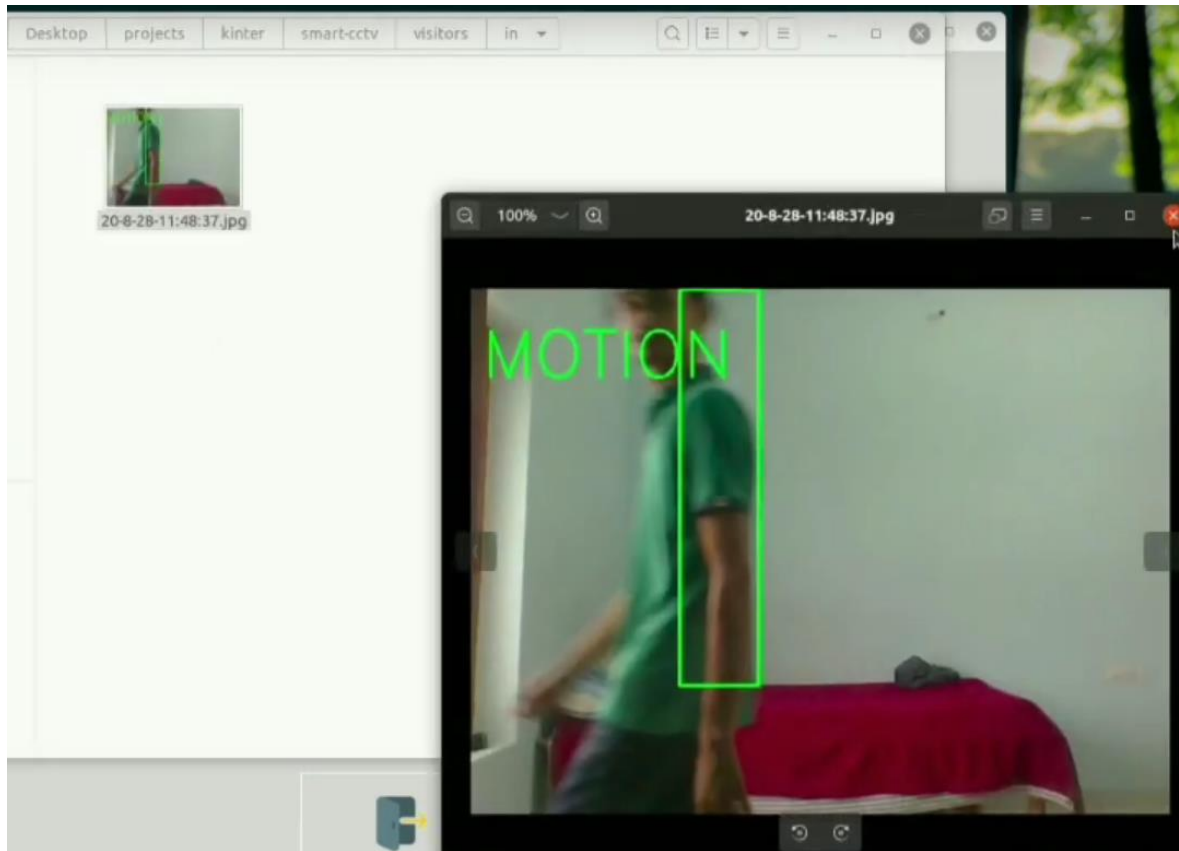


Fig A.9. In and out detection

B. SOURCE CODE:

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=main_call, 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

```

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)

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

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

        if len(contours) > 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(contours) < 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("running again")
return

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

else:
cv2.putText(thresh, "no motion detected", (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 2, 2 55)

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]

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

    if len(contours):
        for c in contours:
            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

```

```

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

```

```

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'{label[list[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(ro ot, 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-%d-%H:%M:%S')} .jpg", frame1)

            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()
```

•