MOVEMENT TRACKER

Computer Science and Engineering

By

Aakarsh Gupta 181232

Under the supervision of

Dr. Monika Bharti

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

DECLARATION

We on our behalf declare that the project that we made has been made by us under the direction of **(**Dr Monica Bharti) the Jaypee University of Information Technology. We also announce that neither this project nor any part of this project has been submitted elsewhere for any purpose.

Supervised to:
Dr. Monika Bharti
Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology

Submitted by:
Aakarsh Gupta 181232
Computer Science & Engineering Department
Jaypee University of Information Technology

CERTIFICATE

This is to certify that the project which is being represented in the project report named “Movement Tracker” in partial fufilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and represented to the Department of Computer Science and Engineerin(CSE), Jaypee University of Information Technology, Waknaghat is an authentic record of work done by Aakarsh Gupta (181232) during the period from January 2022 to May 2022 under the superviision of Dr Monica Bharti, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Aakarsh Gupta 181232

The above statement made is correct to the best of my knowledge.

Monika Bharti

Dr. Monika Bharti
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

ACKNOWLEDGEMENT

Firstly, I am gratefulness to Almighty God and thankful for his divine grace that made it possible to successfully complete the project work in a smooth way.

I am grateful and wish my profond indebtedness to supervisor Dr. Monika Bharti, Department of CSE Jaypee University of Information Technology, Waknaghat.
where she was very helpful and guided us throughout the thesis to carry out this project. Her endless scholarly guidance, patience, continual encouragment, const scholarly guidance and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude and thankful to to Dr. Monika Bharti**,** Department of CSE, for her considerate nature and kind help to finish my project on time.

I would also generously thank each of the individuals who have helped me continuously and straightforwardly or in a roundabout way to make this project a success. In this odd and strange situation, I also want to thank the various staff individuals, both educating and non-educating, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Aakarsh Gupta 181232

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

We've heard of many situations where we need to track at what time someone entered/left a particular place or to check if number of people entered a place is equal to the number of people left or to count the number of people in a static room at a particular time.

A complete track of movement can be very useful and have enormous applications.

Due to advancement in technology, everything is becoming automated and this thesis details the process on how the movement tracker was made using Python which can be used for solving theft, calculating real time population of a hall, take attendance and many more.

A camera can be placed at the entry of hall and this project will simply tell us at what particular time someone entered the hall and when he/she left

CHAPTER 01

INTRODUCTION

1.1 INTRODUCTION

Movement tracker is a camera application which is capable of not only detecting the movement by estimating the movement of interesting features in successive images of scene but also its time through which we can conclude many useful insights with the data collected.

1.2 PROBLEM STATEMENT

Inspired from crime serials on how the police team uses CCTV cameras to see if a person is entering a particular place and at what time he/she leaves. To make it simpler with the help of technology, we can automate it and make an application that only uses a camera and automatically tells the time when someone entered or left the place.
This tracking can also be used to find useful insights like what was the population of a particular hall at a given time and even by adding facial recognition, we can conclude that whether the person who entered the room left it or not.

1.3 OBJECTIVE

All in all the main aim to make this application is to make the best use of technology to not only save the time but also reduce the labour and help to solve the crime in efficient way, take attendance, calculate population, etc.

1.4 METHODOLOGY

Used a very popular algorithm named '**Lucas-Kanade algorithm**'
The Lucas-Kanade algorithm is commonly used to calculate Optical flow with a small feature set. The major idea of this method is basically based on the consistent assumptions of space, where the nearby pixels have the same departure path. This consideration helps us to find a balanced mathematical solution with two variables.

The Lucas-Kanade algorithm is an efficient method for obtaining optical flow information at interesting points in an image (i.e. those exhibiting enough intensity gradient information). It works for moderate object speeds.

The main advantage of the algorithm is, that for a neighbourhood of fixed size, the number of operations needed to compute $(S\ T\ S)\ {-1}S\ T \rightarrow t$ are constant, and therefore the complexity of the algorithm is linear in the number of pixels examined in the image. Alternative algorithms that match similar regions using a neighbourhood, and scanning the second image, have quadratic complexity.

The Lucas-Kanade algorithm makes a ”best guess” of the displacement of a neighborhood by looking at changes in pixel intensity which can be explained from the known intensity gradients of the image in that neighborhood. For a simple pixel we have two unknowns (u and v) and one equation (that is, the system is underdetermined). We need a neighborhood in order to get more equations. Doing so makes the system overdetermined and we have to find a least squares solution.

The Lucas-Kanade algorithm makes a ”best guess” of the displacement of a neighborhood by looking at changes in pixel intensity which can be explained from the known intensity gradients of the image in that neighborhood. For a simple pixel we have two unknowns (u and v) and one equation (that is, the system is underdetermined). We need a neighborhood in order to get more equations. Doing so makes the system overdetermined and we have to find a least squares solution.

It works by guessing in which direction an object has moved to conclude the local changes in intensity.

There are following steps for the feature tracker:
1. First compute the intensity for each pixel.
2. For each pixel position compute the gradient matrix and store an eigen value of matrix.
3. Store each pixel position in the score matrix S and separate the high scoring pixels by flag matrix F and region size k and flag region size f.
4. Take the top n eigen values and use those for the trackable features.

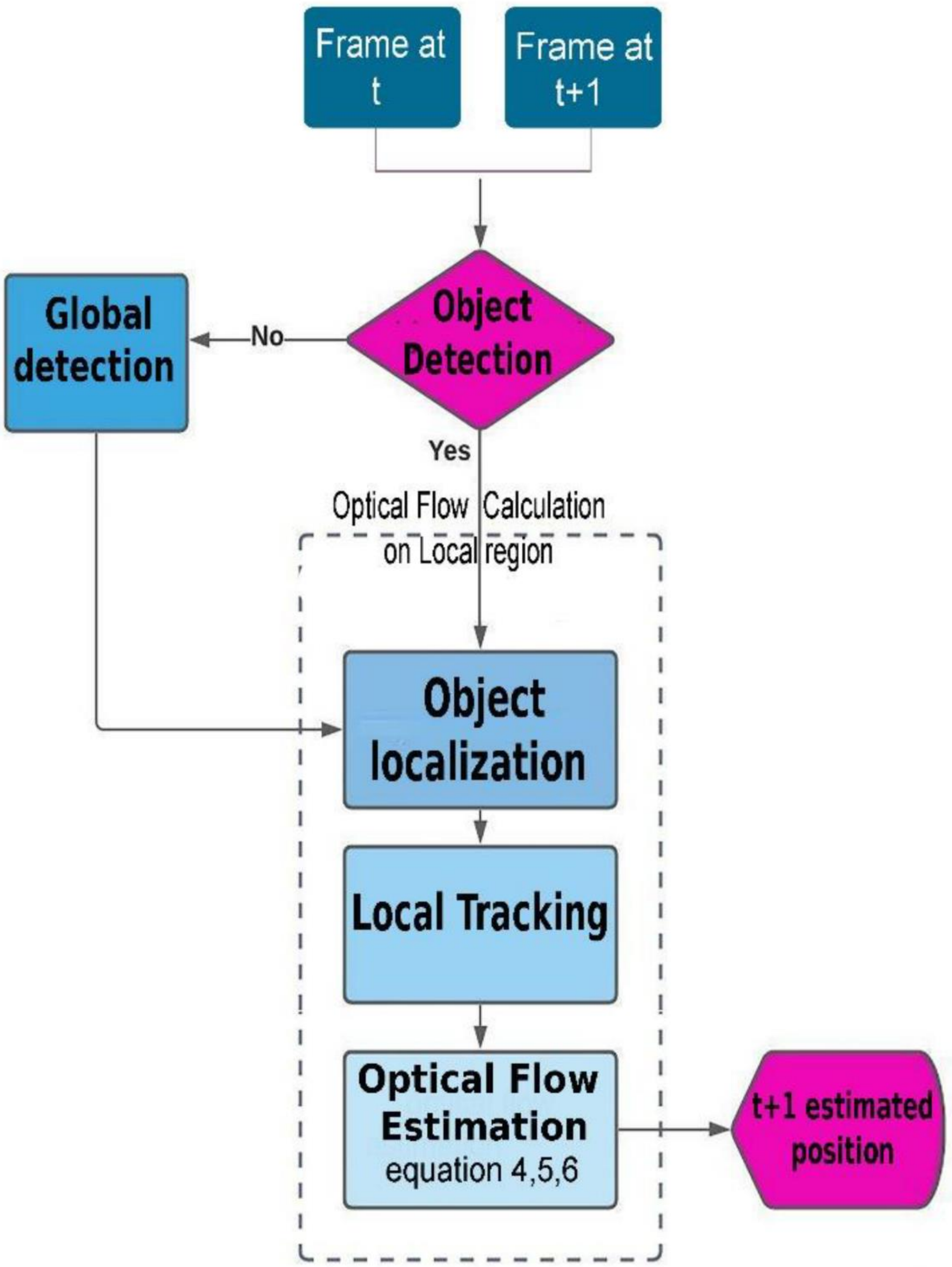


Fig 1.1 Lucas-Kanade Flow Chart

Theory:

Let's assume that the neighboring pixels have the same motion vector $(\Delta x, \Delta y)$. We can take a fixed-size window to create a system of equations. Let $p_i = (x_i, y_i)$ be the pixel coordinates in the chosen window of n elements . Hence, we can define the equation system as:

$$\begin{cases} I'_x(p_1)u + I'_y(p_1)v = -I'_t(p_1) \\ I'_x(p_2)u + I'_y(p_2)v = -I'_t(p_2) \\ \dots \\ I'_x(p_n)u + I'_y(p_n)v = -I'_t(p_n) \end{cases}$$

This equation system can be rewritten in matrix form:

$$A = \begin{bmatrix} I'_x(p_1) & I'_y(p_1) \\ I'_x(p_2) & I'_y(p_2) \\ \vdots & \vdots \\ I'_x(p_n) & I'_y(p_n) \end{bmatrix}, \ \gamma = \begin{bmatrix} dx \\ dy \end{bmatrix}, b = \begin{bmatrix} -I'_t(p_1)dt \\ -I'_t(p_2)dt \\ \vdots \\ -I'_t(p_n)dt \end{bmatrix}$$

As a result, we have a matrix equation: $A\gamma = b$. Using the Least Squares we can compute the answer vector γ :

$$A^TA\gamma = A^Tb \quad \Rightarrow \quad \underbrace{(A^TA)^{-1}(A^TA)}_{Identity\, matrix}\gamma = (A^TA)^{-1}A^Tb \quad \Rightarrow \quad \gamma = (A^TA)^{-1}A^Tb$$

1.5 ORGANIZATION

Chapter 1 contains the Introduction, Problem Statement, Objective, Methodology of the Project or System.

Chapter 2 contains the literature survey in which some of the previous works are studied and compared in order to make this model.

Chapter 3 discusses the system development as it contains all the computational, mathematical, analytical, theoretical data.

Chapter 4 is all about the performance analysis in which different models' results are compared, with algorithm improvement and outputs.

Chapter 5 discusses the conclusions, future scope of the model, its applications in real world, advantages and limitations.

Then we have the References section.

Last is the Appendices section which includes the source code of model.

CHAPTER 02

LITERATURE SURVEY

As per the information given in Chapter 1, the movement tracking of objects can have a lot of applications in real world. In this section, some of the previous works are discussed which were studied and compared in order to make this model.

There are many different ways to find out optical flow of objects out of which Gradient method is basic one. But due to its aperture problem, gradient method cannot give the complete optical flow fields solution. After survying , we got to know two different Differential techniques named Lucas - Kanade algorithm and Horn - Schunck algorithm and compared them based on their results.

Bhumika Gupta (2017) [1] the acquisition of the proposed object by a well-known computer technology associated with computer vision and image processing focused to find its objects or conditions for a particular category (e.g. people, flowers, animals) in digital photos and videos. There are various applications for acquisition of assets well researched which includes facial recognition, character recognition, and vehicle calculator. Object acquisition can be used for a variety of purposes including retrieval and surveillance. Different basic concepts were used in this study object discovery while using the OpenCV library of python 2.7, improves the efficiency and accuracy of the object discovery introduced

Kartik Umesh Sharma (2017) [2], proposed something the discovery system detects real-world objects be it in digital photography or video, where the object can for any category of objects namely people, cars, etc. to get something in a photo or video in the system it needs to have a few parts to complete visual function, they are a model database, a feature detector, hypothesiser and hypothesiser confirmation. This paper provides an overview of the variety techniques used to find an object, localize an object, separate object, remove features, appearance information, and more, in photos and videos. I ideas are drawn based on textbooks as well important issues are also identified as appropriate to the object adoption. Information about source and online codes data sets are provided to help the new researcher enter a place to find something. An idea about a possible solution of the discovery of a multi-class object object was also introduced. This the paper is suitable for beginner researchers on this domain.

Mukesh Tiwari (2017) [3] introduced object discovery again tracking is one of the most important areas of research for a common change in object movement and scene variability size, occlusions, appearance differences, and ego-motion as well light changes. In particular, choosing a feature is an important role in tracking an object. It has to do with a lot of real time applications such as car view, video surveillance and and so on. To win the issue of adoption, track related to the movement and appearance of an object. Most of this The algorithm focuses on the tracking algorithm to smooth it out video sequence. On the other hand, only a limited number of methods are used pre-existing information about object shape, color, texture and more. Tracking algorithm integrates The parameters mentioned above are also discussed analyzed in this study. The purpose of this paper is to analyze and update the previous way of looking at an object tracking and identification using video sequencing different categories. Also, point to a gap and suggest a new one how to improve video object tracking frame.

Rupesh Kumar Rout
[4] developed feedback based object detection algorithm. It adopts dual layer updating model to update the background and segment the foreground with an adaptive threshold method and object tracking is treated as a object matching algorithm.

CHAPTER 03

SYSTEM DEVELOPMENT

3.1 Functions/Methods used

Cv2.VideoCapture(0):
Cv2.flip(inp_img , 1):
cv2.blur(inp_img, (4,4)):
cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY):
Cv2.flip(inp_img , 1):
cv2.calcOpticalFlowPyrLK():
cv2.imshow()
datetime.datetime.now()
cv2.waitKey(1)

Cv2.VideoCapture(0):
Import cv2. Create variables to save video using the VideoCapture () function. Pass the 0 parameter in VideoCapture (0) to access the webcam. Create an endless loop to show each frame of a webcam video continuously.

Cv2.flip(inp_img , 1):
cv2.flip () method used to browse 2D lists. The cv :: flip function rotates a 2D round axis straight, horizontal, or both axes. We used this to take the input and flip so that in screen we can avoid the error caused by the mirror image when we use the camera as the input.

cv2.blur(inp_img, (4,4)):
cv2.blur () method is used to blur an image using a standard box filter. The function makes the image smoother using a kernel represented by:

$$K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \dots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY):
The main work of the function in to convert a color space of image into the required one . There are over 150 color conversion options available in OpenCV.

Syntax

cv2.cvtColor(img, co[,oi[]]):

- img: The required image that you want to change the color space of.
- Co : It represent the code in which we want to change the color space to.
- Oi   : It represent the output image that we will get after changing the color space of the image

Cv2.flip(inp_img , 1):
cv2.flip () method used to browse 2D lists. The cv :: flip function rotates a 2D round axis straight, horizontal, or both axes. We used this to take the input and flip so that in screen we can avoid the error caused by the mirror image when we use the camera as the input.

cv2.calcOpticalFlowPyrLK():
Visual flow is the function of measuring the movement of each pixel between two consecutive frames in a single video. Basically, the Optical Flow function refers to the pixel shift vector calculation as the difference of object removal between two neighbouring images.

cv2.imshow()
OpenCV-Python is an integrated Python library designed to solve computer vision problems. cv2.imshow () method which used to display an image in the current screen. The window automatically enters image size.

We use this method to show the output screen of the both datetime and the real time motion detection

datetime.datetime.now()
We use this function to take the real time input from the system in order to get the exact output that we need to get for the project.

Cv2.flip(inp_img , 1):
cv2.flip () method used to browse 2D lists. The cv :: flip function rotates a 2D round axis straight, horizontal, or both axes. We used this to take the input and flip so that in screen we can avoid the error caused by the mirror image when we use the camera as the input.

cv2.waitKey(1)
waitkey () Python OpenCV function allows users to display a given millisecond window or until any key is pressed.

3.2 Theoretical and Mathematical development

3.2.1 Background Subtraction:
Background subtraction is one of the important computer vision applications. In our project we use this method to find out the number of people crossing through a particular area.

Working of Background Subtraction:
However, in determining the movement, we often do the following considerations:

The background of our video playback has stopped and does not change in consecutive video frames. So, if we can model the background, we monitor it to make major changes. If there is a major change, we can get it - this change is usually accompanied by movement in our video.

It is now clear that in the real world, lumbering elephants are exposed by the aggression of speeding midgets. Because of the shade, light, light conditions, and any other changes that may be taking place in the area, our background may look very different from the various video frames. And if the background looks different, it may discard our algorithms. This is why the most effective removal / retrieval systems use fixed, fixed cameras and in controlled lighting conditions.

3.2.2 Motion estimation

It the the prcess of determing motion vectors tha dscribe the transformation from one 2-D img to another.

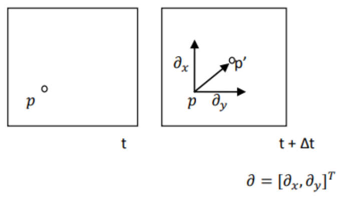
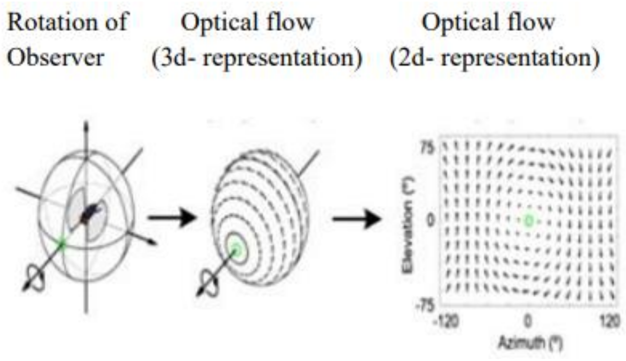


Fig 3.1 2-D image of the pixel location

Optic flow is the pattern of apparent motion of objects, edges and surface in a visual scene caused by the relative motion between an observer and the scene.



Rotation of Observer
Optical flow (3d- representation)
Optical flow (2d- representation)
75
0
-75
Elevation (°)
-120
0
120
Azimuth (°)

Fig 3.2 The optical flow experienced by a rotating observer

Motion field is the real world 3-D motion and Optical Flow Field is the projection of the motion field onto the 2D img.

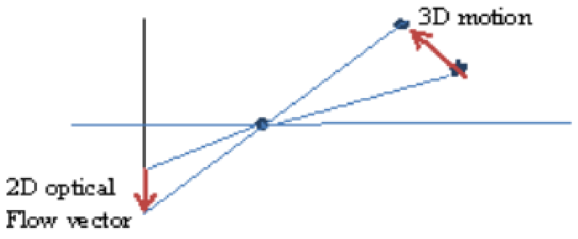


Fig 3.3 The relation between motion field & optical flow field

3.2.3 Mathematical Computation

The LucasKanade method assumes that the displacement of the image contents between two nearby frames are small and approximately constant within a neighborhood of the point p

$$I_x(p_1)V_x + I_y(p_1)V_y = -I_t(p_1)$$
$$I_x(p_2)V_x + I_y(p_2)V_y = -I_t(p_2)$$
$$\vdots$$
$$I_x(p_n)V_x + I_y(p_n)V_y = -I_t(p_n)$$

Where P1,P2…Pn are pixels inside the window, are the partial derivatives of the image I with respect to position x, y and time t, evaluated at the point Pn and at the current time.

The equations can be written in matrix form:

$$Av = b, \qquad where$$

$$A=\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ . & . \\ . & . \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, v=\begin{bmatrix} V_x \\ V_y \end{bmatrix}, b=\begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ . \\ . \\ -I_x(p_n) \end{bmatrix}$$

Now , we can write this equation in another form:

$$V = (A^T A)^{-1} A^T b$$

So,

$$\begin{bmatrix} Vx \\ Vy \end{bmatrix} = \begin{bmatrix} \sum_i I_x(p_n)^2 & \sum_i I_x(p_n) I_y(p_n) \\ \sum_i I_x(p_n) I_y(p_n) & \sum_i I_y(p_n)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_n) I_t(p_n) \\ -\sum_i I_y(p_n) I_t(p_n) \end{bmatrix}$$

Here, n=1 to N.

$A^T WAV = A^T Wb$, We can write this equation in another form so we get:

$$V = (A^T WA)^{-1} A^T Wb$$

Where W is an $n \times n$ diagonal matrix containing the weights Wii = wi to be assigned to the equation of pixel p_n . That is, it computes: $\begin{bmatrix} Vx \\ Vy \end{bmatrix} =$

$$\begin{bmatrix} \sum_i w_i \, I_x(p_n)^{\,2} & \sum_i w_i \, I_x(p_n) I_y(p_n) \\ \sum_i w_i \, I_x(p_n) I_y(p_n) & \sum_i w_i I_y \, (p_n)^{\,2} \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(p_n) I_t(p_n) \\ -\sum_i w_i I_y(p_n) I_t(p_n) \end{bmatrix}$$

The weight wi is usually set to a Gaussian function of the distance between p_n and p.

Lucas Kanade Algorithm

The Lucas-Kanade optical flow algorithm is a simple technique which can provide an estimate of the movement of interesting features in successive images of a scene. We would like to associate a movement vector (u, v) to every such ”interesting” pixel in the scene, obtained by comparing the two consecutive images. The Lucas-Kanade algorithm makes some implicit assumptions: – The two images are separated by a small time increment Δt, in such a way that objects have not displaced significantly (that is, the algorithm works best with slow moving objects). – The images depict a natural scene containing textured objects exhibiting shades of gray (different intensity levels) which change smoothly. The algorithm does not use color information in an explicit way. It does not scan the second image looking for a match for a given pixel. It works by trying to guess in which direction an object has moved so that local changes in intensity can be explained.

Technique:

Assume that we watch a scene through a square hole. The intensity a visible through the hole is variable.

In the next frame the intensity of the pixel has increased to b. It would be sensible to assume that a displacement of the underlying object to the left and up has occurred so that the new intensity b is now visible under the square hole.

If we know that the increase in brightness per pixel at pixel (x, y) is Ix(x, y) is the x-direction, and the increase in brightness per pixel in the y direction is Iy(x, y), we have a total increase in brightness, after a movement by u pixels in the x direction and v pixels in the y direction of:

$Ix(x, y) \cdot u + Iy(x, y) \cdot v$

This matches the local difference in intensity (b − a) which we call It(x, y),

so that $Ix(x, y) \cdot u + Iy(x, y) \cdot v = -It(x, y)$

The negative sign is necessary because for positive Ix, Iy, and It we have a movement to the left and down

Neighbourhoods:

Of course, a simple pixel does not usually contain enough ”structure” useful for matching with another pixel. It is better to use a neighborhood of pixels, for example the 3 × 3 neighborhood around the pixel (x, y). In that case we set 9 linear equations:

$Ix(x + \Delta x, y + \Delta y) \cdot u + Iy(x + \Delta x, y + \Delta y) \cdot v = -It(x + \Delta x, y + \Delta y)$

for $\Delta x = -1, 0, 1$ and $\Delta y = -1, 0, 1$

The linear equations can be summarized as the matrix equality

$Ix(x + \Delta x, y + \Delta y), Iy(x + \Delta x, y + \Delta y)$

and t is a vector containing the 9 terms $-It(x+ \Delta x, y + \Delta y)$.

The above equation cannot be solved exactly (in the general case). The Least Squares solution is found by multiplying the equation by S T S T S u v ! = S T → t and inverting S T S, so that u v ! = (S T S) −1S T → t

Conclusions:

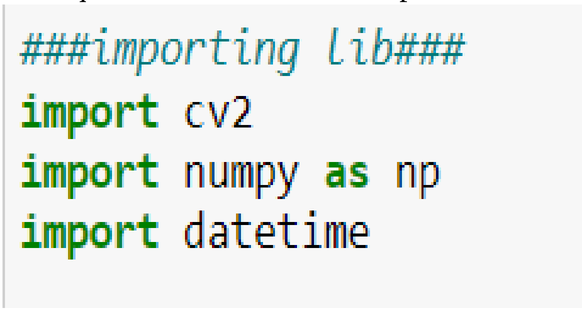
The Lucas-Kanade algorithm makes a ”best guess” of the displacement of a neighborhood by looking at changes in pixel intensity which can be explained from the known intensity gradients of the image in that neighborhood. For a simple pixel we have two unknowns (u and v) and one equation (that is, the system is underdetermined). We need a neighborhood in order to get more equations. Doing so makes the system overdetermined and we have to find a least squares solution. The LSQ solution averages the optical flow guesses over a neighborhood.

We assume that all intensity changes can be explained by intensity gradients. The method breaks down when the gradients are random (think of an image of random points) or when the gradients are negligible (no structure, as in flat surfaces). The Lucas-Kanade algorithm eliminates regions without structure by looking at the invertibilily of the matrix S T S in an indirect way, that is, through the eigenvalues of this matrix. The result of the algorithm is a set of optical flow vectors distributed over the image which give an estimation idea of the movement of objects in the scene. Of course, some optical flow vectors will be erroneous. The main advantage of the algorithm is, that for a neighborhood of fixed size, the number of operations needed to compute (S T S) −1S T → t are constant, and therefore the complexity of the algorithm is linear in the number of pixels examined in the image. Alternative algorithms that match similar regions using a neighborhood, and scanning the second image, have quadratic complexity. Summarizing: The Lucas-Kanade algorithm is an efficient method for obtaining optical flow information at interesting points in an image (i.e. those exhibiting enough intensity gradient information). It works for moderate object speeds.

3.3 Computational Development

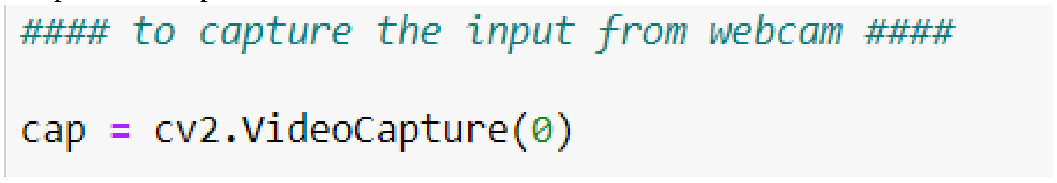
3.3.1 Code Breakage

- We will import all the required libraries for the implementation of the project.



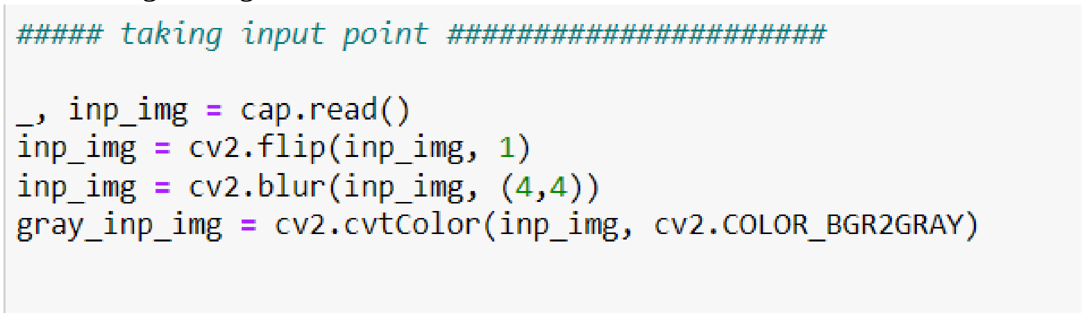
```
###importing lib###
import cv2
import numpy as np
import datetime
```

- To capture the input from our web cam we will use this function.



```
#### to capture the input from webcam ####

cap = cv2.VideoCapture(0)
```

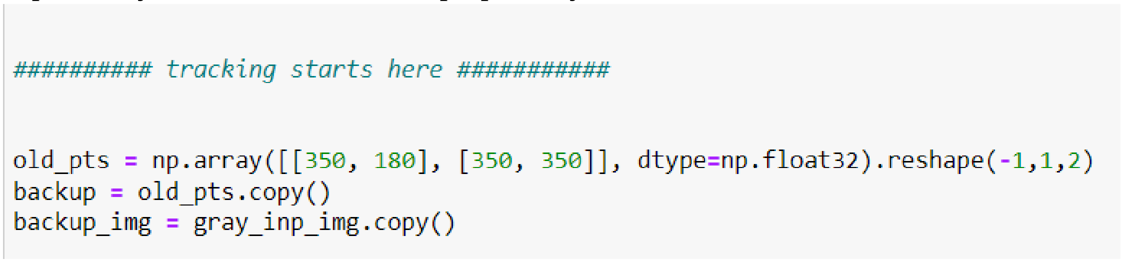
- To take the input point which is one of the main part of the project we use this logic in order to get things done.



```
##### taking input point ######################

_, inp_img = cap.read()
inp_img = cv2.flip(inp_img, 1)
inp_img = cv2.blur(inp_img, (4,4))
gray_inp_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)
```

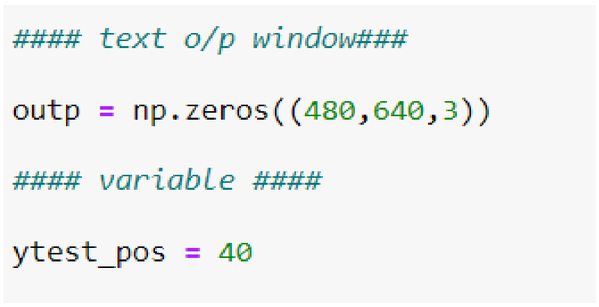
The read function makes array of the image and then we flip the image so that there is no problem of mirror image and we blur the image so that it get soft and all the extra information will not affect the final output of the project and in order to change the image colour scale we used cvtcolor function to change the input into grayscale so that we can implement our logic into the image without any problem.

- We build the logic in order to track any movement in the video which is being captured by the web cam of our laptop or any other devices.



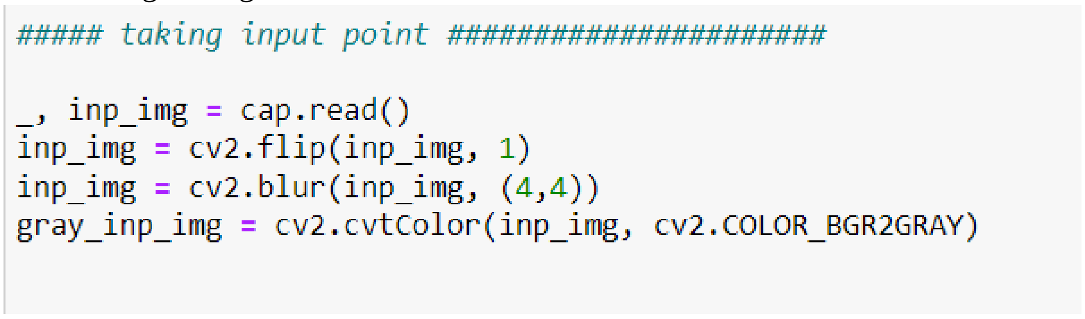
```
########## tracking starts here ###########


old_pts = np.array([[350, 180], [350, 350]], dtype=np.float32).reshape(-1,1,2)
backup = old_pts.copy()
backup_img = gray_inp_img.copy()
```

- After completing this we made a text output window that will help us to track the progress of our project in real time and will help us to find any hidden bug in our project.



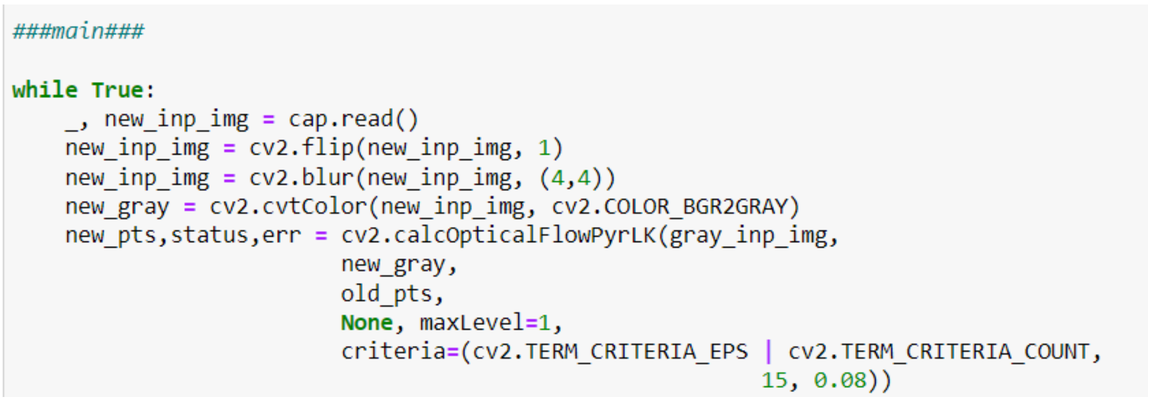
```
#### text o/p window###

outp = np.zeros((480,640,3))

#### variable ####

ytest_pos = 40
```

- To take the input point which is one of the main part of the project we use this logic in order to get things done.



```
##### taking input point ######################

_, inp_img = cap.read()
inp_img = cv2.flip(inp_img, 1)
inp_img = cv2.blur(inp_img, (4,4))
gray_inp_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)
```

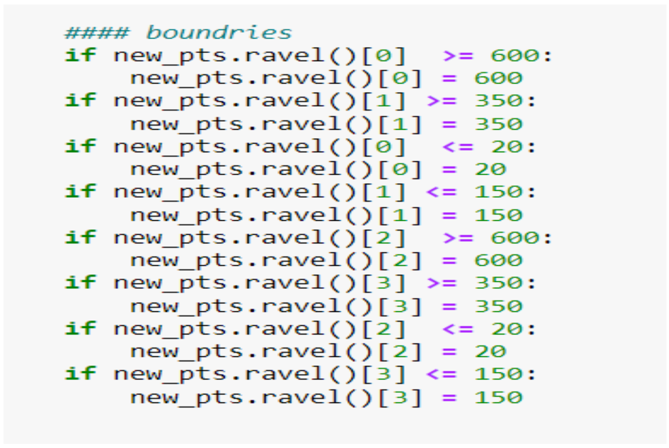
- The main function helps us to use the Lucas-Kanade algo and by putting in the desired and pre processed input to the algo we get the positive result from our project.



```
###main###

while True:
    _, new_inp_img = cap.read()
    new_inp_img = cv2.flip(new_inp_img, 1)
    new_inp_img = cv2.blur(new_inp_img, (4,4))
    new_gray = cv2.cvtColor(new_inp_img, cv2.COLOR_BGR2GRAY)
    new_pts,status,err = cv2.calcOpticalFlowPyrLK(gray_inp_img,
                         new_gray,
                         old_pts,
                         None, maxLevel=1,
                         criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
                                                         15, 0.08))
```

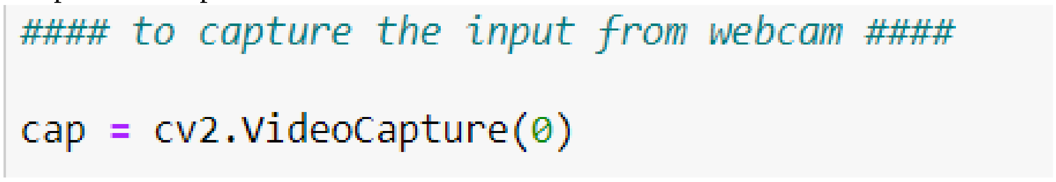
We used while true because we want to add an endless loop so that we can implement the algo in real time.

- In order to make the boundaries so that our project will not get a lot of garbage data and this boundary will help us to make our output look cleaner and easier to understand.



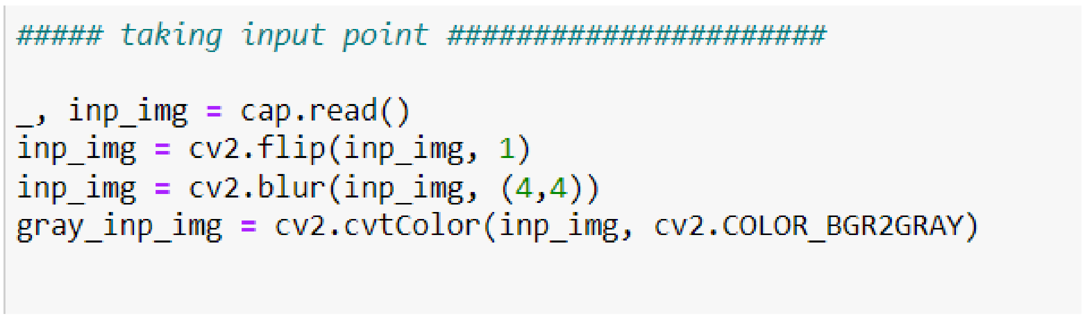
```
#### boundries
if new_pts.ravel()[0]  >= 600:
    new_pts.ravel()[0] = 600
if new_pts.ravel()[1] >= 350:
    new_pts.ravel()[1] = 350
if new_pts.ravel()[0]  <= 20:
    new_pts.ravel()[0] = 20
if new_pts.ravel()[1] <= 150:
    new_pts.ravel()[1] = 150
if new_pts.ravel()[2]  >= 600:
    new_pts.ravel()[2] = 600
if new_pts.ravel()[3] >= 350:
    new_pts.ravel()[3] = 350
if new_pts.ravel()[2]  <= 20:
    new_pts.ravel()[2] = 20
if new_pts.ravel()[3] <= 150:
    new_pts.ravel()[3] = 150
```

- To capture the input from our web cam we will use this function.



```
#### to capture the input from webcam ####

cap = cv2.VideoCapture(0)
```

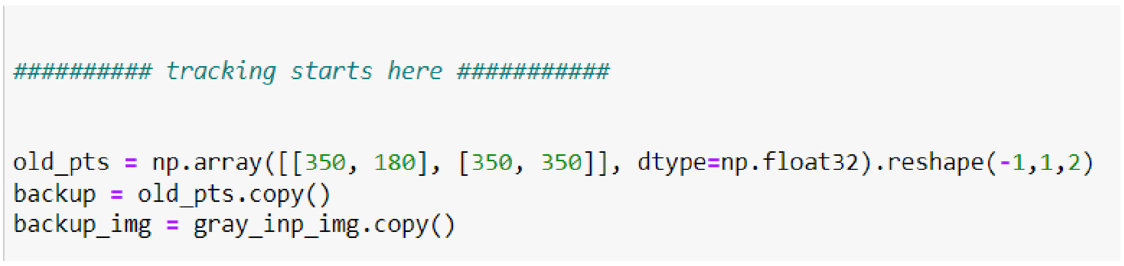
- To take the input point which is one of the main part of the project we use this logic in order to get things done.



```
##### taking input point #######################

_, inp_img = cap.read()
inp_img = cv2.flip(inp_img, 1)
inp_img = cv2.blur(inp_img, (4,4))
gray_inp_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)
```

The read function makes array of the image and then we flip the image so that there is no problem of mirror image and we blur the image so that it get soft and all the extra information will not affect the final output of the project and in order to change the image colour scale we used cvtcolor function to change the input into grayscale so that we can implement our logic into the image without any problem.

- We build the logic in order to track any movement in the video which is being captured by the web cam of our laptop or any other devices.



```
########## tracking starts here ###########

old_pts = np.array([[350, 180], [350, 350]], dtype=np.float32).reshape(-1,1,2)
backup = old_pts.copy()
backup_img = gray_inp_img.copy()
```

```
if new_pts.ravel()[0]  > 400 or new_pts.ravel()[2]  > 400:
    if new_pts.ravel()[0] > 550 or new_pts.ravel()[2]  > 550:
        new_pts = backup.copy()
        new_inp_img = backup_img.copy()
        ytest_pos += 40
        cv2.putText(outp, "gone at {}".format(datetime.datetime.now().strftime("%H:%M")), (10,ytest_pos),
            cv2.FONT_HERSHEY_PLAIN, 3, (0,255,0))




elif new_pts.ravel()[0]  < 200 or new_pts.ravel()[2]  < 200:
    if new_pts.ravel()[0] < 50 or new_pts.ravel()[2]  < 50:
        new_pts = backup.copy()
        new_inp_img = backup_img.copy()
        ytest_pos += 40
        cv2.putText(outp, "came at {}".format(datetime.datetime.now().strftime("%H:%M")), (10,ytest_pos),
            cv2.FONT_HERSHEY_PLAIN, 3, (0,0,255))
```

We in this place also used background subtraction in order to find the position of the person and find weather the person is coming from left to right or the person is coming from right to left. We used the combination of if else statement along with the use of date time library of python to record the exact time at which a person came or left the room.

- The final output in which we are able to keep the track of the persons motion and able to show the implementation in the real time along with the output of the exact time and the type of motion.

```
cv2.imshow('final', outp)
gray_inp_img = new_gray.copy()
old_pts = new_pts.reshape(-1,1,2)

if cv2.waitKey(1) & 0xff == 27:
    break# Data Augumentation
```

- This logic we used to make a red line in the middle of the screen that will act as the centre point and any object that will pass the line our program will keep the track of that this line help us to ignore the unnecessary data that will be captured by the webcam of the system.

```
##### drawing line ####
x,y = new_pts[0,:,:].ravel()
a,b = new_pts[1,:,:].ravel()
cv2.line(new_inp_img, (int(x),int(y)), (int(a),int(b)), (0,0,255), 15)


cv2.imshow("ouput", new_inp_img)
```

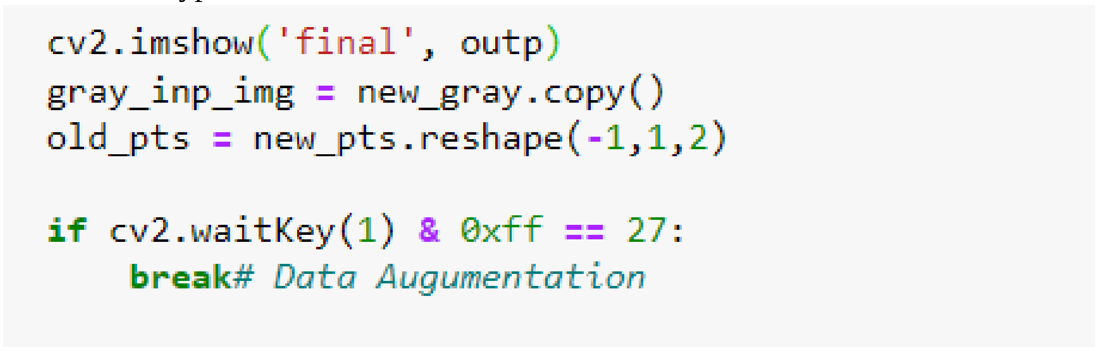
The imshow() function is used to show the output of the program.

- This is one of the main features of our project we made logic such that our program will keep the track of the object while it is coming to a particular room or that object is going out of the room.

```
if new_pts.ravel()[0]  > 400 or new_pts.ravel()[2]  > 400:
    if new_pts.ravel()[0] > 550 or new_pts.ravel()[2]  > 550:
        new_pts = backup.copy()
        new_inp_img = backup_img.copy()
        ytest_pos += 40
        cv2.putText(outp, "gone at {}".format(datetime.datetime.now().strftime("%H:%M")), (10,ytest_pos),
            cv2.FONT_HERSHEY_PLAIN, 3, (0,255,0))



elif new_pts.ravel()[0]  < 200 or new_pts.ravel()[2]  < 200:
    if new_pts.ravel()[0] < 50 or new_pts.ravel()[2]  < 50:
        new_pts = backup.copy()
        new_inp_img = backup_img.copy()
        ytest_pos += 40
        cv2.putText(outp, "came at {}".format(datetime.datetime.now().strftime("%H:%M")), (10,ytest_pos),
            cv2.FONT_HERSHEY_PLAIN, 3, (0,0,255))
```

We in this place also used background subtraction in order to find the position of the person and find weather the person is coming from left to right or the person is coming from right to left. We used the combination of if else statement along with the use of date time library of python to record the exact time at which a person came or left the room.

- The final output in which we are able to keep the track of the persons motion and able to show the implementation in the real time along with the output of the exact time and the type of motion.



```
cv2.imshow('final', outp)
gray_inp_img = new_gray.copy()
old_pts = new_pts.reshape(-1,1,2)

if cv2.waitKey(1) & 0xff == 27:
    break# Data Augumentation
```

3.3.2 Tools and Technologies used

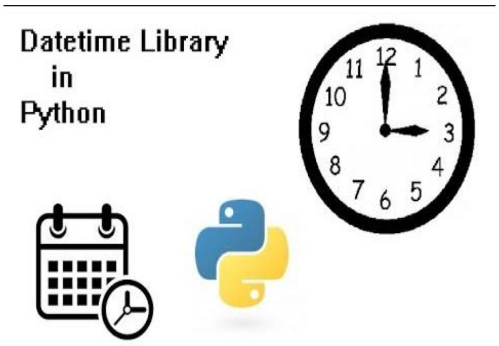
We used following tools and technologies to make this model:

- Python programming language
- OpenCV in Python
- NumPy in Python
- Datetime in Python
- Jupyter Notebook
- Sublime Text
- A Webcam

python™

OpenCV

NumPy



jupyter

Sublime Text

Python:

Python is a robust programming language with a wide range of capabilities.
Its broad features make working with targeted programs (including meta-programming and meta-objects) simple, and it fully supports object-oriented programming. Many additional paradigms, such as contract generation and logic programming, are supposed through extensions. Python takes advantage of power typing as well as the integration of reference computation and waste management waste collecting. lt also supports advanced word processing (late binding), which binds the way the words change during the process.

Patches to less essential sections of C Python that can give a minor improvement in performance at an obvious price are rejected by Python developers who try to prevent premature execution. When speed is crucial, the Python program developer can use mod-written modules in C-languages or PyPy, a timely compiler, to submit time-sensitive jobs. Python is a Python interpreter that converts Python scripts to C and invokes the Python interpreter directly from the C-1evel API. Python developers strive to make the language enjoyable to use. Python's architecture supports Lisp culture in terms of functionality. Filters, maps, and job reduction, as well as a list comprehension, dictionaries, sets, and generator expressions, are all included.

Two modules (itertools and functools) in the standard library use realistic Haskell and Standard ML tools.

Why Python?

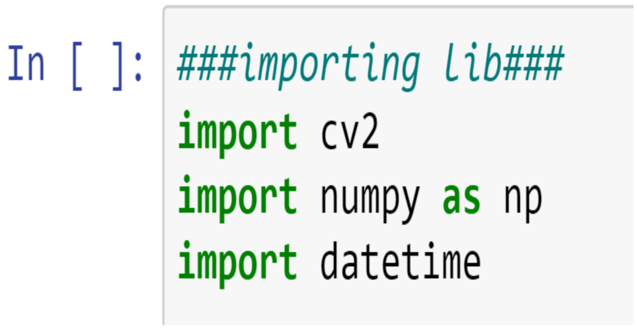
We used the python language since it is the only language which can be used on the maximum platforms that we can work on like Win, Linux, Mac, etc. Itis a free language having all the libraries for free . It is easy to use and is just like the common language that we use to communicate.

Python supports library collections and has a straightforward language structure like English while java and C ++ have complex codes. Python programs have fewer lines than other programming languages. That is why we use Python language to know human-made, AI, and deal with a large amount of knowledge. Python is the default language for an article.

Libraries Used:

Open CV	Numpy	DateTime

To import the required libraries, we’ll first use the keyword import and then library name.



```
In [ ]: ###importing lib###
        import cv2
        import numpy as np
        import datetime
```

Computer Visualization:
Computerized visualization can be considered as a process in which we will understend how imanges and videoos can be stored and the way we are able to operate and retrieve data from them. Computer Vision was that the basic or most generally used tool in computing. Computer-Vision plays an significant role that is used in self-driving cars, robots and photo editing programs.

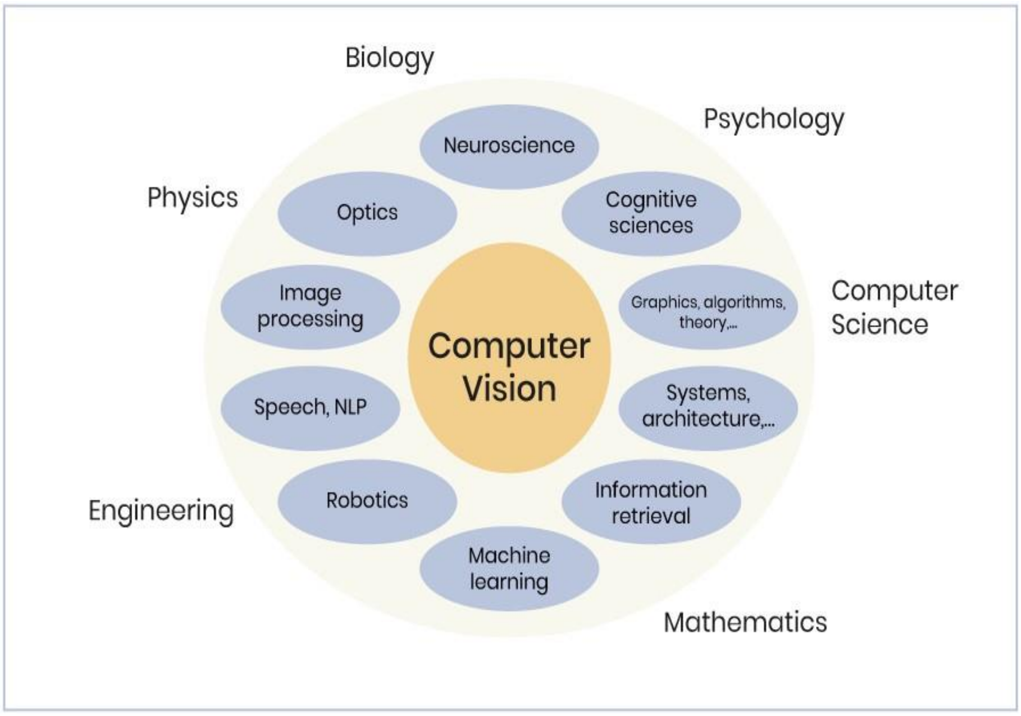


Fig 3.4 Computer Vision Applications

Open CV:

Fig 3.5 Features of OpenCV

OpenCV is a great source of open source computer information, machine learning, and image processing and now plays a serious role in real-time performance, which is so important in today's systems. By using it, we will be able to process photos or the videos to work out objects, faces, or maybe handwriting. When combined with various libraries, like NumPy, the python is ready to process the OpenCV listing for analysis. to spot an image pattern and its various features we use a vector space and perform mathematical functions on these elements.

First version of OpenCV and is named as opencv1.0. It is of no coast under the BSD license which is therefore free for both education or industry use. It contains C ++, C, Python and Java interface and supports various systems like Linux, Window, iOS, Android and Mac OS. It was designed, the main focus was on real-time computing. All items are coded with C &C ++ designed to take a upper hand from the multi-core process. From the image, we can extract a lot amount of information that is present in the original image. As in this picture abobe there are 2 faces available and the person (s) in the picture is wearing a bracelet, a watch, etc. So by taking a hand from OpenCV we can find all the information in the given image. It is a basic introduction to the OpenCV we can continue with applications and all things in our future articles.

Applications of CV:

- To recognize face
- Automaited inspiction and survaillance
- No. of individuals – count (foot traffic during a mall, etc)

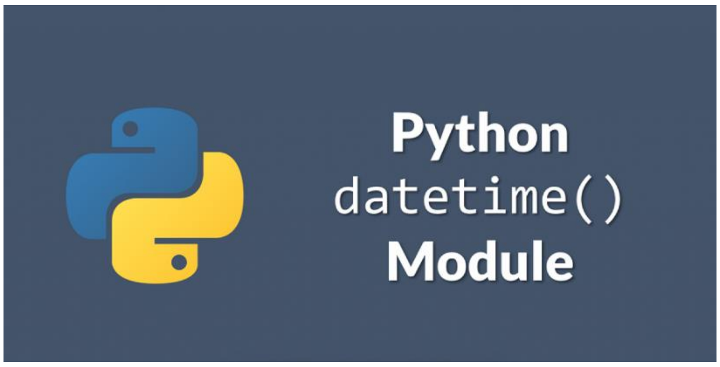
- Vehicle relying on road together with their speeds
- Interactive art installations
- Anomaly (defect) detection within the manufacturing process (the odd defective products)
- Road view image stitching
- Video&image search and retrieval
- Robot and driver-less car navigation and control

NumPy:

Numpy is a standard purpose processing package. It provides high performance for multidimensional object, and tools for working with these components. It is a basic computer science package via Python. NumPy, which represents Numerical Python, is a multidisciplinary library and a group of methods for filtering those members. Using NumPy, maths and logical operations in arrays can be determined. This study describes the basics of NumPy as its structure and surrounding. It also discusses the various functions of similar members, types of references, etc. An introduction to Matplotlib was also provided. All of this is explained with the help of examples of better understanding.

In addition to its obvious scientific uses, Numpy could be used as an efficient multi-sided container of standard data.

DateTime:



At Python, date & time is not the data itself, but the module that is called date may be imported to work with date and time. The Python Datetime module will be built into the Python, so there is no need to install it outside. The Python Datetime module provides classes to work with the date and time. These classes offer several activities to deal with days, times and intervals. Date and time of day are in Python, so if you fool yourself, you are actually changing things and not the character unit or time stamps.

Jupyter Notebook:

One of the Project Jupiter projects is that the Jupiter notebook. Jupiter notebook is an open source web application that equips users with coding tools to urge various results like data clearing, data visibility, editing, retransmission, mathematical modeling, machine learning, etc. The Jupyter notebook originally contained only three main languages, Julia, Python, and R which, when combined, gave birth to the name Jupyter. Currently, the jupyter notebook supports over 40 editing languages. These scripts are often easily shared via github or sent to a colleague via dropbox.

Sublime Text:

To make working on our project code and editing it , we wanted something that was very user friendly and easy to use , for that Sublime Text editor was the best in business for us at least . The main reason to use this was that it generates a project wide index of every class, method and function.

Some of its other Key features are:

1)You can personalize anything from key bindings, menus

Snippets , macros , completions and more

2)It has an unmatched syntax highlighting , and is quick in its responsiveness

3)You can do split editing i.e , edit files side by side or Edit two locations in one file.

CHAPTER 04

PERFORMANCE ANALYSIS

Algorithm improvements:
Optical Flow algorithms suffer from sudden movement because of algorithm limitations. the foremost common method of operation is to use a multi-measurement strategy. we want to make what's called a picture pyramid, within which each subsequent image are going to be larger than the previous one in terms of scale (for example the dimensions element. in keeping along th r fixred size of our window, the sudden movement in an exceedingly small-sized image is way larger than the larger one. The vectors established within the smaller images are going to be utilized in the subsequent large sections of the pyramid to realize the simplest results. As mentioned earlier, Dense Optical Flow algorithms calculate the moving vectore of atiny low feature set, therefore the commonest method here is to use a Shi-Tomasi corner detector. it's wont to find angles in a picture and to calculate the angle vector between two consecutive frames.

An attempt has been made to compare various Optical flow algorithms. Although Lucas-Kanade algorithm, does not yield a very high density of flow vectors, it is robust in presence of Noise.[11]. It is window based local method. Average angular error is less in Lucas-Kanade algorithm than Horn-Schunck algorithm. Average angular error of L-K is 4.3 while that of H-K is 9.8.Combining local and global methods can reduce average angular error to 4.2 as shown in Table 4.1. So it is advantageous to combine both local and global methods. There exist fast and accurate optical flow algorithms which can be applied in future.[12] Average angular error is less in Lucas-Kanade algorithm than Horn-Schunck algorithm but it is less density of flow vectors. Hence, in future local and global methods can be combined for requirement of dense flow estimate, preserve discontinuities and to make it robust to noise. Horn-Schunck and Lucas-Kanade algorithms work only for small motion. If object moves faster, the brightness changes rapidly, derivative masks fail to estimate spatiotemporal derivatives. Pyramids can be used to compute large optical flow vectors.

The Horn Schunck algorithm at a glance is as follows :

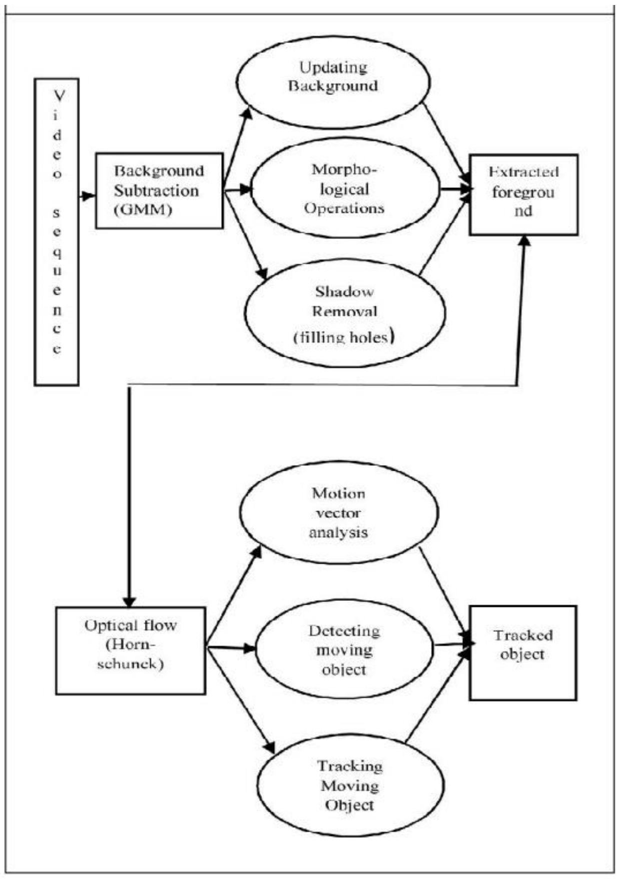


Fig 4.1 Horn Schunck algo flowchart

The Lucas – Kanade algorithm at a glance is as follows :

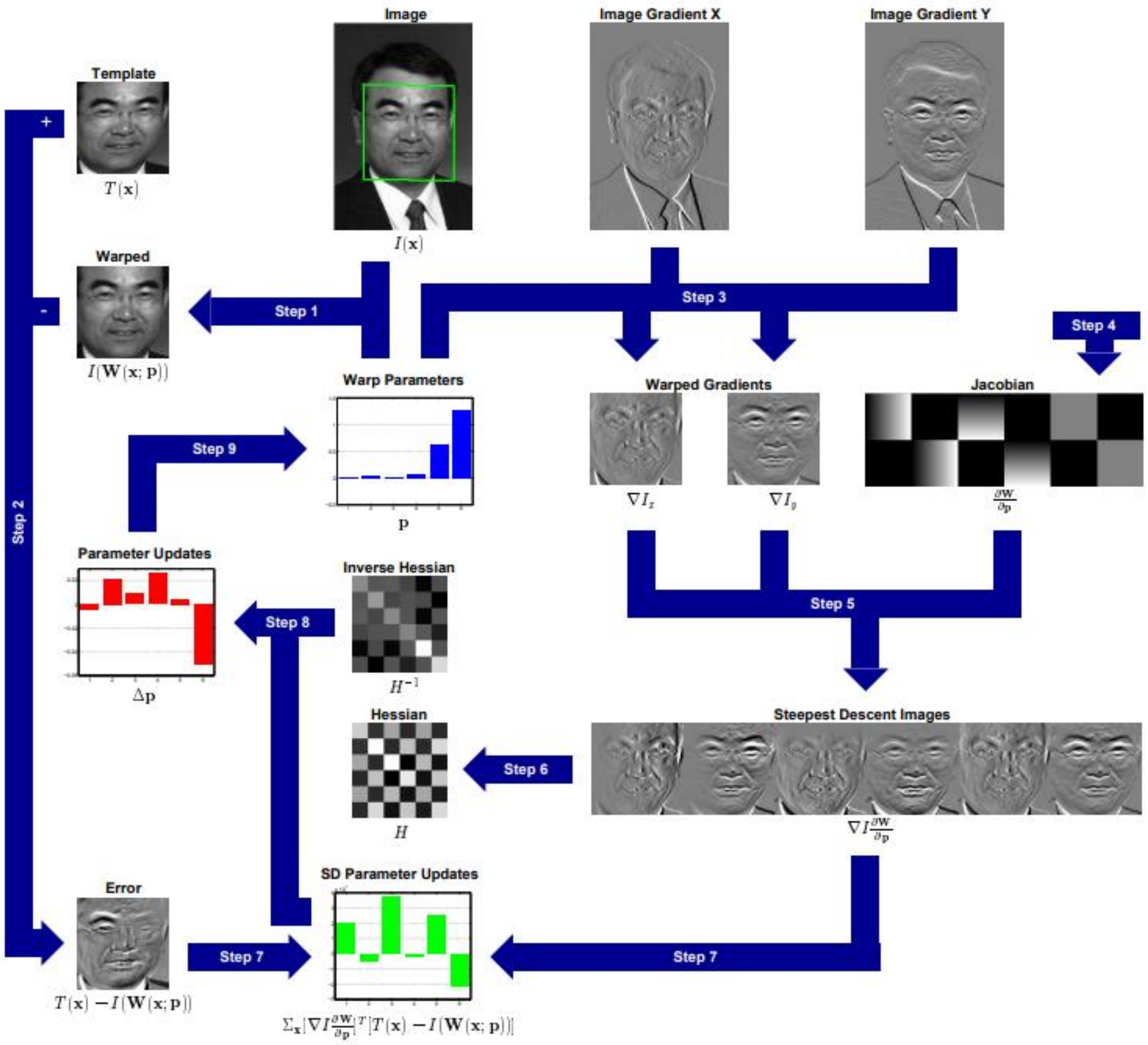


Image
Image Gradient X
Image Gradient Y
Template
$T(\mathbf{x})$
$I(\mathbf{x})$
Warped
Step 1
Step 3
Step 4
$I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
Warp Parameters
Warped Gradients
Jacobian
Step 9
∇I_x
∇I_y
$\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
Step 2
$\mathbf{p}$
Parameter Updates
Inverse Hessian
Step 5
Step 8
H^{-1}
$\Delta \mathbf{p}$
Hessian
Steepest Descent Images
Step 6
H
$\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
SD Parameter Updates
Error
Step 7
Step 7
$T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
$\Sigma_{\mathbf{x}}[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
+
-

Fig 4.2 Lucas Kanade algo flowchart

Based on these two algorithms, we compared the outputs and concluded the right algorithm.

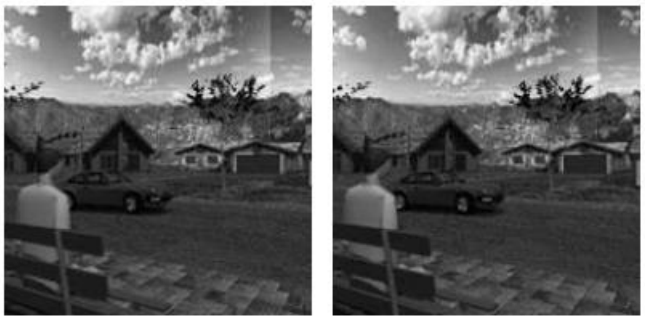


Fig 4.3.1 Original Images at time t and t+dt

After applying different algorithms:

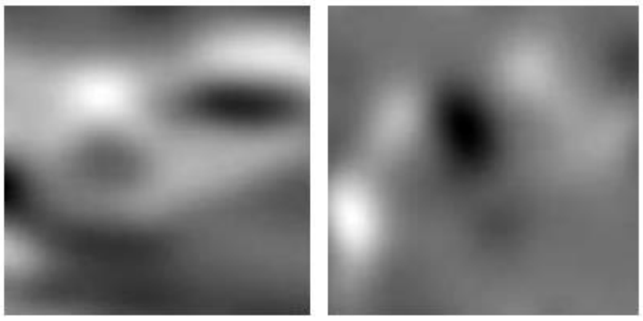


Fig 4.3.2 Images after applying differential technique

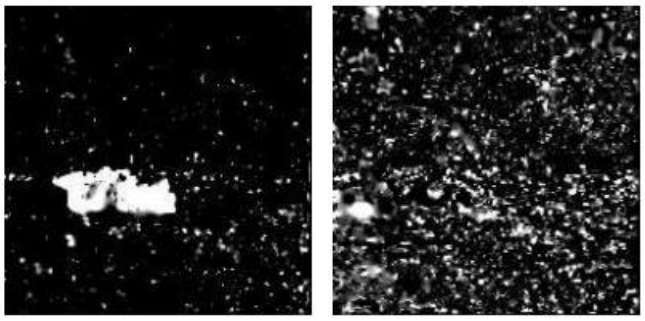


Fig 4.3.3 Images after applying HornSchunck algo

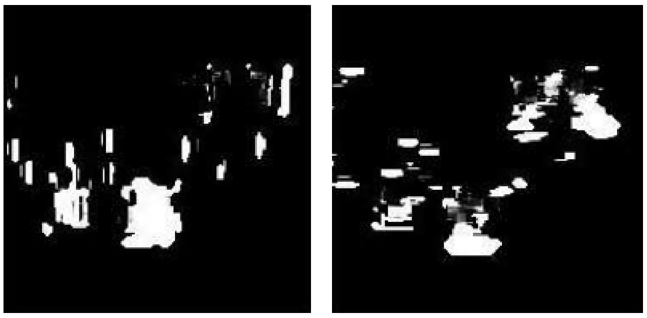


Fig 4.3.4 Images after applying Lucas Kanade algo

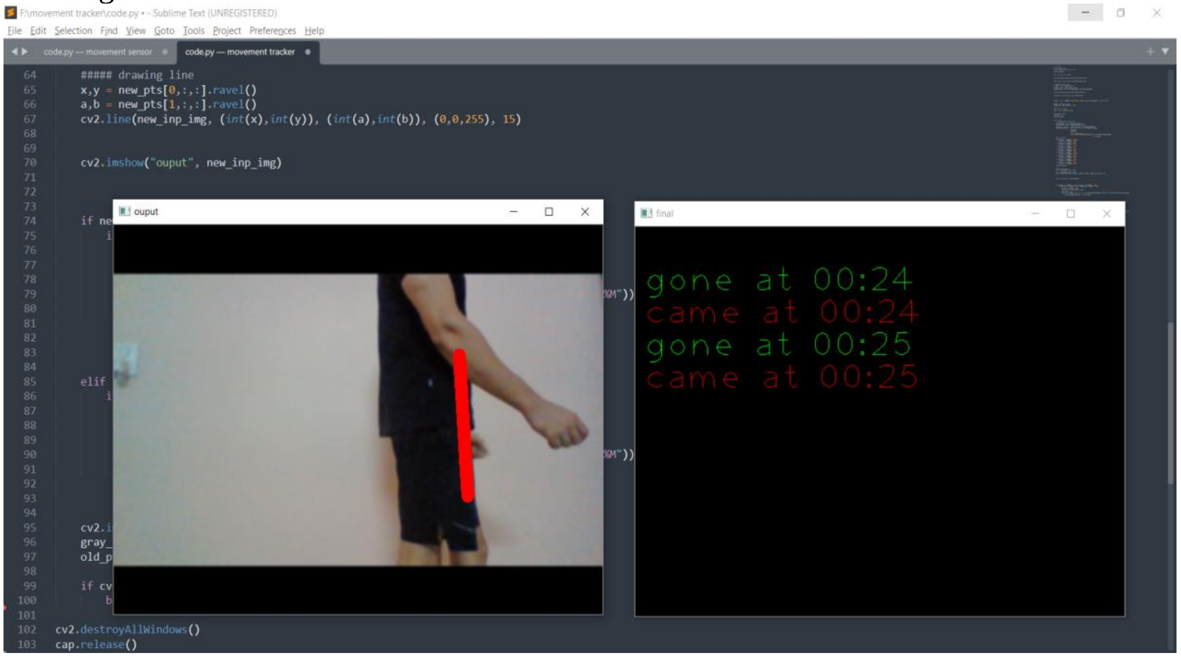
Comparison of Velocities:

Method	u	v
Horn-Schunck method	9.6837	7.1435
Lucas-Kanade method	3	4

These values are obtained by the application of lucas and horn schunck algorithm. In this table, v represents tangential component perpendicular to gradient direction and u represents normal component parallel to gradient direction. The velocity values are calculated in pixels per frame.

Now we can conclude that Lucas algorithm, when comapared to Horn S algo improves the signal strength as well as reduce noise which gives more accurate and relatively speed results.

Output we got after making the project to track the movement direction with respect to time is given as follows :



F:\movement tracker\code.py • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
code.py — movement sensor
code.py — movement tracker
ouput
final

```
        ##### drawing line
        x,y = new_pts[0,:,:].ravel()
        a,b = new_pts[1,:,:].ravel()
        cv2.line(new_inp_img, (int(x),int(y)), (int(a),int(b)), (0,0,255), 15)


        cv2.imshow("ouput", new_inp_img)



        if ne
            i




        elif
            i









        cv2.i
        gray_
        old_p

        if cv
            b

    cv2.destroyAllWindows()
    cap.release()
```

```
gone at 00:24
came at 00:24
gone at 00:25
came at 00:25
```

Fig 4.4 Output

CHAPTER 05

CONCLUSIONS

5.1 Conclusion

This project presented in this thesis not only detects the motion but also tracks the time that when an object is coming or leaving a particular place. It can have numerous applications and can be very useful in real time. There is no need of internet connectivity or any costly hardware. A camera with good pixels can get all the job done.

5.2 Applications

This project can have numerous applications in real world.

- Can be used to investigate crime in an efficient way.
  This project can tell you that whether the number of people entered in a particular place equal to the number of people left that particular place.
  It can tell at what time the person entered and left the hall.
  If we integrate facial recognition, we can even conclude that which person has not left the hall.
  So, we can get many useful insights which can help to solve crimes from this project.

- Can calculate population of a particular place at a given time.
  We can take a counter variable and initialize it with zero.
  Whenever a person entered a place, out project will detect that a person entered and increment the counter value with one and decrement it with one when a person left the place. So, the final value of counter variable can tell that how many people are presently available at a particular place.
- Take attendance in a better way than biometric or face detecting attendance system.
  We can add facial recognition and check if a person who entered the class/hall attended the class for a threshold time or not. If he/she fulfilled the time requirement, he'll be marked present else not.
  This can be done by subtracting his exiting time and entering time.

There can be many more applications of this project.

5.3 FUTURE SCOPE

- Integrate Facial Recognition
  Train the model which can identify the person who possibly can enter the room. Use database to keep the track.

- In order to allow the application to cover multiple rooms an architecture could be setup that would allow multiple devices to communicate.

- Implement classifiers created from machine learning algorithms which are capable of distinguishing between humans and objects.

- Predict hotel prices by calculating population of a particular place

5.4 ADVANTAGES

This can be operated on a simple camera. Nowadays almost every apartment/hall have cameras fitted. So it is a plus point that there is no need to buy expensive movement detectors and spend extra money to use the features of this project.

It works in real time and will continuously give you the status of room/hall.

It will reduce labour and save time which can be invested a somewhere else.

5.5 LIMITATIONS

The camera in use should be of good quality.

Sometimes it may not give the desired results if the background is not clear.

It is based completely on motion detection; it will not tell the difference between moving objects and a human's movement.

It can only be applied where the people have an agreement on surveillance policy.

REFERENCE

[1] Bhumika Gupta (2017) et al
https://www.researchgate.net/profile/PriyankaBadhani/publication/317058859_Study_of_Twitter_Sentiment_Analysis_using_Machine_Learning_Algorithms_on_Python/links/60fbebe50c2bfa282af92131/Study-of-Twitter-Sentiment-Analysis-using-Machine-Learning-Algorithms-on-Python.pdf

[2] Kartik Umesh Sharma (2017) et al
https://www.inderscienceonline.com/doi/abs/10.1504/IJCVR.2017.081234

[3] Mukesh Tiwari (2017) et al
http://www.ripublication.com/ijcir17/ijcirv13n5_07.pdf

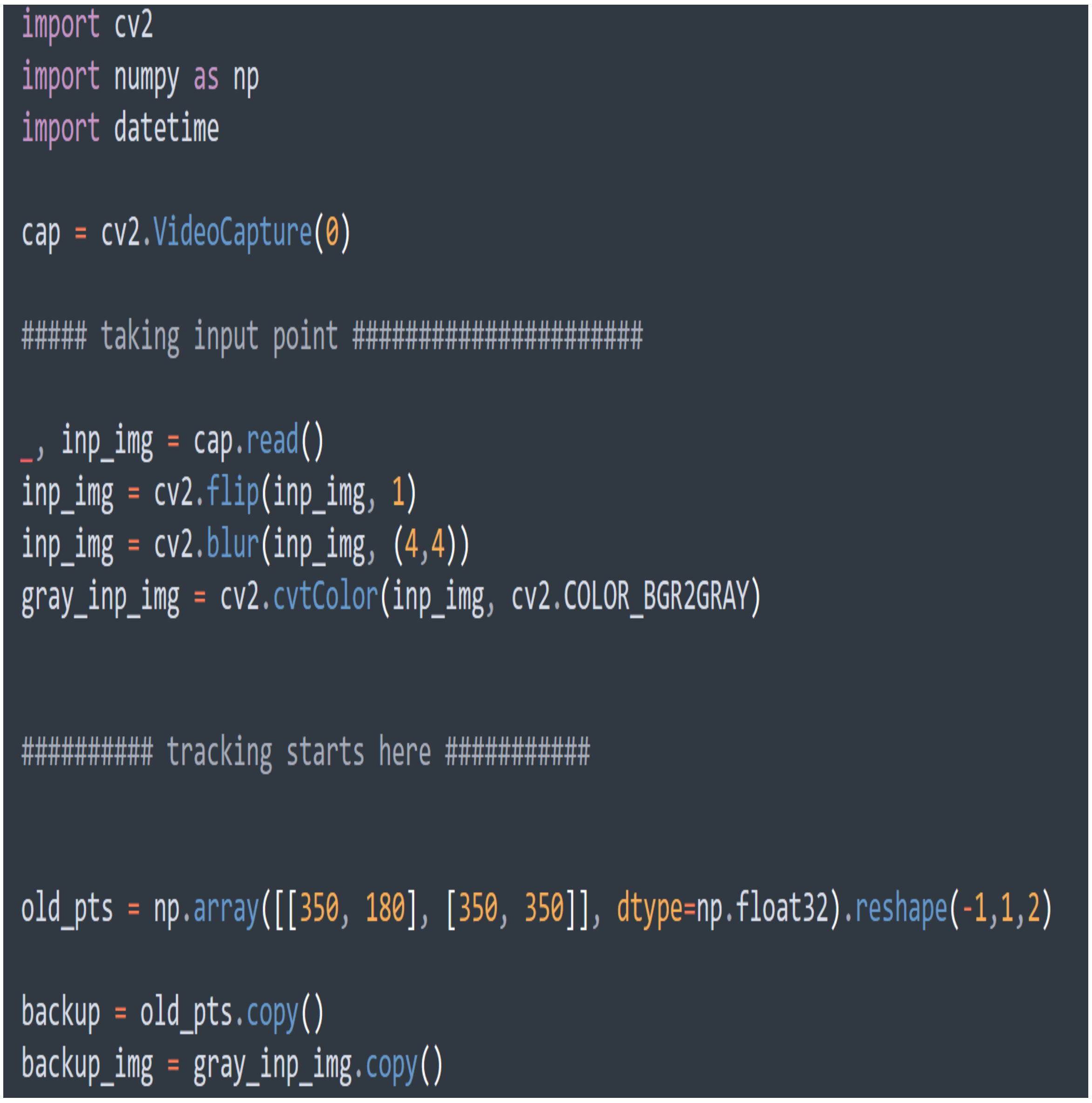
[4] Rupesh Kumar Rout (2015) et al
http://ethesis.nitrkl.ac.in/4836/1/211CS1049.pdf

Dealt with bugs and errors with the help of YouTube and Stack Overflow.

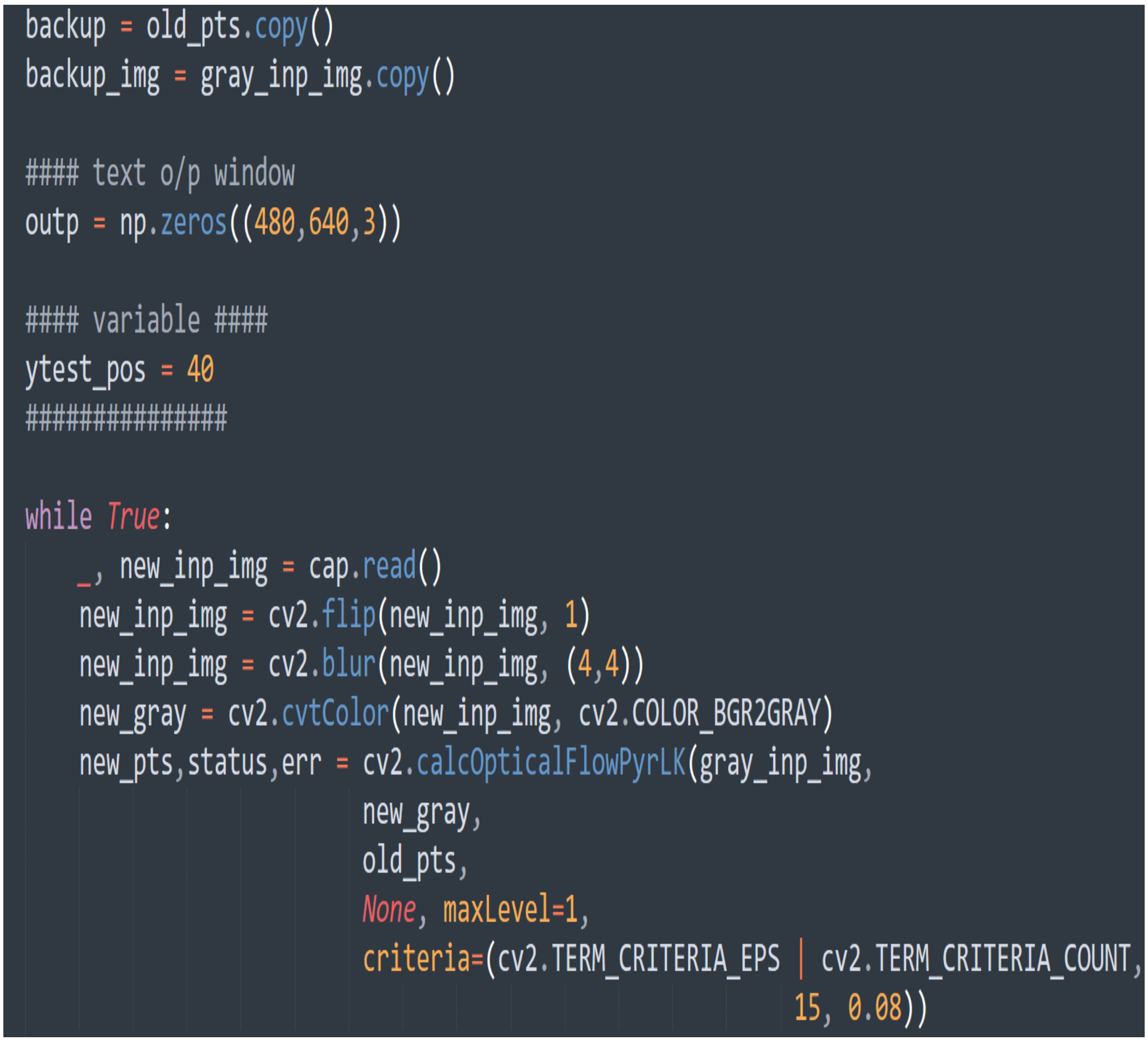
Studied about the required libraries/modules from GeeksforGeeks.

APPENDICES

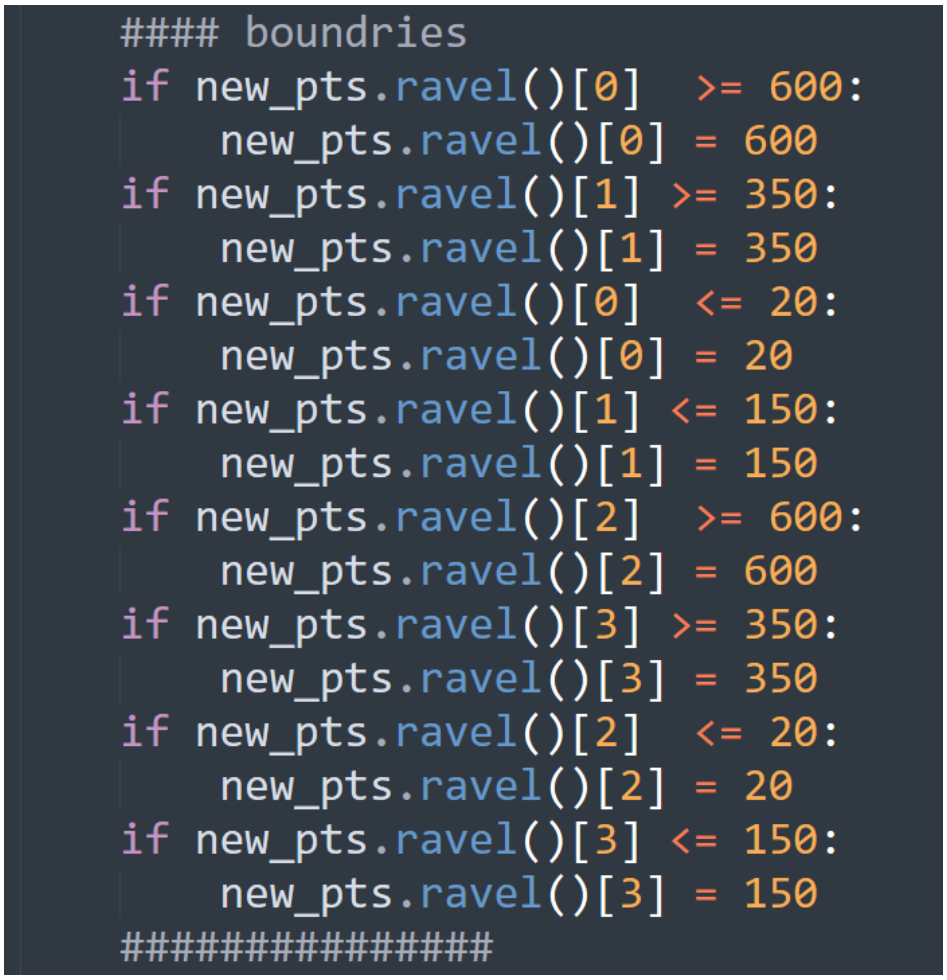
This section includes the source code:



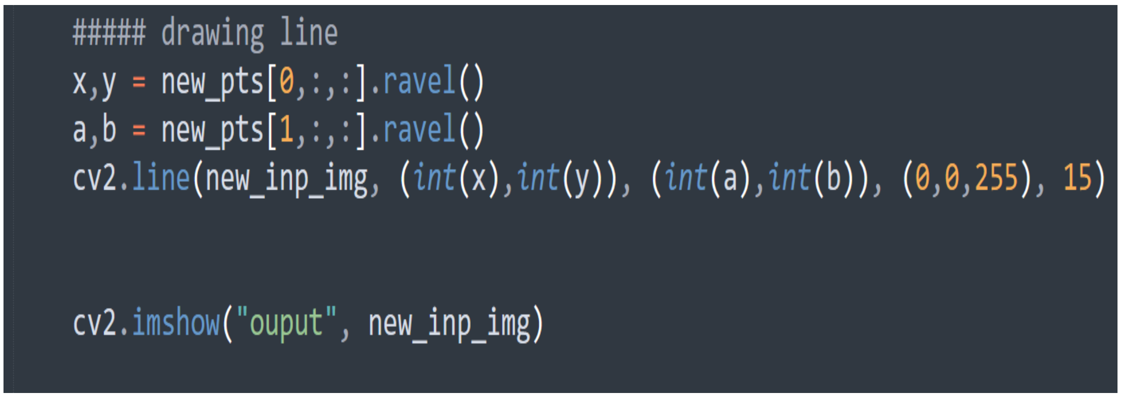
```python
import cv2
import numpy as np
import datetime

cap = cv2.VideoCapture(0)

##### taking input point #####################

_, inp_img = cap.read()
inp_img = cv2.flip(inp_img, 1)
inp_img = cv2.blur(inp_img, (4,4))
gray_inp_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)


########## tracking starts here ###########


old_pts = np.array([[350, 180], [350, 350]], dtype=np.float32).reshape(-1,1,2)

backup = old_pts.copy()
backup_img = gray_inp_img.copy()
```



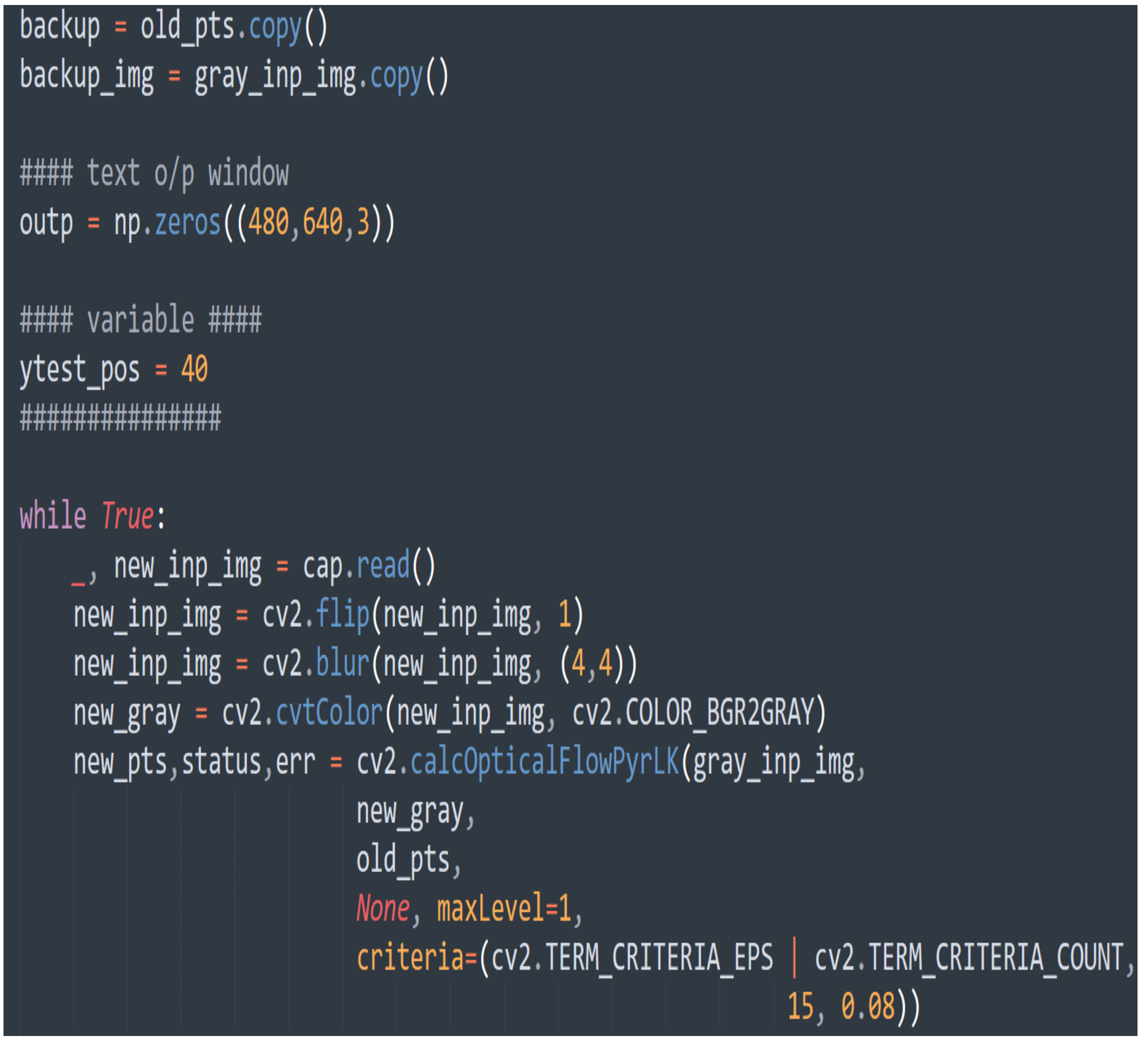
```
backup = old_pts.copy()
backup_img = gray_inp_img.copy()

#### text o/p window
outp = np.zeros((480,640,3))

#### variable ####
ytest_pos = 40
###############

while True:
    _, new_inp_img = cap.read()
    new_inp_img = cv2.flip(new_inp_img, 1)
    new_inp_img = cv2.blur(new_inp_img, (4,4))
    new_gray = cv2.cvtColor(new_inp_img, cv2.COLOR_BGR2GRAY)
    new_pts,status,err = cv2.calcOpticalFlowPyrLK(gray_inp_img,
                         new_gray,
                         old_pts,
                         None, maxLevel=1,
                         criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
                                                                15, 0.08))
```



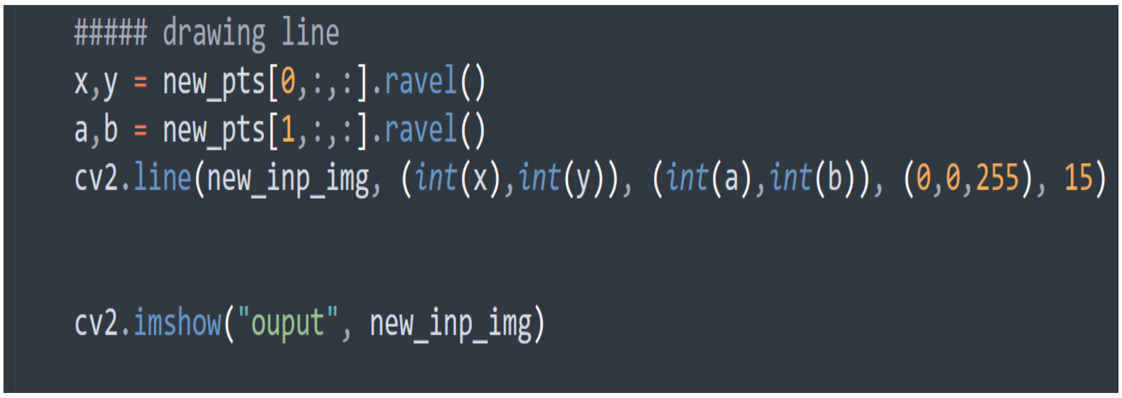
```
#### boundries
if new_pts.ravel()[0]  >= 600:
    new_pts.ravel()[0] = 600
if new_pts.ravel()[1] >= 350:
    new_pts.ravel()[1] = 350
if new_pts.ravel()[0]  <= 20:
    new_pts.ravel()[0] = 20
if new_pts.ravel()[1] <= 150:
    new_pts.ravel()[1] = 150
if new_pts.ravel()[2]  >= 600:
    new_pts.ravel()[2] = 600
if new_pts.ravel()[3] >= 350:
    new_pts.ravel()[3] = 350
if new_pts.ravel()[2]  <= 20:
    new_pts.ravel()[2] = 20
if new_pts.ravel()[3] <= 150:
    new_pts.ravel()[3] = 150
################
```



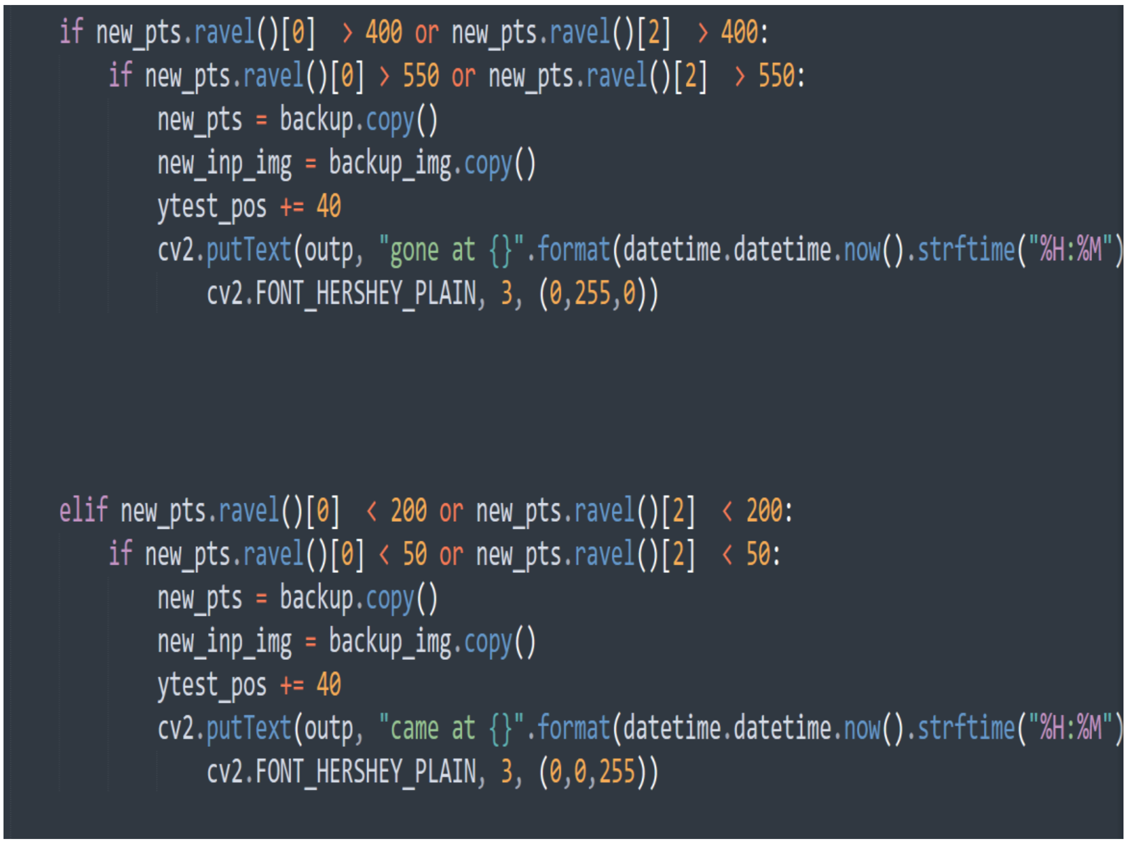
```
##### drawing line
x,y = new_pts[0,:,:].ravel()
a,b = new_pts[1,:,:].ravel()
cv2.line(new_inp_img, (int(x),int(y)), (int(a),int(b)), (0,0,255), 15)


cv2.imshow("ouput", new_inp_img)
```

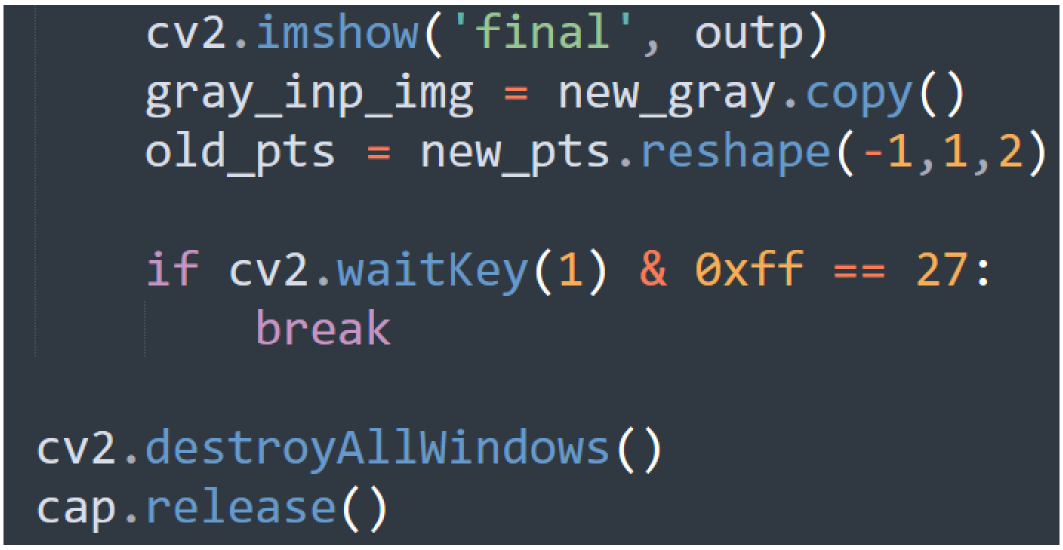


```
backup = old_pts.copy()
backup_img = gray_inp_img.copy()

#### text o/p window
outp = np.zeros((480,640,3))

#### variable ####
ytest_pos = 40
###############

while True:
    _, new_inp_img = cap.read()
    new_inp_img = cv2.flip(new_inp_img, 1)
    new_inp_img = cv2.blur(new_inp_img, (4,4))
    new_gray = cv2.cvtColor(new_inp_img, cv2.COLOR_BGR2GRAY)
    new_pts,status,err = cv2.calcOpticalFlowPyrLK(gray_inp_img,
                         new_gray,
                         old_pts,
                         None, maxLevel=1,
                         criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,
                                                         15, 0.08))
```



```
    ##### drawing line
    x,y = new_pts[0,:,:].ravel()
    a,b = new_pts[1,:,:].ravel()
    cv2.line(new_inp_img, (int(x),int(y)), (int(a),int(b)), (0,0,255), 15)


    cv2.imshow("ouput", new_inp_img)
```


```
if new_pts.ravel()[0]  > 400 or new_pts.ravel()[2]  > 400:
    if new_pts.ravel()[0] > 550 or new_pts.ravel()[2]  > 550:
        new_pts = backup.copy()
        new_inp_img = backup_img.copy()
        ytest_pos += 40
        cv2.putText(outp, "gone at {}".format(datetime.datetime.now().strftime("%H:%M")
            cv2.FONT_HERSHEY_PLAIN, 3, (0,255,0))



elif new_pts.ravel()[0]  < 200 or new_pts.ravel()[2]  < 200:
    if new_pts.ravel()[0] < 50 or new_pts.ravel()[2]  < 50:
        new_pts = backup.copy()
        new_inp_img = backup_img.copy()
        ytest_pos += 40
        cv2.putText(outp, "came at {}".format(datetime.datetime.now().strftime("%H:%M")
            cv2.FONT_HERSHEY_PLAIN, 3, (0,0,255))
```


```
    cv2.imshow('final', outp)
    gray_inp_img = new_gray.copy()
    old_pts = new_pts.reshape(-1,1,2)

    if cv2.waitKey(1) & 0xff == 27:
        break

cv2.destroyAllWindows()
cap.release()
```