

Motion Detection Of An Object Using OpenCV

A Project

Report

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

Bachelor of Technology

In

Department of Computer Science Engineering

by

Jaladi Sai Sowri Vikas(180030271)

Tulabandu Mani Shankar(180030374)

under the supervision of

Ms. M .Praveena

ASST.PROFESSOR



Department of Computer Science Engineering

K L E F, Green Fields,

Vaddeswaram- 522502, Guntur(Dist), Andhra Pradesh, India.

2021-2022

KONERU LAKSHMAIAH EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Declaration

The Project Report entitled “**Motion Detection Of An Object Using OpenCV**” is a record of bonafide work of J.Sai Sowri Vikas (180030271), T.Mani Shankar (180030374) submitted in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science and Engineering** to the **Koneru Lakshmaiah Education Foundation** during the academic year 2021 - 2022. The results embodied in this report have not been copied from any other departments / University/ Institute.

180030271 – J.Sai Sowri Vikas

180030374 – T.Mani Shankar

K L (Deemed to be) University

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Certificate

This is to certify that the Project Report entitled “**Motion Detection Of An Object Using OpenCV**” is being submitted by J.Sai Sowri Vikas(180030271), T.Mani Shankar(180030374) submitted in partial fulfillment for the award of B.Tech in Computer Science Engineering to the K L University is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

Signature of Supervisor

M.Praveena

(Asst.Professor)

Signature of HOD

Hari Kiran Vege

(Professor)

Signature of the Examiner

ACKNOWLEDGEMENT

Our sincere thanks to **Ms M.Praveena mam** in the project for her outstanding support throughout the project for the successful completion of the work.

We express our gratitude to **Mr V. HARIKIRAN** sir, Head of the Department for computer science and Engineering for providing us with adequate facilities, ways and means by which we can complete this project-1.

We would like to place on record the deep sense of gratitude to the honourable Vice Chancellor, K L University for providing the necessary facilities to carry the project-1.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially our classmates and our friends for their support in the completion of our project-1.

Sai Sowri Vikas (180030271)

T.Mani Shankar (180030374)

TABLE OF CONTENTS

CHAPTERS	PAGENO
1. ABSTRACT	8
2. INTRODUCTION & PROBLEM DEFINITION	10
3. LITERATURE SURVEY & REQUIREMENTS	13
4. DIFFERENT TECHNIQUES FOR MOTION DETECTION	15
5. CONCEPT BEHIND THIS	18
6. ALL ABOUT OPENCV	20
7. OVERVIEW	28
8. CNN	30
9. MODULE IMPLEMENTATION & EXPERIMENTAL RESULTS	35
10. OUTPUTS	41
11. VARIOUS OBJECT DETECTION ALGORITHMS IMPLEMENTED IN PYTHON	46
12. CONCLUSION & FUTURE SCOPE	53
13. REFERENCES	55
14. PLAGIARISM CHECK – (PAPERPRESS,TURNITIN)	57

CHAPTER 1 : ABSTRACT

ABSTRACT

It is titled "Motion Detection Application Using OpenCV". From the title, it is obvious that the project is going to use the "OpenCV package" to accomplish the objective of "motion detection". The Web Cam is configured to store a picture whenever a movement occurs in front of it.

This study also aims to reduce the amount of storage space required for activities like salient item identification and face recognition.

In this example, we will detect motion from the default webcam using OpenCV, and a bounding box will be displayed along with the motion-detected frames. With this knowledge, we can recognize that multiple frames make up videos. To detect motion, we would calculate the absolute difference between two continuous frames. If the difference is greater than the threshold, we would detect the motion.

Open Computer Vision (OpenCV) A Python Library, Written In C++ Powered By Deep Learning Algorithms, Designed By Neural Networks, Called Convolutional Neural Networks (CNN) To Understand The Sense Of Images.

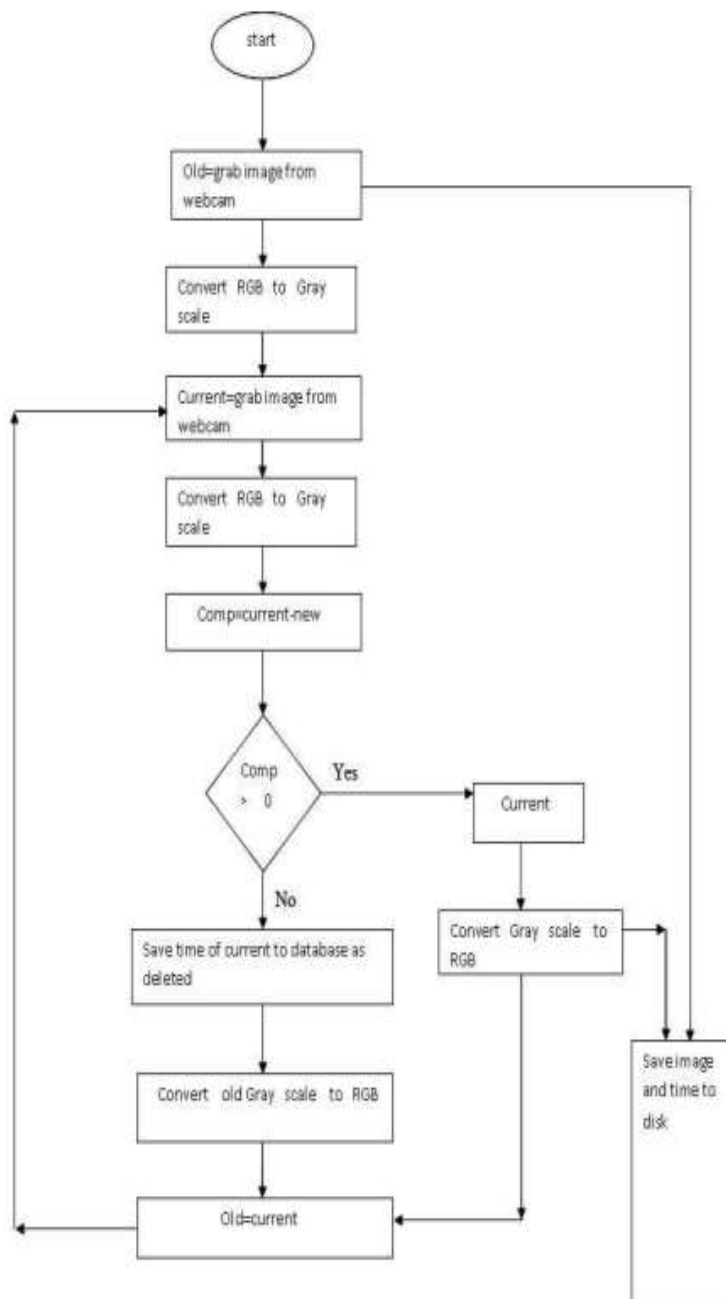
Computer Vision Can Be Considered As A Part Of Artificial Intelligence, Where Computers Are Trained With Machine Learning And Deep Learning Algorithms, Such That They Become Capable To Recognize And Process The Objects From The Video/Images Same As A Human Recognizes

CHAPTER 2 : INTRODUCTION

INTRODUCTION

This project focuses on using webcams for automatic motion detection for purposes of security, which is one of the most difficult problems in the world. Our project Motion Detection of an object Using OpenCV is an example of how tech can help us achieve this goal in present day's society. Having a human eye on the world 24x7 is simply impossible. As a means to achieve safety, smart surveillance systems with image processing techniques are very useful. These features, provided by our surveillance systems, are necessary for autonomic monitoring. In the first part of this project, related works and problems of our previous surveillance systems are discussed. A description of the new version of the surveillance system is then provided. There are many approaches for the comparison process, but here the RGB images are converted to grayscale and then compared, then they are converted back to RGB before being stored. To reduce the pixel values, grey scale reduction is used. Meanwhile the frames are pixel-by-pixel matched. Only position changing objects on the frame are revealed by the change in the frame. The suggested approach's major goal is to minimise storage space by recording only the frames with motion rather than the entire movie.

The user must first initialise the camera before selecting the frame size in the web cam menu. Because the frame is saved in the provided place, the frame size and frequency can be changed. This is where the image is saved as a frame. It will then begin recording the video, setting the first image as a backdrop and taking the following image with the same frame size. . We may examine the motion by subtracting the background picture from every given time window of footage (the detection for motion is entirely pixel to pixel detection). Pixels are those with a result nearer to zero are considered background, whereas pixels with a bigger value are considered foreground. As a result, once we have the backdrop image model, this one is simple, efficient, and straightforward to apply.



Configuring the Web Cam & Capturing the Videos

PROBLEM DEFINITION

Our task is to give a webcam, that can detect the motion or any movement in front of it. This should return a file, this file should contain for how long the human/object was in front of the camera. This all process must perform using an Deep Learning approach.

CHAPTER 3 - LITERATURE SURVEY

LITERATURE SURVEY

Object detection has been the focus of most of the previous research (Chandan G, Ayush Jain, Harsh Jain, Mohana, BenAyedetal., 2015; Foytik et al.,2011; RamyaandRajeswari,2016;RishaandKumar,2016;Shenetal.,NajvaandBijoy,2016 ;2013; Viswanath et al.,2015Soundrapandiyan and Mouli, 2015;) ,Object tracking (Bagherpour et al., 2012; Cokun and Åænal, 2016; Lee et al., 2012; Poschmann et al.,2014; Weng et al., 2013; Yilmaz et al., 2006;Zhang et al., 2016) and Object recognition (Chakravarthy et al., Ha and Ko, 2015; 2015; Gang et al., 2010; Elhariri et al.,2015; Nair et al., 2011) The existing system does not detect objects using OpenCV. Therefore, live tracking of an object is difficult. My project uses OpenCV to increase the accuracy of the Object detection in the live video. It also provides a method to track the moving object using effective techniques and high accuracy deep learning concepts to improve the accuracy of the Object detection. As well as detecting the object, it calculates how long the object is in our frame (calculates the time the object entered and left).

REQUIREMENTS

Hardware /Software	Hardware / Software element	Specification /version
Hardware	Processor	i3
	RAM	2GB
	Hard Disk	250GB
Software	OS	Windows , Jupyter NoteBook. Python 3.
Web	Anaconda Navigator	Python Code

CHAPTER 4 – DIFFERENT TECHNIQUES FOR MOTION DETECTION

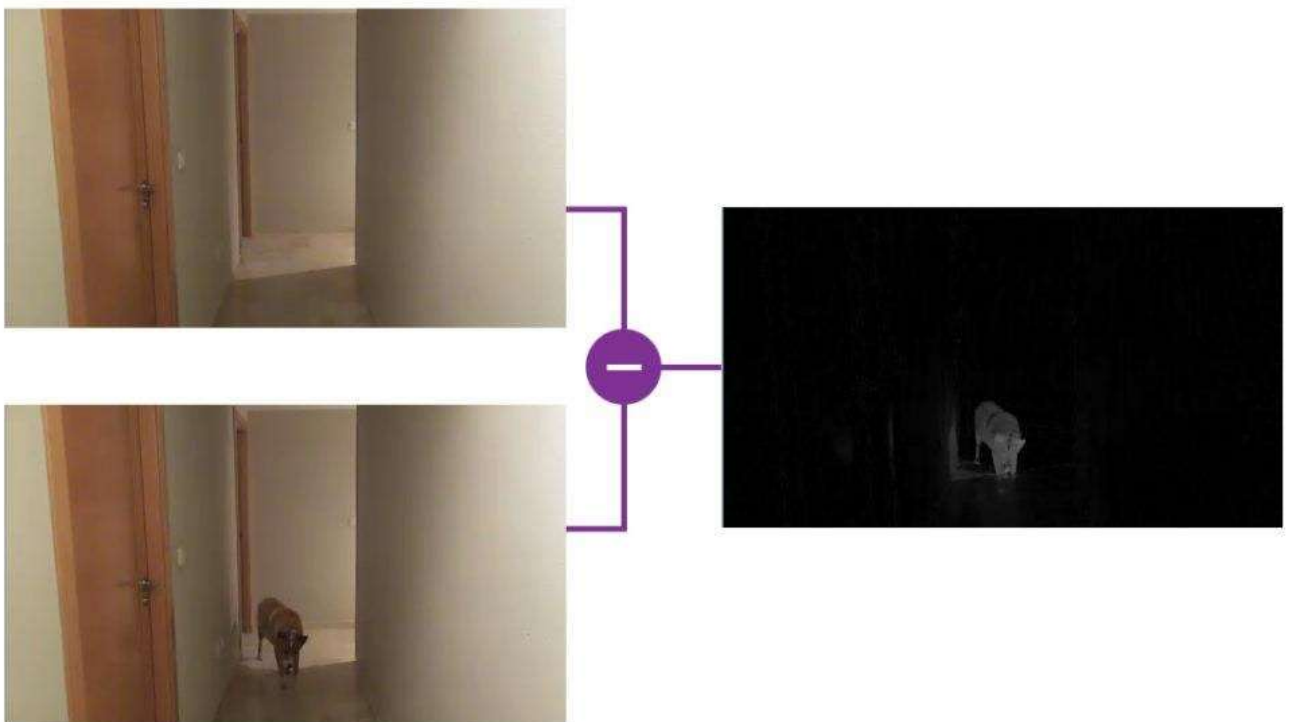
Different Techniques For Motion Detection

Motion detection is usually necessary when using a machine vision system , such as for counting how many cars pass through a toll or how many people cross a certain area. When these situations occur, we should evacuate the people or vehicles from the area as quickly as possible.

The performance theory does not have any general cases like other studies, but instead relies on various strategies, methods, and algorithms. There are lots of techniques used in OpenCV and Computer Vision, which will depend on the application. Let's look at a few of them.

--Background subtraction--

We remove the consecutive frames of a video by taking a snapshot of the event and removing it. The framework we will examine will begin with a background image, which is an image without movement. Because of this, we issue different frames for different domains.



As shown in the image, in this scene the background is black and different colors represent motion. This is a simple method that does not require attachments to the subject to identify it as a sensor, beacon, or special suit, and it works even when the subject moves. However, background subtraction is sensitive to slight changes in light, such as shadows or natural light. For

objects or subjects with similar colors to the background, either the movements are not detected or the movements are poorly detected.

–It is possible to achieve background subtraction by using either reference images or previous frame frames. There are two methods, depending on how the background is obtained.

Reference Image Subtraction

In general, the first frame is taken from a video sequence, and each frame is subtracted from the reference image to create the moving elements.

The natural light changes at 10:00 and 18:00 will affect a reference image taken in natural light. In addition, camera movements will alter the image. Although this method successfully tracks the silhouettes of moving objects in controlled lighting environments, even when very small movements are present, in controlled lighting environments this method is unable to detect the silhouettes of moving objects.

Subtraction with previous frames

The background color is determined by previous frames. In this algorithm, a reference image is allowed to remain static for a period of time, then a delay is applied and the resulting frames are compared. The length of this delay will be affected by several parameters such as speed.

It really is, however, a pretty strong method to alterations in lighting and camera motions, and it stabilises itself towards identifying silhouettes after a period. It isn't very good at identifying humans or directional movement.`.

CHAPTER 5 – CONCEPT BEHIND THIS

Concept

Theory

Each object breaks down into its own set of features that help it be identified in images and videos. Object recognition is part of computer vision. There have been several developments in recent years that have helped identify objects even when partially hidden from direct view with better algorithms:

Edge compatibility

- Calculates the number of edges using edge tracking techniques;
- Impact of lights and colors on the edges.

Matching gray

- Pixel range is calculated from both the pixel density and location of grayscale images. You can do the same thing again with color.

Gradient Matching

Using gradient comparisons can also increase the dynamic range of the image. Matching is done by matching grayscale images.

Since images are made in pixels, it is easy to find the next pixel multiple times and connect it to the current one at any time.

Pandas

Pandas is **an open source Python package** that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays

CHAPTER 6 – ALL ABOUT OPENCV

All About of Open CV

Computer vision and machine learning are performed using OpenCV (Open Source Computer Vision). In addition to providing a standard infrastructure for computer vision applications, Machine perception will be used more widely in commercial products with OpenCV. Because OpenCV is a BSD-licensed software, it is simple for enterprises to use and change the current code. Currently, the OpenCV library contains around 3000 algorithms, all of which have been efficiently tuned. In these categories can be found computer vision algorithms, classic algorithms, and machine learning algorithms. Operating systems such as Windows, Mac OS, Linux, and Android support classic algorithms as well as computer vision algorithms and machine learning algorithms. Java, MATLAB, Python, C, C++, etc. Nearly 500 algorithms exist, and countless functions support or assist them. CUDA and OpenCL interfaces can be used for implementation for the advancement of technology. In order for OpenCV to work with Python 2.7, we need to install the NumPy library first. OpenCV is written in C++, and provides a templated interface that integrates with STL containers beautifully.

In the software, low level functions are used to process image data as well as high level algorithms for face detection and matching, object tracking, and feature matching. Some of the key image processing methods are listed below.:

Image Filtering:

There are two types of image filtering. In a linear image filter, pixel values around the input pixel are used to determine the output pixel value. As for henonlinear image filtering, it does not produce a linear output based on its inputs

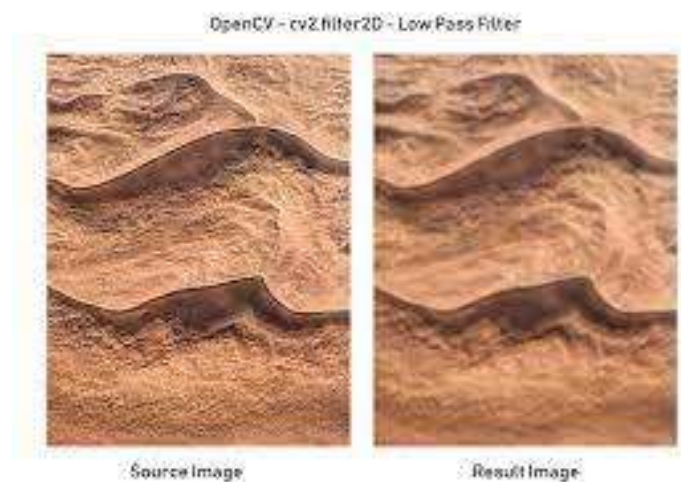
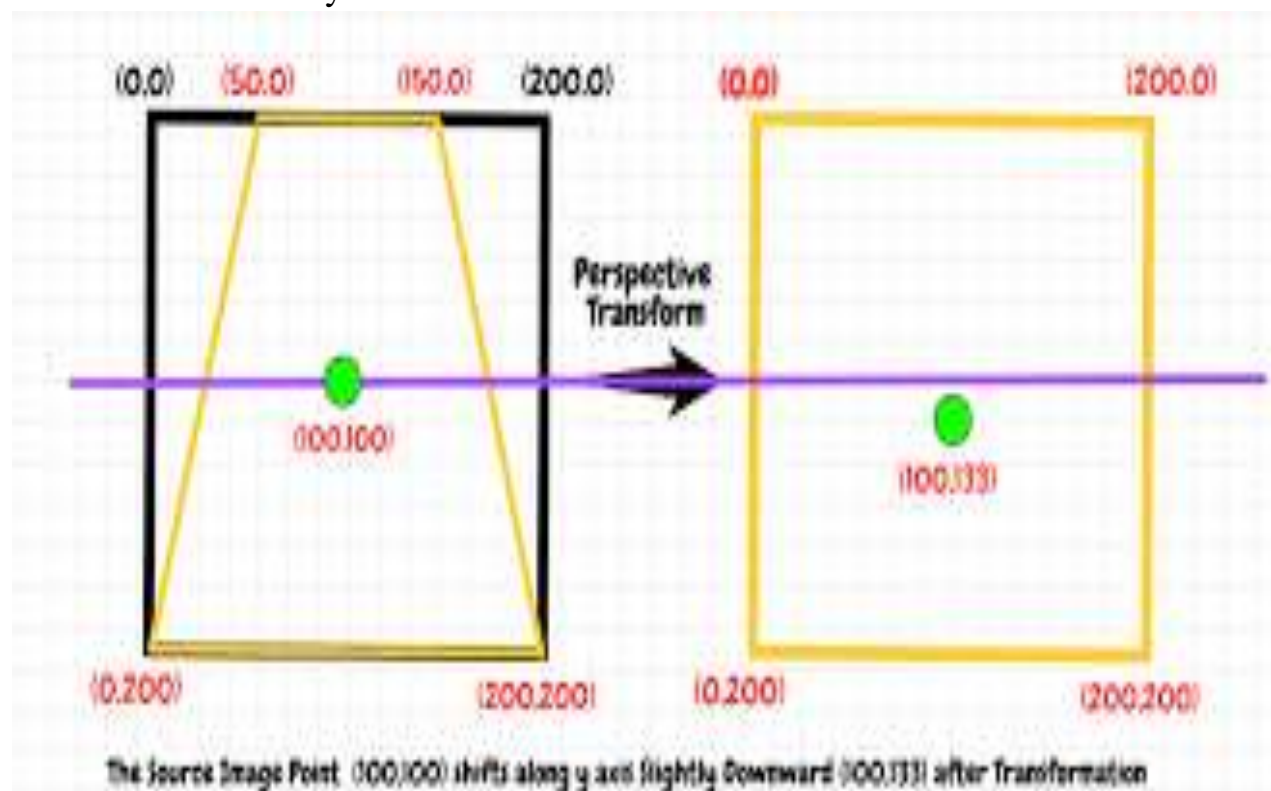


Image Transformation:

In the case of basic image transformations, simple arithmetic operations are applied to the image data. Image transformations create "new" images from two or more inputs that highlight particular features or properties that are of interest to the user. When comparing images collected at different dates, image subtraction is commonly used to identify changes.

Radon Transform: Images can be reconstructed using fan-beam and parallel beam projection data

- Discrete Cosine Transform: Compresses images and videos
- Discrete Fourier Transform: Analyzes frequency and filters
- Wavelet Transform: It can be used for denoising, fusing, and performing discrete wavelet analysis



Object Tracking:

There are many applications of object tracking in computer vision, including surveillance, human - computer interaction, and medical imaging. Object tracking involves the process of finding an object (or more than one object) over

a series of images.



Feature Detection:

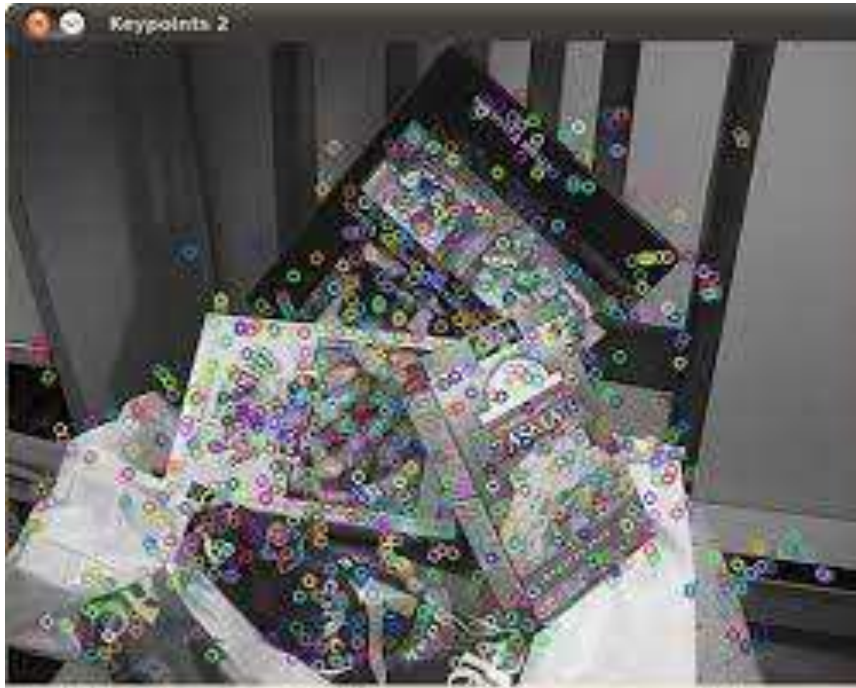
In computer vision, features are used as starting points for many algorithms because they are what make an image interesting. The feature detector is often the determining factor concerning how good the overall algorithm will be, since features constitute the starting point and primary primitive for subsequent algorithms. Feature detection involves detecting specific visual elements, such as lines, edges, and angles in the visual stimulus. This is useful for determining the local structure of the image (local information content).

The module CORE contains the basic data structures and functions used by the other modules of openCV for image processing applications.

The IMGPROC module contains image processing related functions, such as filtering linear and non-linear images and transforming geometric images.

Module Video contains algorithms for object tracking and motion estimation. Module ML provides machine learning tools. The HighGUI module provides the necessary interfaces for interfacing with the computer and supporting

multiplatform windowing.



SAMPLE IMAGE PROCESSING APPLICATIONS-OPENCV

A. Intruder alarm system using motion dection

We can use this type of alarms to detect or prevent criminal activity. It uses the simple motion detection for detecting the intruder. Motion detection is usually a softwarebased monitoring algorithm. It signals the camera to begin capturing the event when it detects motions. In imageprocessing, the image is treated as a two dimensional signal. The video captured by the camera is analyzed by the OPENCV program to detect motion.

Steps for finding motion-detection using OpenCV

1. Capture the video from the camera `CvCapture* capture = cvCaptureFromCAM(CV_CAP_ANY);`
2. Capture the frame1 `IplImage* frame1 = cvQueryFrame(capture);`
3. Capture the frame2 `IplImage* frame2 = cvQueryFrame(capture);`
4. Blur the images slightly to reduce the noise `blur (frame1, frame1_blur, Size(4, 4)); blur (frame2, frame2_blur, Size(4, 4));`
5. Find the absolute difference `absdiff (frame2_blur, frame1_blur, result_frame);`

B. Authentication System using Face Detection

Face authentication is commonly offered as an alternative to passwords for logging into the system. We can write an efficient authentication application using openCV. This application needs web camera to capture the face. The captured image is verified with the images stored in the database of faces. The 'face recognition' involves two--phases:

Face Detection

OpenCV has a face detector called "Haar Cascade classifier" that searches for faces in input images, then cropping and extracting the face. Face detection examines an input image from a camera or live video to determine whether or not it contains a face. The classifier uses xml file (haarcascade_frontalface_default.xml) to determine how each picture spot should be classified. In openCV 2.4.10, the xml file is in path "opencv/sources/data/haarcascades".

Face Recognition

An image of a face is compared with a database of images of faces in another phase. The openCV library contains a face detector module that can work 90-95% on clear images. Blurred images and glasses make it more difficult to identify faces

The frames can be grabbed from web camera by using following function.

```
/* Take the next frame of the camera. Waits for the next frame to be available before giving direct access to it, therefore don't change or release the returned picture! On the first frame, it will automatically configure the camera. */
```

```
IplImage* getCameraFrames(CvCapture* &cameras)
```

```
{
```

```
    IplImage *frame;
```

```
    int w;
```

```
    int h;
```

```
    // If the camera has not been initialized, then open it.
```

```
    if (!camera) //not there camera
```

```
{
```

```

printf("Accessing the camera for use...\n");
camera = cvCreateCameraCapture( 0 );
if (!camera) //not there camera
{
printf("we Couldn't access the camera.\n");
exit(1);
}
// Try to set the camera resolution to 320i
//x= 240.
cvSetCaptureProperty(camera, CV_CAP_PROP_FRAME_WIDTH, 320);
cvSetCaptureProperty(camera, CV_CAP_PROP_FRAME_HEIGHT, 240);
// Grab the first frame, to make sure that the
//camera is initialized.
frame = cvQueryFrame( camera );
if (frame) //frame is activated
{
w = frame->width; h = frame->height;
printf("Got the camera required at %dx%d resolution.\n", w, h);
}
// Wait a little, so that the camera can auto-adjust its brightness. Sleep(1000);
// (in milliseconds) }.
frame = cvQueryFrame( camera );
// Wait until the next camera frame is ready, then grab it
if (!frame) //not the frame
{
printf("Could not grab a camera frame.\n");
exit(1);
}

```

```
}  
return frame;  
}
```

C. Edge Detection System

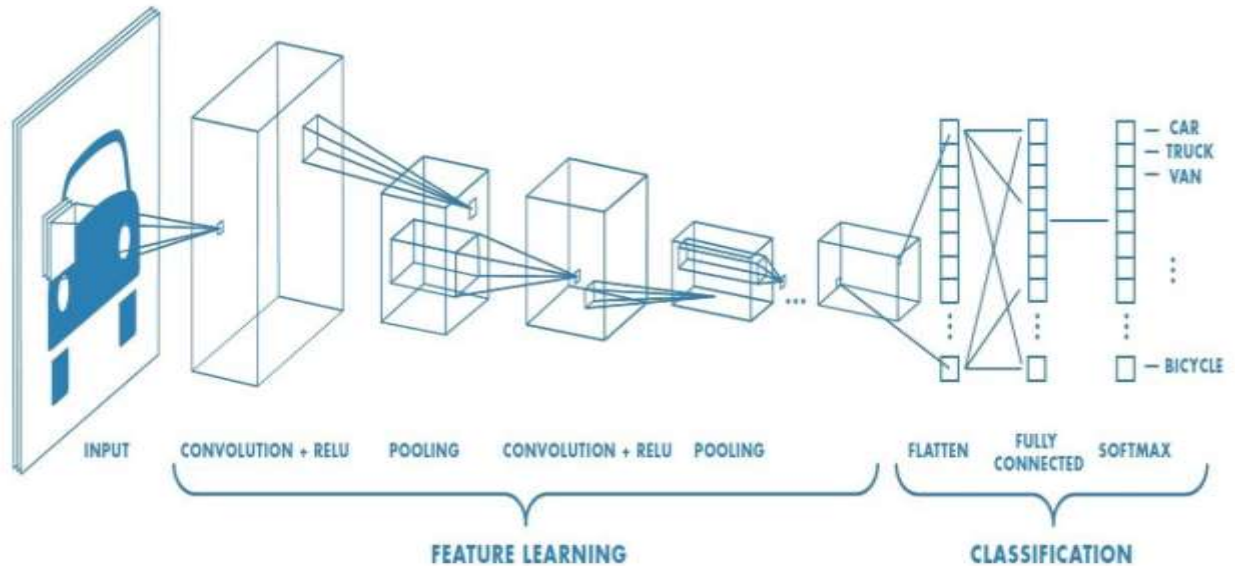
It uses the popular edge detection algorithm called canny edge detection. It was created in 1986 by John F. Canny. The structural information can be extracted from the input image by applying this technique.

CHAPTER 7 – CNN

Convolutional Neural Networks

A convolutional neural network (ConvNet / CNN) may be a deep learning algorithm which will take an input image, assign importance (weights and learnable biases) to varied aspects / objects of the image, and be ready to differentiate one from the opposite . 'other. The preprocessing required in a ConvNet is much less than in other classification algorithms. While in primitive methods the filters are designed by hand, with sufficient training, ConvNets has the ability to learn these filters / features.

The architecture of a ConvNet is analogous thereto of the neuron connectivity model within the human brain and was inspired by the organization of the visual area. Individual neurons respond to stimuli only in a narrow region of the visual field known as the receptive field. A collection of such fields overlaps to cover the entire visual area



(Architecture of Convolutional neural network)

CHAPTER 8 – OVERVIEW

Over view

We Begin By Importing The Necessary Libraries Required,

Step 1 :-

During this step, we would import Cv2 (OpenCV) and Pandas. In general, videos are a mixture of images, so OpenCV would be used to preprocess the images.

```
In [3]: # importing OpenCV, time and Pandas library
import cv2, time, pandas
from datetime import datetime
```

Assignment Of A Static Frame To Compare With-

Every frame needs to be compared with the first frame, which will be called a static frame. This way, any motion in the frames will be detected based on the difference between the frames. When the webcam first opens, the static frame will be none and will then be initialized with the first frame.

Step 2 :-

Create An Object To Capture Video-

In order to capture a live stream, we would create a VideoCapture object, passing the 0 argument to make it clear that we are using the default webcam. Next, we would execute a while loop to capture all the frames from the webcam until it closes.

```
df = pandas.DataFrame(columns = ["Start", "End"])
video = cv2.VideoCapture(0)
# Infinite while loop to treat stack of images as video
```

Step 3 :-

Reading The Frames And Converting Them To Grayscale

Utilize the above-mentioned object to read the frames and convert them into grayscale.

```
check, frame = video.read()
motion = 0
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Step 4 :-

Image Blurring

After Blurring The Image, Motion In Consecutive Frames Will Be Easily Detected After Removing The Noise From Each Frame, I.E Basically Removing High Frequencies From The Images.

```
gray = cv2.GaussianBlur(gray, (21, 21), 0)
```

Step 5 :-

Update Static Frame

Assign The First Frame From The Webcam To The Static Frame As No Changes To The Initial Level As There Are No Changes In The Initial Frame. This Frame Can Now Be Used To Calculate The Absolute Difference Between Consecutive Frames.

```
if static_back is None:
    static_back = gray
    continue
```

Step 6 :-

Calculate The Abs Difference In The Frames To Detect Motion

You can calculate the absolute difference between an initial frame (indicated by a static frame) and successive frames captured from the webcam. Whenever the difference exceeds 30, apply thresholding to grayscale so that all regions where

motion is detected would be white and diffract that threshold image.

```
diff_frame = cv2.absdiff(static_back, gray)

thresh_frame = cv2.threshold(diff_frame, 30, 255, cv2.THRESH_BINARY)[1]
thresh_frame = cv2.dilate(thresh_frame, None, iterations = 2)

cnts, _ = cv2.findContours(thresh_frame.copy(),
                           cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Step 7 :-

Finding Contours

After the Region Of Motion Has Been Converted To White Using Threshold, That Area Now Represents Contours, So Use Find Contours To Locate Coordinates Of Moving Objects. Afterwards, we would plot the rectangle (bounding box) on the coordinates of each moving object and display the text "Motion Detected" every time the object moves.

```
for contour in cnts:
    if cv2.contourArea(contour) < 10000:
        continue
    motion = 1
```

Step 8 :-

Displaying The Final Frames Live

As With All Motion Detection, The Next Step Is To Release The Camera And Destroy All The Windows After The Motion Has Been Successfully Detected. After That, Release The Camera And Destroy All Windows.

```
# if q entered whole process will stop
if key == ord('q'):
    if motion == 1:
        time.append(datetime.now())
    break
```

Step 9 :-

Storing the time period

An csv file stores the overall time between when an object entered our frame and left it.

```
# Creating a CSV file in which time of movements will be saved
df.to_csv("time_of_movements.csv")
video.release()

# Destroying all the windows
cv2.destroyAllWindows()
```

```
In [4]: import pandas as pd
df=pd.read_csv("Time_of_movements.csv")
df
```

```
Out[4]:
```

	Unnamed: 0	Start	End
0	0	2021-06-15 18:37:08.656264	2021-06-15 18:37:10.443298
1	1	2021-06-15 18:37:13.094642	2021-06-15 18:37:14.084620
2	2	2021-06-15 18:37:19.523672	2021-06-15 18:37:21.152031
3	3	2021-06-15 18:37:25.948502	2021-06-15 18:37:27.118209
4	4	2021-06-15 18:37:30.350538	2021-06-15 18:37:31.501566
5	5	2021-06-15 18:37:33.774248	2021-06-15 18:37:34.815540

Artivisto Windows:

CHAPTER 9 :-MODULE IMPLEMENTATION & EXPERIMENTAL RESULTS

MODULE IMPLEMENTATION & EXPERIMENTAL RESULTS

```
In [3]: # importing OpenCV, time and Pandas library
import cv2, time, pandas
from datetime import datetime

# Assigning our static_back to None
static_back = None
motion_list = [ None, None ]
time = []
df = pandas.DataFrame(columns = ["Start", "End"])
video = cv2.VideoCapture(0)

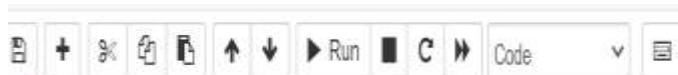
# Infinite while loop to treat stack of image as video
while True:
    check, frame = video.read()
    motion = 0
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
|
gray = cv2.GaussianBlur(gray, (21, 21), 0)

if static_back is None:
    static_back = gray
    continue

diff_frame = cv2.absdiff(static_back, gray)
```

Activate Windows
Go to Settings to activate Windows



```
thresh_frame = cv2.threshold(diff_frame, 30, 255, cv2.THRESH_BINARY)[1]
thresh_frame = cv2.dilate(thresh_frame, None, iterations = 2)

cnts,_ = cv2.findContours(thresh_frame.copy(),
                          cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for contour in cnts:
    if cv2.contourArea(contour) < 10000:
        continue
    motion = 1

    (x, y, w, h) = cv2.boundingRect(contour)
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

motion_list.append(motion)

motion_list = motion_list[-2:]
```

```
# Appending Start time of motion
if motion_list[-1] == 1 and motion_list[-2] == 0:
    time.append(datetime.now())

# Appending End time of motion
if motion_list[-1] == 0 and motion_list[-2] == 1:
    time.append(datetime.now())

# Displaying image in gray_scale
cv2.imshow("Gray Frame", gray)
cv2.imshow("Difference Frame", diff_frame)
cv2.imshow("Threshold Frame", thresh_frame)
cv2.imshow("Color Frame", frame)
```

Activate Window
Go to Settings to activate

This above pictures are the implementation of the project

Project Code

```
# importing the required OpenCV, time and Pandas library
import cv2, time, pandas
from datetime import datetime

# Assigning our static to None
static = None
motion = [ None, None ]
time = []
df = pandas.DataFrame(columns = ["Start", "End"])
video = cv2.VideoCapture(0)

# Infinite while loop to treat stack of image as video
while True:
    check, frame = video.read()
    motion = 0
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    gray = cv2.GaussianBlur(gray, (20, 20), 0)

    if static is None:
        static = gray
```

```

        continue

diff_frame = cv2.absdiff(static, gray)

thresh_frame = cv2.threshold(diff_frame, 30, 255,
cv2.THRESH_BINARY)[1]
thresh_frame = cv2.dilate(thresh_frame, None, iterations = 2)

cnts,_ = cv2.findContours(thresh_frame.copy(),
                        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for contour in cnts:
    if cv2.contourArea(contour) < 10000:
        continue
    motion = 1

    (x, y, w, h) = cv2.boundingRect(contour)
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

motion.append(motion)

motion = motion_list[-2:]

# Appending Start time of motion
if motion[-1] == 1 and motion[-2] == 0:

```

```

        time.append(datetime.now())

# Appending End time of motion
    if motion[-1] == 0 and motion[-2] == 1:
        time.append(datetime.now())

# Displaying image in gray_scale
    cv2.imshow("Gray Frame", gray)
    cv2.imshow("Difference Frame", diff_frame)
    cv2.imshow("Threshold Frame", thresh_frame)
    cv2.imshow("Color Frame", frame)
    key = cv2.waitKey(1)

# if q entered whole process will stop
    if key == ord('q'):
        if motion == 1:
            time.append(datetime.now())
        break

for i in range(0, len(time), 2):
    df = df.append({"Start":time[i], "End":time[i + 1]}, ignore_index = True)
df.to_csv("Time_movements.csv")
video.release() #releasing the video
cv2.destroyAllWindows()

```


CHAPTER 10 - OUTPUTS

Outputs

Analysis of all Windows :

When the code is run, there will be four new windows on the screen along with a CSV file.

1. Gray Frame:

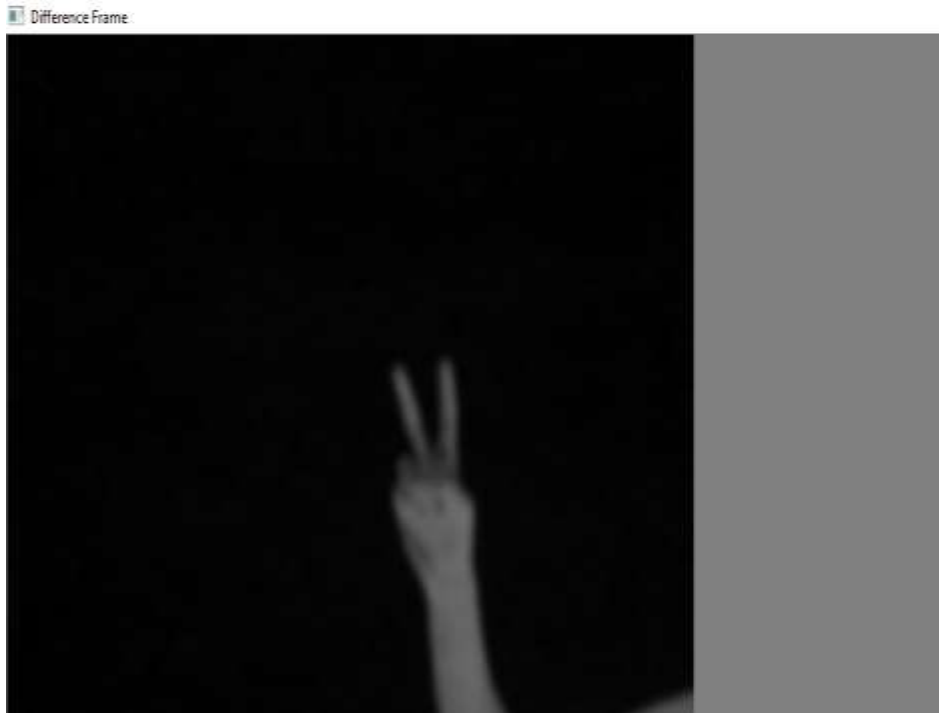
There is some blurring in grayscale because there is only one intensity value, while there are three different intensities in RGB (Red, Green, Blue). This will allow us to easily calculate the difference in intensity.

 Gray Frame



2. Difference Frame :

In difference frames, the intensities of the first-frame and the current-frame are shown as a difference.



3. Threshold Frame:

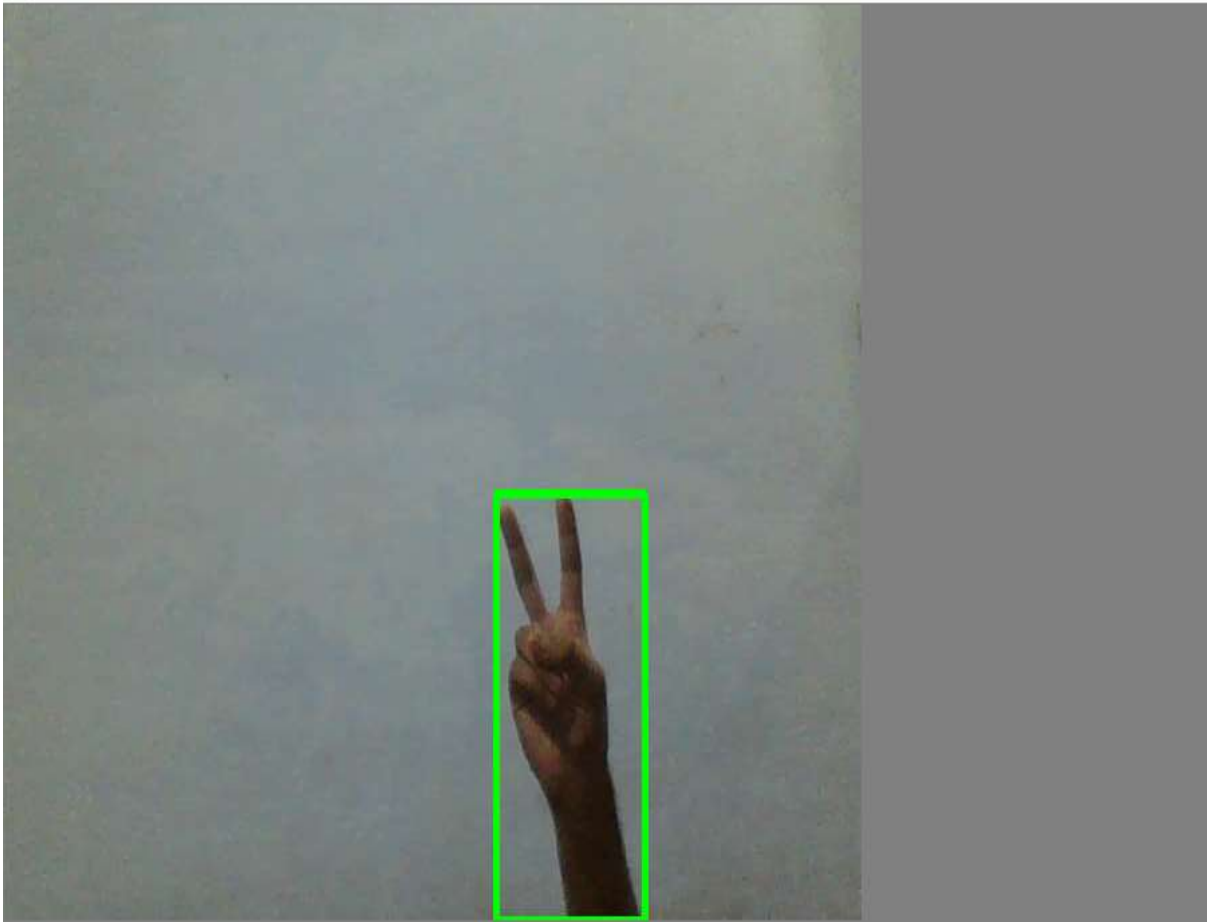
Those pixels that have an intensity difference of over 30(in my code) will be white, and those that have an intensity difference of less than 30 will be black.



4. Color Frame:

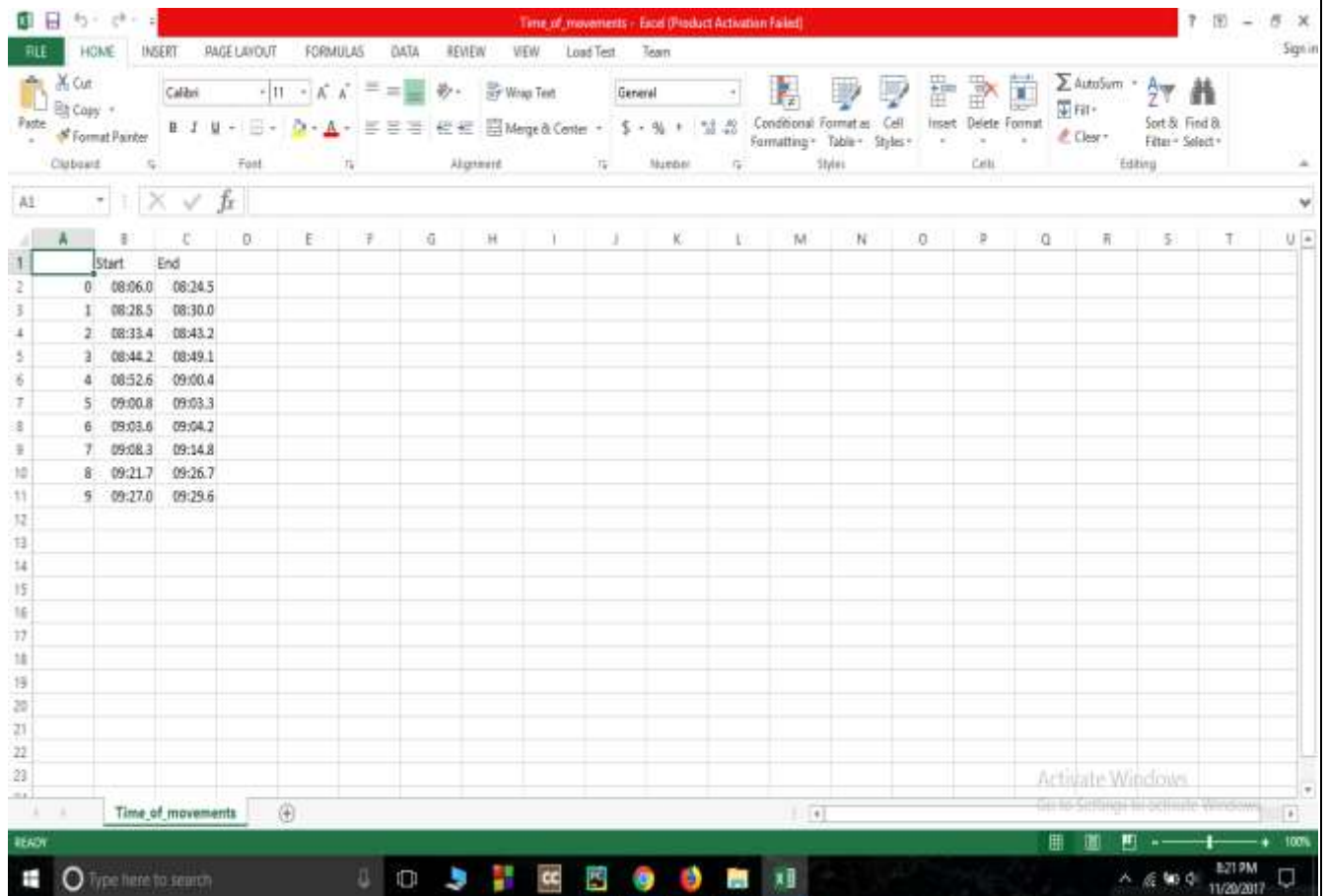
Color images with green contours around moving objects can be seen in this frame.

 Color Frame



5.CSV File :-In the Time_movements file, the start and end times of motion will be recorded. This file will be in the same folder where your code file is

located.



The screenshot shows a Microsoft Excel spreadsheet titled "Time_of_movements - Excel (Product Activation Failed)". The spreadsheet contains a table with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Start	End																		
2	0	08:06.0	08:24.5																		
3	1	08:28.5	08:30.0																		
4	2	08:33.4	08:43.2																		
5	3	08:44.2	08:49.1																		
6	4	08:52.6	09:00.4																		
7	5	09:00.8	09:03.3																		
8	6	09:03.6	09:04.2																		
9	7	09:08.3	09:14.8																		
10	8	09:21.7	09:26.7																		
11	9	09:27.0	09:29.6																		
12																					
13																					
14																					
15																					
16																					
17																					
18																					
19																					
20																					
21																					
22																					
23																					

PYTHON VS OTHER LANGUAGES FOR MOTION DETECTION

It has interfaces in a lot of programming environments, including Matlab, Octave, R, Python, C#, etc. Intel's OpenCV library has been developed in C++ and is accessible in a number of programming environments. Motion detection is a domain-specific application of machine learning prediction. When it comes to object detection, Python codes offer several benefits over other languages:

This programming language takes advantage of zero-based indexing.

- It provides dictionary (hash) support.
- It is simple and elegant, and it is open source.
- It is free and open..

CHAPTER 11 - VARIOUS OBJECT DETECTION ALGORITHMS IMPLEMENTED IN PYTHON

VARIOUS OBJECT DETECTION ALGORITHMS IMPLEMENTED IN PYTHON

As the object detection is also one of the application of this .lets see the different algorithms for it.

Haar-like features

Paul Viola and Michael Jones proposed this effective, machine learning-based method in 2001. A classifier is trained through the use of a great deal of images. Once this classifier has been trained, it detects objects within images. By first training a classifier with positive and negative images, this method introduces a cascade of concepts into the system . Grouping the features together instead of applying them all at once is a better method than applying them all at once. A window that fails the first stage should be discarded, but a window that passes all stages should be continued. The window that passes all stages is the desired region.

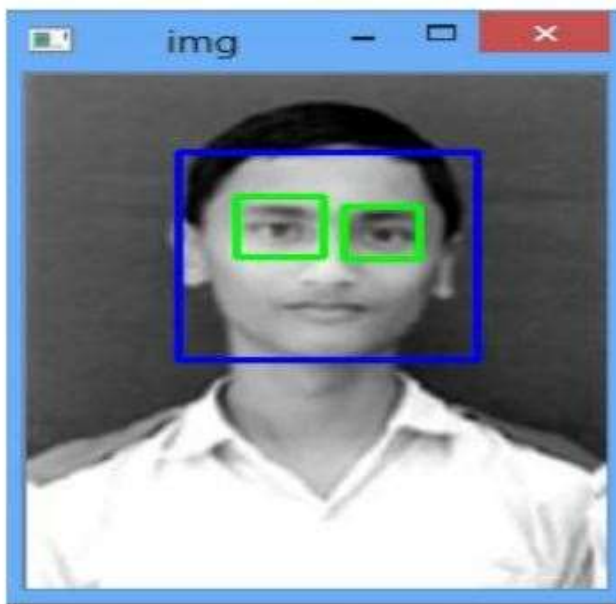
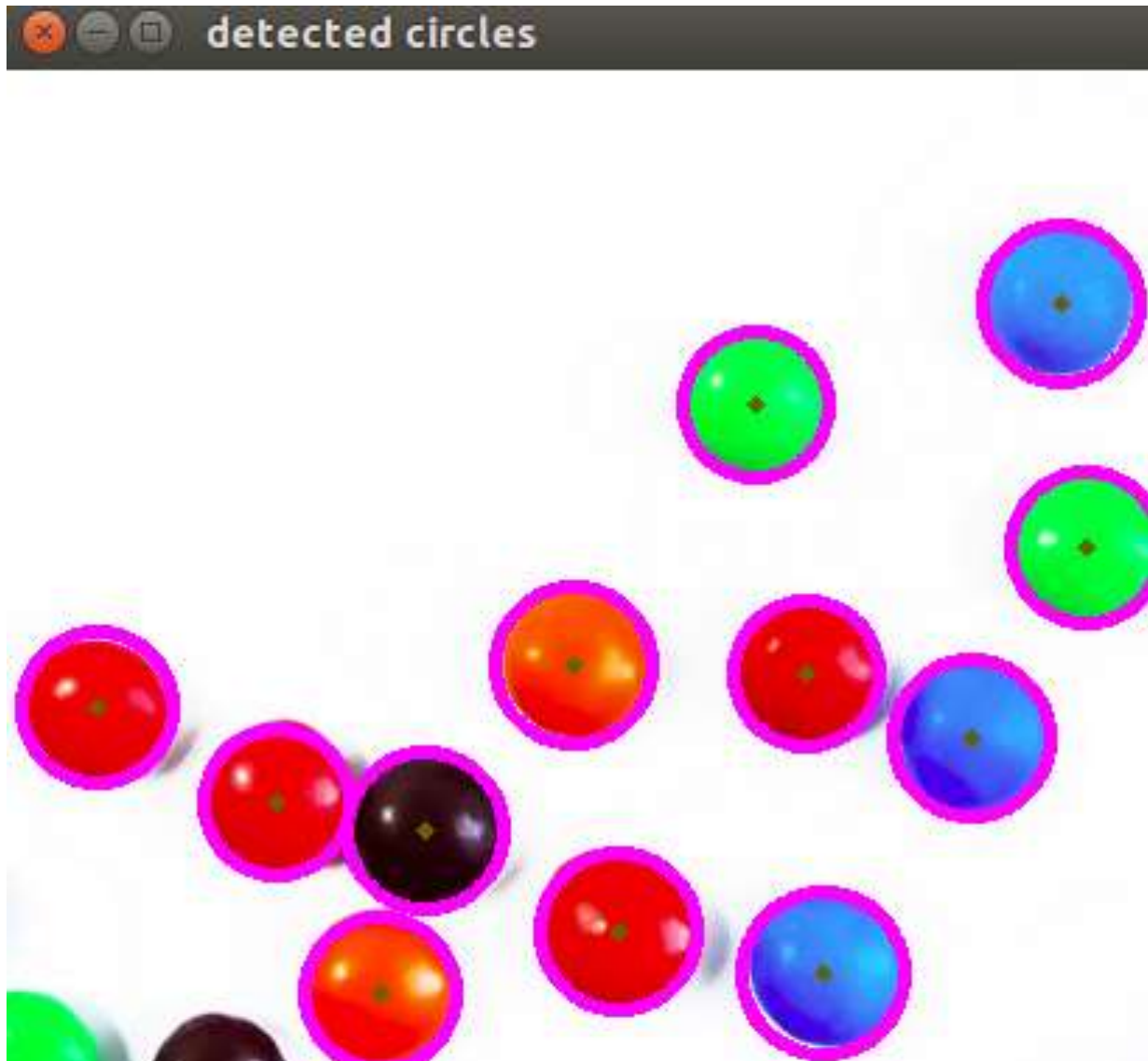


Fig:2 Face Detection using Haar Cascade

Circular Hough Transformation

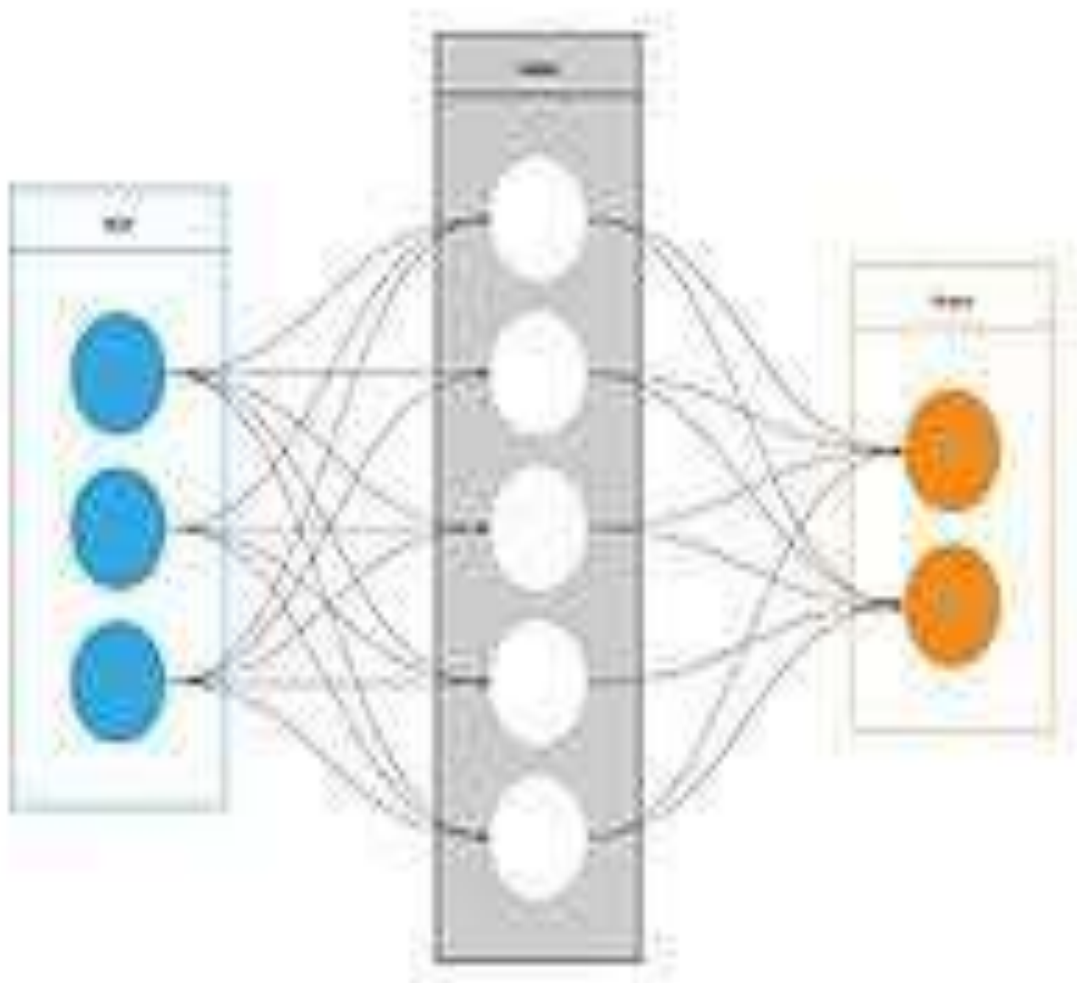
Developed in 1992 by Richard Duda and Peter Hart, the Hough transformation identifies arbitrary shapes in an image. It was afterwards adapted to recognize circular objects in noisy lowcontrast images, hence its marketed name Circular

Hough Transformation. A circle's center is computed in CHT using equations [2]: a and b are its coordinates, and r is its radius. Because CHT relies on three parameters, the computation takes more time and memory, along with an increase in error rates from the image. In order to simplify CHT programs, a constant radius value or a range of radius values are provided before they are run



Template matching

Objects that match the template image are detected in the source image by comparing it with the template image or patch. It is possible to use a feature-based approach if the template image has strong features, otherwise one could use the template-based approach.



Blob detection

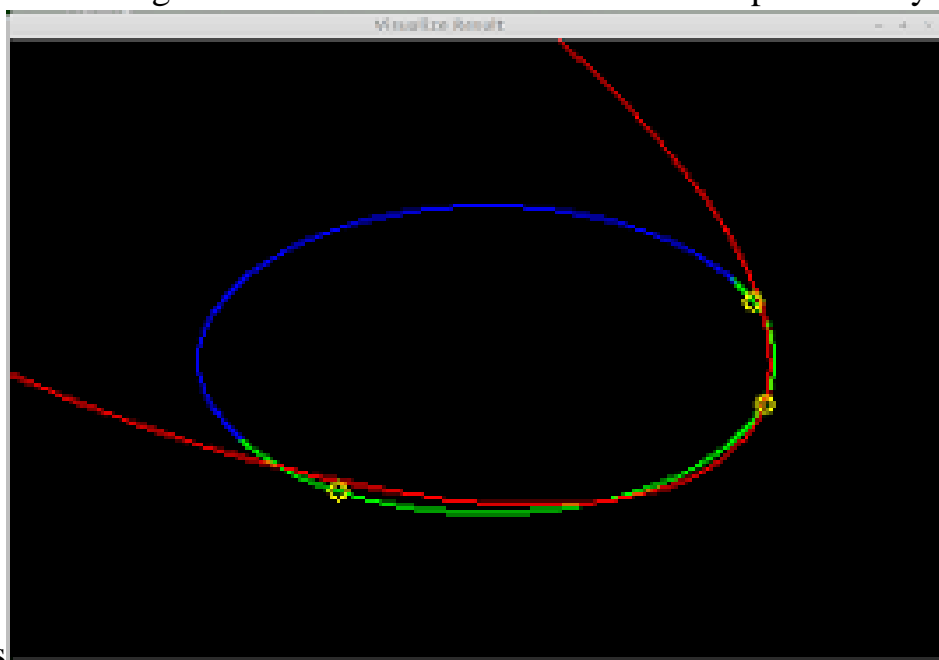
Blob detection is a popular technique for detecting areas of an image with different property values. In an image, a blob is a region in which all the points are mutually similar. There are two types of blob detection: local extreme and

differential methods.



The Gradient-based method

Using Partial derivatives spatially and temporally, gradient-based methods estimate the flow of image data at every point in an image. In the absence of a priori knowledge that the motion is limited to a limited set of potential values, the smoothing scale must be appropriate to the scale of the motion prior to determining its derivative. The method can be computationally costly due to

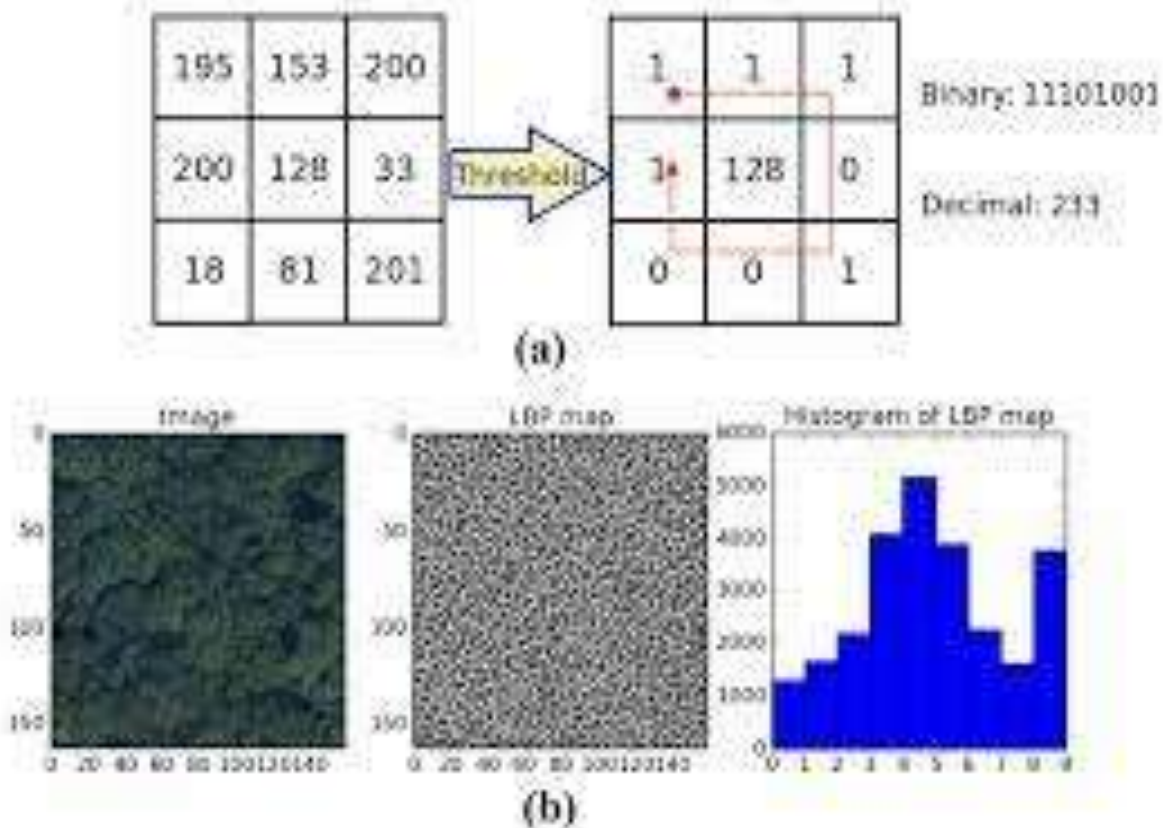


this

Local Binary Pattern

A visual descriptor called LBP was introduced in 1990. It's a strong texture classification approach. If one pixel is greater than its neighbor, write "0**", if the value of the pixel is higher than its neighbor, write "1".

- the aggregate histograms of all the cells are figured out and compared
- the normalized histograms are concatenated. With this feature vector, you can classify the images of the entire window.



Bag-of-words method

In image classification, features can be extracted from images. Images are treated as if they were written in words. In the visuals, these visual words are crucial. These significant points of the photographs are the characteristics of the images. Using this technique, you can classify images. The histograms are represented as frequency words through the creation of a vocabulary of many visual words.

Deep Face method

The Deep Face software developed by Google's AI research group simulates neurons by using neural networks as an approximation. Deep Learning neural networks are used in Deep Face to simulate deep

learning, which were developed in Menlo Park, California. Deep Face Learning involves the use of Machine Learning. It is the process of finding recurring faces in a large body of data that leads to a high-level abstraction.



CHAPTER 12 - CONCLUSION

CONCLUSION

- OpenCV has a wide range of applications, and this project has been designed to cover one of them. Detect motion by subtracting frames from the live streaming, where each new frame will be subtracted from the very first frame.
- The frames will show motion detection if any region has an image smoothing value less than a specific threshold. Use contours later to plot the bounding boxes on the region where motion has been detected.

APPLICATIONS AND FUTURE SCOPE

In computer vision, we are still at the early stages of the field; it hasn't yet reached maturity to be used directly to solve real-world problems. Within a few years, computer vision and especially the ability to detect objects will no longer be futuristic and will become ubiquitous.

As a sub-branches of machine learning, we can currently consider motion detection of objects as follows:

Security

It can also be used for personal security purposes to identify movements of suspicious or unwanted objects in a given area. These techniques are used for detecting thieves as well as identifying thieves.

Tracking

Used to track the motion of a person and counts the screen time of an object or person present in front of the frame.

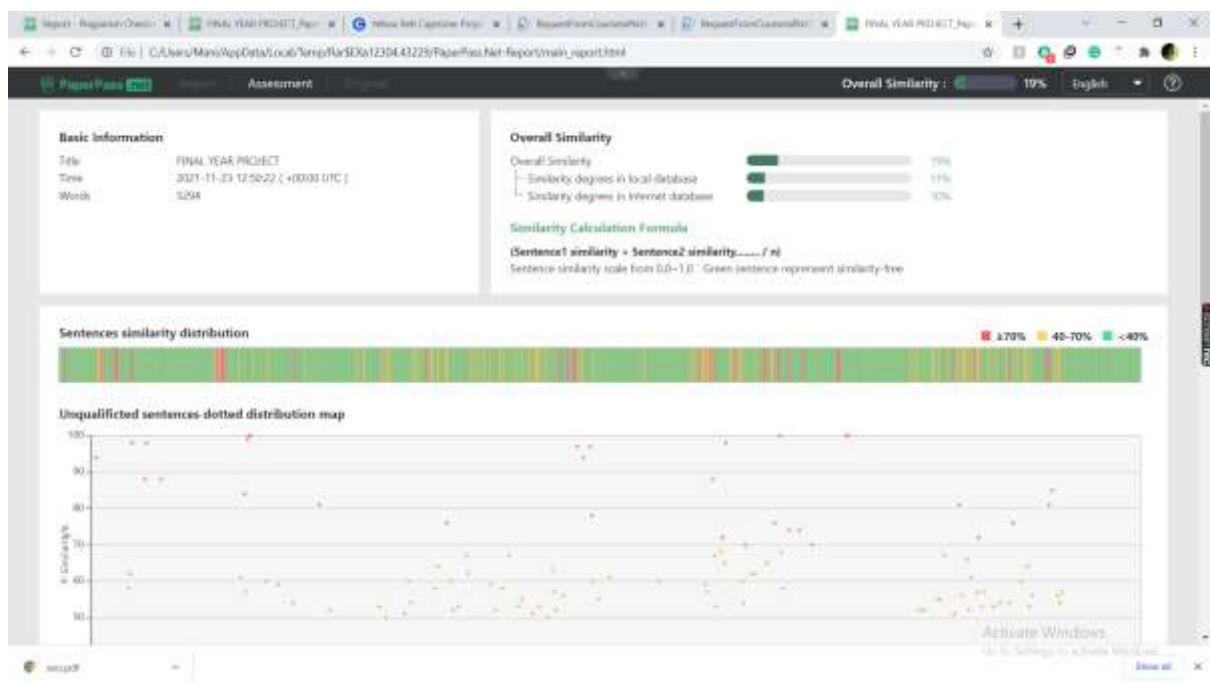
CHAPTER 13 - REFERENCES

REFERENCES

- [1] P. Rajkumar, B. Tech., V.P. Gladis Pusupa Rathi, “Motion Detection Based Interactive Surveillance Systems for Mobile Clients”, 2011 International Conference on Information and Network Technology, IACSIT Press, Singapore.
- [2] R.Collins, A.Lipton, H.Fujiyoshi, and T.Kanade, “Algorithms for cooperative multisensor surveillance,” in the IEEE Conf., vol. 89, pp.1456-1477, Oct. 2001.
- [3] F.W. Mounts. A video encoding system with conditional picture-element replenishment. Bell Systems Technical Journal, 48, no. 7:2545–2554, September 1969.
- [4] B.Gelbord and G.Roelofsen, “New surveillance techniques raise privacy concerns,” Com. ACM, vol. 45, no. 11, pp.
- [5] 23-24, Nov. 2002. [6] T.Kanade, R.Collins, and A.Lipton, “Advances in cooperative multi-sensor video surveillance,” in DARPA Image Understanding Workshop, pp. 3–24, Nov. 1998. ISO/IEC 15444-1. JPEG 2000 image coding system, 2000.
- [7] D. Taubman. High performance scalable image compression with EBCOT. IEEE Transactions on Image Processing, 9(7):1158– 1170, July 2000.
- [8] R.Collins, A.Lipton, H.Fujiyoshi, and T. Kanade, “Algorithms for cooperative multisensor surveillance,” in the IEEE Conf., vol. 89, pp. 1456-1477, Oct. 2001.
- [9] P. Rajkumar, “Motion Detection Based Interactive Surveillance Systems for Mobile Clients” B.Tech. PG , Dept. of CSE Sudharsan Engineering College,

CHAPTER 14 – PLAGRISM CHECK

PAPER PRESS :-



TURNITIN :-

