

FACIAL RECOGNITION

**An Internship training Report submitted in partial fulfillment of the requirements
for the award of the degree of**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

PAVULURI YASWANTH CHOWDARY (1210315539)

Under the esteemed guidance of

Mr.Rajneesh Vetukuri, Head - IOT Business

Clove Technologies Pvt Ltd



DEPT. OF COMPUTER SCIENCE & ENGINEERING

GITAM UNIVERSITY

(Declared as deemed to be university u/s3 of the UGC Act 1956)

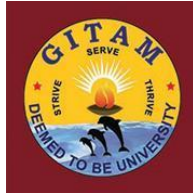
VISAKHAPATNAM, A.P, INDIA.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)



DECLARATION

We, hereby declare that the internship review entitled “**FACIAL RECOGNITION**” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of Btech. in Computer Science and Engineering.

The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration Number:

1210315539

Name:

P.Yaswanth Chowdary

Signature



Offer of Internship

This is to confirm that Clove Technologies Pvt Ltd is offering an internship, for the below candidate with their selected project topic.

Name of the Candidate	Reg No	Selected Topic
Pavuluri Yaswanth Chowdary	1210315539	Facial Recognition and Gait Analysis

The duration of internship would be 40 Days starting from 30/04/2018 to 08/06/2018. We shall provide them with Practical on job training and enabling them to put their theoretical subjects into practical usage. Upon successful completion of the project, the candidate is to submit his thesis/ report to the management and only then they would be provided with the relevant project certificate.

We believe that we can provide you with an interesting and rewarding internship, and look forward to you join with us.

Best Regards,

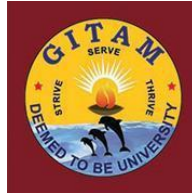
Balaji Janyavula

Balaji Janyavula
HR Manager



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM
INSTITUTE OF TECHNOLOGY GITAM

(Deemed to be University)



CERTIFICATE

This is to certify that the internship report entitled —**FACIAL RECOGNITION IN IMAGE PROCESSING** is a bonafide record of work carried out by **P.YASWANTH CHOWDARY (1210315539)** student submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

SUPERVISOR

Mr. Pramodh
Product Engineer, Clove Technologies

INTERNSHIP REVIEWER

Mr.Rajneesh Vetukuri
Head-IOT Business

Acknowledgement

The internship opportunity I had with Clove Technologies Pvt Limited is a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the Head of IOT Mr. Rajneesh Vetukuri and Product Engineer Mr.Pramodh of Clove Technologies to guide and allowing me to carry out my project at their esteemed organization.

I express my deepest thanks to Mr. Pramodh [Product Engineer] for taking part in useful decision & giving necessary advises and guidance throughout the duration of my project and to all the staff members of Clove Technologies for their careful and precious guidance which were extremely valuable for my study both theoretically and practically.

I deeply express my sincere thanks to our Head of Department (HOD) Prof. K. Thammi Reddy for encouraging and allowing us to do Software industrial exposure and thanks to all my lecturers who have directly or indirectly helped my project.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives

TABLE OF CONTENTS

- 1. Abstract**
- 2. Company Profile**
- 3. Schedule of the Internship**
- 4. Internship Activities**
 - a. OPENCV and Other Tools installation**
 - b. Functions of OpenCV**
 - c. Face Detection and its classifiers**
 - 1. Haar classifier**
 - 2. LBHP classifier**
 - d. Face Recognition**
 - e. Object Tracking**
 - f. Pedestrian Tracking**
 - g. Virtual Fencing**
 - h. MQTT**
 - i. BRESENHAMS ALGORITHM**
 - j. KAAFKA and Zookeeper**
- 5. Assessment of Internship**
- 6. References**



CERTIFICATE

This is to certify that **Mr. Pavuluri Yaswanth Chowdary** student of Computer Science from Gitam University, Vishakhapatnam has undergone project work on “**Facial Recognition and Gait Analysis**” with “A” Grade at Clove Technologies Pvt. Ltd. Hyderabad under the supervision and guidance of Mr. Rajneesh (Head – IOT).

Mr. Pavuluri Yaswanth Chowdary has successfully completed his Internship and submitted the Internship project report. During the period of Internship he was found sincere, punctual and regular. His conduct and behavior was good.

The details of Internship are as follows.

The period of Internship: 40 Days

No. of days present: 40 Days

Certificate issued on: 08- JUNE-2018

Balaji Janyavula

HR Manager





COMPANY PROFILE

Empowering engineers across the world to build operate and maintain intelligent infrastructure by accelerating technology breakthroughs in Digital Engineering. We solve complex problems of Infrastructure Industry with our innovative services of BIM, Geospatial, Geomatics and Technology.

Since our inception in 2004, with a team of 350 and counting, we are serving customers from various geographical locations including Europe, USA, Australia, Nordic Region, India, UK, and UAE.

We were acknowledged by Industry for consistent innovation in processes and solutions - 'Excellence Award' by Institute of Economic Studies (IES), '25 Most Promising GIS Solution Providers in APAC Region' by CIO Outlook Magazine and many more.

Our way forward is to Accelerate: Digital Construction, Digital Mapping, Digital Measurement, and Digital Transformation to reduce cost and increase productivity across Infrastructure Industry.

ABSTRACT

Project work is one of the important activities of engineering curriculum of every level irrespective of degree or diploma. This provides us an opportunity to apply our knowledge, skill, and aptitude in real life. Project plan should be well trained and have knowledge about the project. In CLOVE TECHNOLOGIES, besides many projects of client like aerial Surveying and Web designing, I am involved in Image Processing (Facial Recognition and Obstacle Detection) .I worked in FACIAL RECOGNITION, which is a real time project for the company to be submitted to client under specific guidelines

I worked in the IOT team of the company as a Assistant Product Engineer under the guidance of Mr.Pramodh and Mr. Rajneesh who helped us through the tool to be used in image processing and how to overcome certain difficulties in Facial Recognition. As a team we present a real time and High Precision Face Recognition system using OPENCV tool with PYTHON as programming language. It can run on Middle as well as Low level with certain additional requirements. The project would allow user access to a particular machine based on an in-depth analysis of a person's facial features. This Application will be developed using Intel's open source computer vision project, OpenCV

Experimental results show the system operates at 30 frames/sec and a recognition rate is 55.59% when a threshold value is set to 0.45%, performance comparable to that of a state-of-the-art system.

INTERNSHIP ACTIVITIES

❖ OPENCV:

INTRODUCTION:

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, IOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Here we are mostly focused on OpenCV with Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

➤ INSTALLING OPENCV IN WINDOWS 10

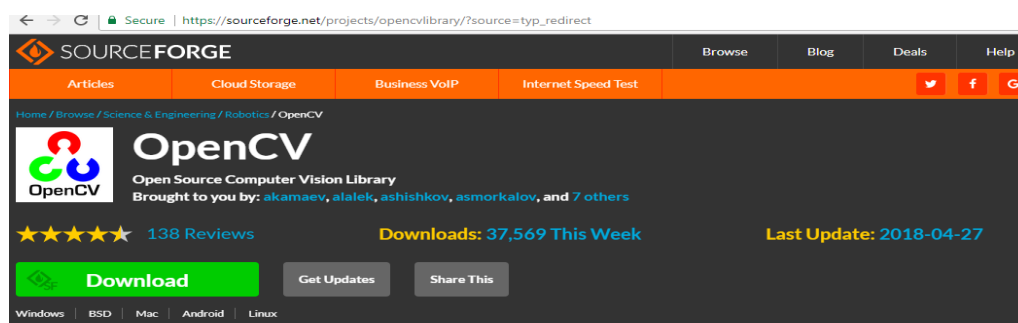
Installing OpenCV from prebuilt binaries

Step 1: These python packages are to be downloaded to their default locations.

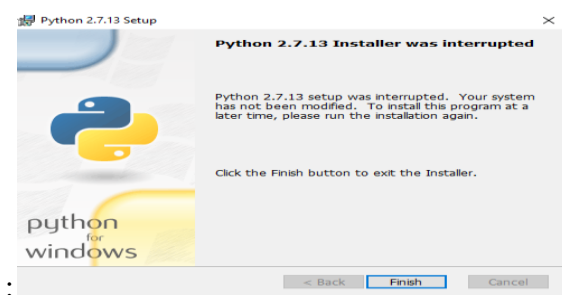
a) Python 2.7.x

b) Numpy

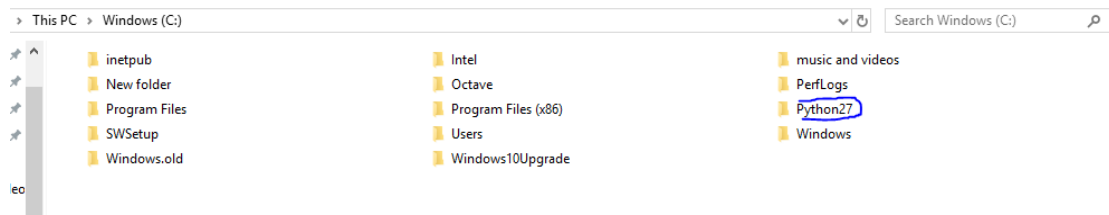
OPENCV from SOURCEFORGE-INSTALLATION:



Python 2.7.x installation:



Step 2 : Installing packages to their default locations where python27 will be placed in **C:/Python27/**.

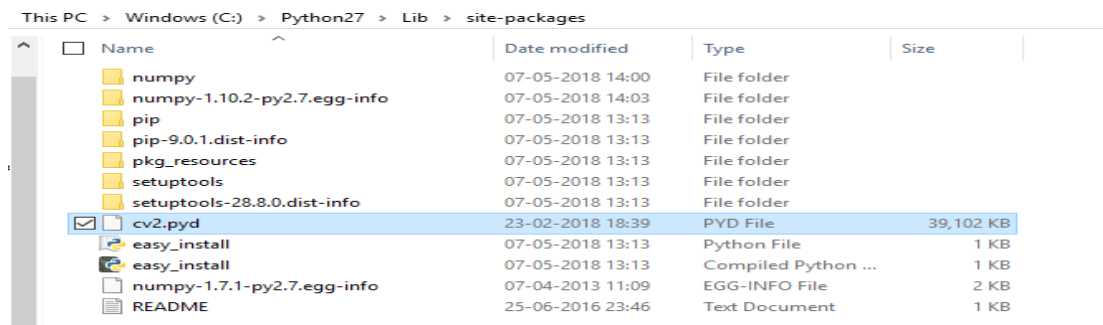


Step 3: After installation, open Python IDLE. Enter import Numpy and make sure Numpy is working fine.

Step 4: Download latest OpenCV release from [sourceforge site](#) and double-click to extract it.

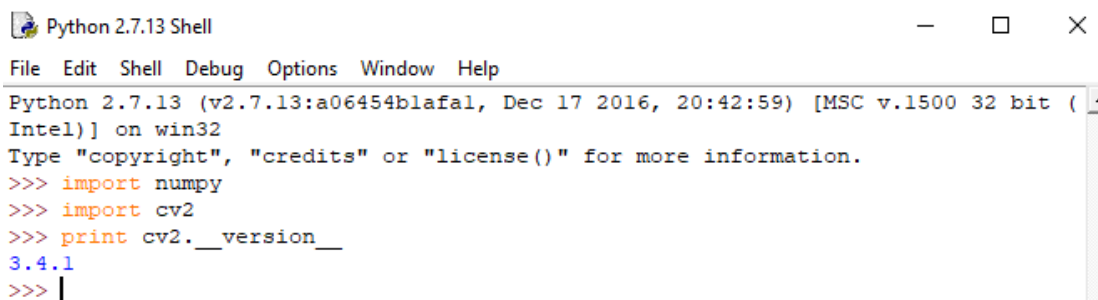
Step 5: Go to **OpenCV/build/python/2.7** folder.

Step 6: Copy **cv2.pyd** to **C:/Python27/lib/site-packages**.



This task is being performed to see that we are using 32-bit binaries of Python packages, But if you want to use OpenCV for x64 bit-binaries of NUMPY. We don't have official 64 bit-binaries of NUMPY. When you start Python IDLE, It shows the compiler details.

Step 7: Open Python IDLE and type following codes in Python terminal.



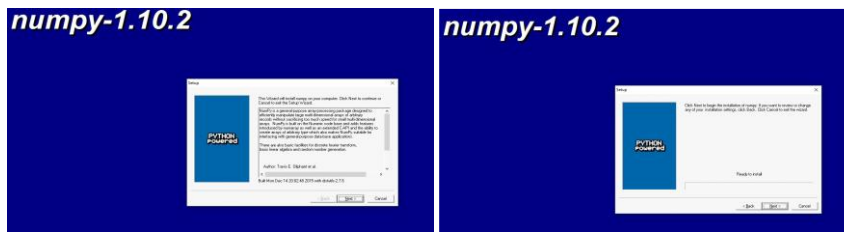
If there

seems to be no error, then we installed python27 correctly.

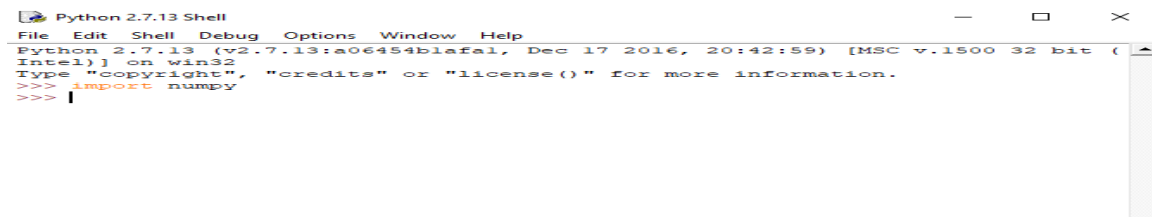
NUMPY:

Numpy makes the task easier. **Numpy** is a highly optimized library for numerical operations. It gives MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV. **Numpy is must to write optimized codes in OpenCV-Python.**

INSTALLATION:



Open Python IDLE and type import Numpy and make sure that no error is occurred

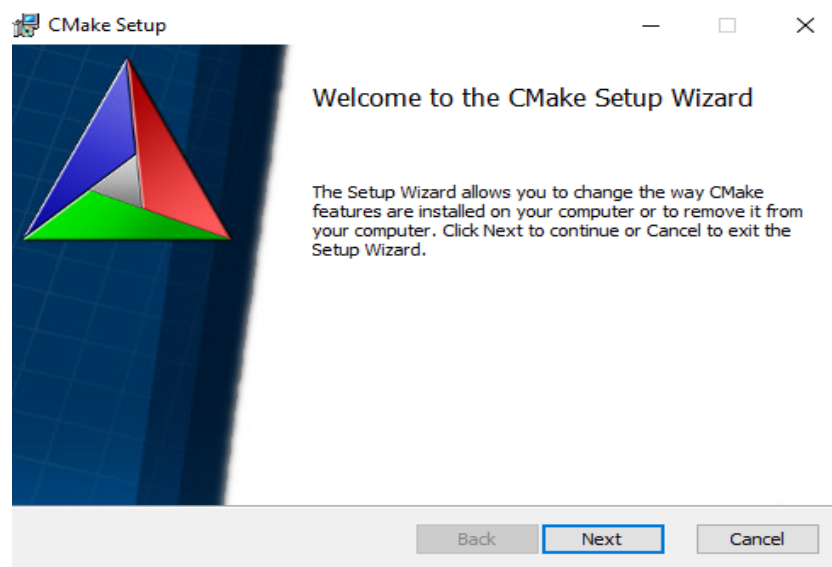
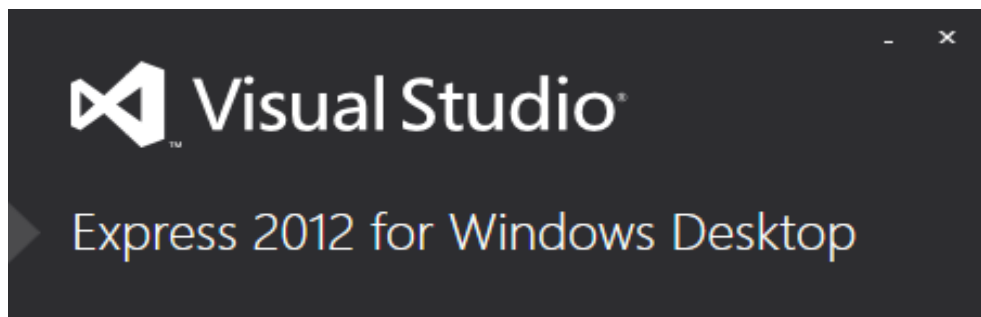


Building OpenCV from source:

Step 1: Download and install Visual Studio and CMake.

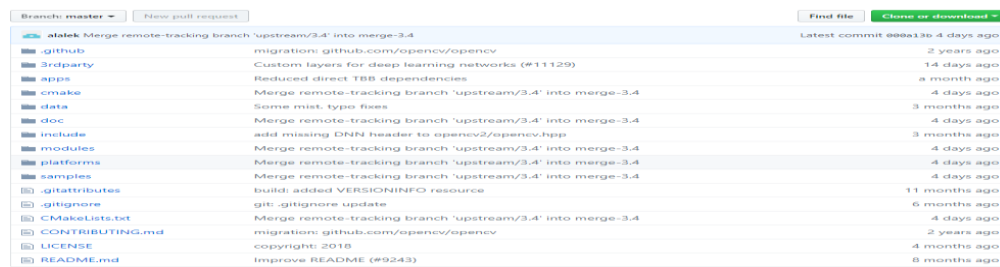
A) Visual Studio 2012

B) CMake(3.5 or above only)



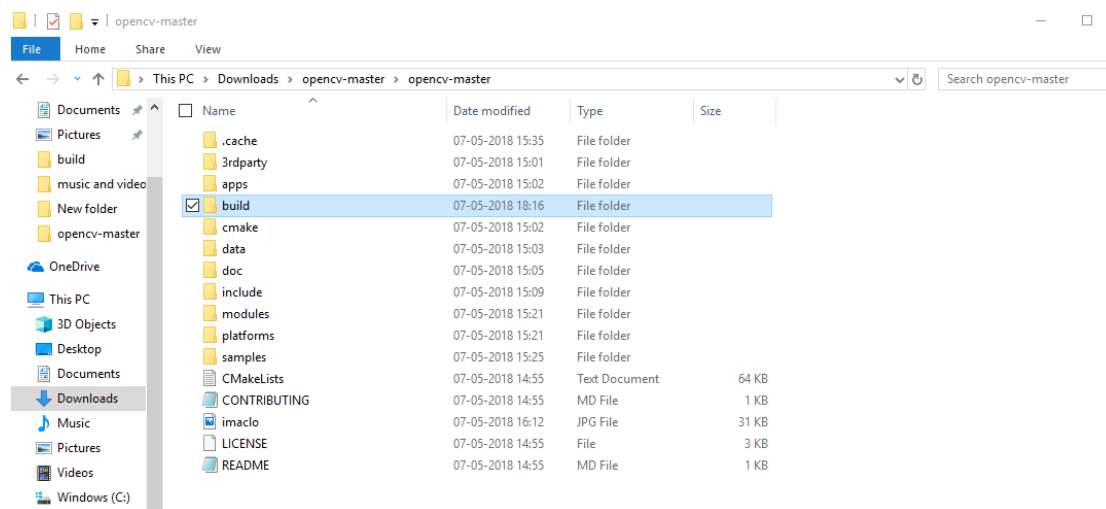
Step 2: Make sure NUMPY and Python are working fine enough without any errors

Step 3:Download OpenCV source. It can be from [Github](#) (for latest source)



File	Commit	Time
alek Merge remote-tracking branch 'upstream/3.4' into merge-3.4	Latest commit 000a33b	4 days ago
github migration: github.com/opencv/opencv		2 years ago
3rdparty Custom layers for deep learning networks (#11129)		14 days ago
apps Reduced direct TBB dependencies		a month ago
cmake Merge remote-tracking branch 'upstream/3.4' into merge-3.4		4 days ago
data Some mist. type fixes		3 months ago
doc Merge remote-tracking branch 'upstream/3.4' into merge-3.4		4 days ago
include add missing DNN header to opencv2/opencv.hpp		3 months ago
modules Merge remote-tracking branch 'upstream/3.4' into merge-3.4		4 days ago
platforms Merge remote-tracking branch 'upstream/3.4' into merge-3.4		4 days ago
samples Merge remote-tracking branch 'upstream/3.4' into merge-3.4		4 days ago
.gitattributes build: added VERSIONINFO resource		11 months ago
.gitignore git: .gitignore update		6 months ago
CMakeLists.txt Merge remote-tracking branch 'upstream/3.4' into merge-3.4		4 days ago
CONTRIBUTING.md migration: github.com/opencv/opencv		2 years ago
LICENSE copyright: 2018		4 months ago
README.md Improve README (#9243)		6 months ago

Extract it and place a new folder of with name build in it.



Step 4: Open CMake-gui (*Start > All Programs > CMake-gui*)

Step 5: Fill the fields.

Click on **Browse Source** and locate the OpenCV-master folder.

Click on **Browse build** and locate the build folder we created.

Click on **Configure**.

Step 6: It will open a new window to select the compiler. Choose appropriate compiler (here, Visual Studio 11) and click **Finish**. Wait until analysis is finished.

Step 7: You will see all the fields are marked in red. Click on “WITH”

WITH_MSMP	<input type="checkbox"/>	WITH_MSMP	<input type="checkbox"/>
WITH_NVCUVID	<input type="checkbox"/>	WITH_NVCUVID	<input type="checkbox"/>
WITH_OPENCCL	<input type="checkbox"/>	WITH_OPENCCL	<input type="checkbox"/>
WITH_OPENCCLAMDBLAS	<input type="checkbox"/>	WITH_OPENCCLAMDBLAS	<input type="checkbox"/>
WITH_OPENCCLAMDFFT	<input type="checkbox"/>	WITH_OPENCCLAMDFFT	<input type="checkbox"/>
WITH_OPENCCL_SVM	<input type="checkbox"/>	WITH_OPENCCL_SVM	<input type="checkbox"/>
WITH_OPENEXR	<input type="checkbox"/>	WITH_OPENEXR	<input type="checkbox"/>
WITH_OPENGL	<input type="checkbox"/>	WITH_OPENGL	<input type="checkbox"/>
WITH_OPENMP	<input type="checkbox"/>	WITH_OPENMP	<input type="checkbox"/>
WITH_OPENNI	<input type="checkbox"/>	WITH_OPENNI	<input type="checkbox"/>
WITH_OPENNI2	<input type="checkbox"/>	WITH_OPENNI2	<input type="checkbox"/>
WITH_OPENVX	<input type="checkbox"/>	WITH_OPENVX	<input type="checkbox"/>
WITH_PNG	<input type="checkbox"/>	WITH_PNG	<input type="checkbox"/>
WITH_PROTOBUF	<input type="checkbox"/>	WITH_PROTOBUF	<input type="checkbox"/>
WITH_PVAPI	<input type="checkbox"/>	WITH_PVAPI	<input type="checkbox"/>
WITH_QT	<input type="checkbox"/>	WITH_QT	<input type="checkbox"/>
WITH_TBB	<input type="checkbox"/>	WITH_TBB	<input type="checkbox"/>
WITH_TIFF	<input type="checkbox"/>	WITH_TIFF	<input type="checkbox"/>
WITH_VFW	<input type="checkbox"/>	WITH_VFW	<input type="checkbox"/>
WITH_VTK	<input type="checkbox"/>	WITH_VTK	<input type="checkbox"/>
WITH_WEBP	<input type="checkbox"/>	WITH_WEBP	<input type="checkbox"/>
WITH_WIN32UI	<input type="checkbox"/>	WITH_WIN32UI	<input type="checkbox"/>
WITH_XIMEA	<input type="checkbox"/>	WITH_XIMEA	<input type="checkbox"/>

Step 8: Now click on **BUILD** field to expand it. First few fields configure the build method.

BUILD_CUDA_STUBS	<input type="checkbox"/>	BUILD_opencv_calib3d	<input checked="" type="checkbox"/>
BUILD_DOCS	<input type="checkbox"/>	BUILD_opencv_core	<input checked="" type="checkbox"/>
BUILD_EXAMPLES	<input type="checkbox"/>	BUILD_opencv_dnn	<input checked="" type="checkbox"/>
BUILD_IPP_IW	<input type="checkbox"/>	BUILD_opencv_features2d	<input checked="" type="checkbox"/>
BUILD_ITT	<input type="checkbox"/>	BUILD_opencv_flann	<input checked="" type="checkbox"/>
BUILD_JASPER	<input type="checkbox"/>	BUILD_opencv_highgui	<input checked="" type="checkbox"/>
BUILD_JAVA	<input checked="" type="checkbox"/>	BUILD_opencv_imgcodecs	<input checked="" type="checkbox"/>
BUILD_JPEG	<input checked="" type="checkbox"/>	BUILD_opencv_imgproc	<input checked="" type="checkbox"/>
BUILD_LIST	<input type="checkbox"/>	BUILD_opencv_java_bindings_generator	<input checked="" type="checkbox"/>
BUILD_OPENEXR	<input checked="" type="checkbox"/>	BUILD_opencv_js	<input type="checkbox"/>
BUILD_PACKAGE	<input type="checkbox"/>	BUILD_opencv_ml	<input checked="" type="checkbox"/>
BUILD_PERF_TESTS	<input type="checkbox"/>	BUILD_opencv_objdetect	<input checked="" type="checkbox"/>
BUILD_PNG	<input checked="" type="checkbox"/>	BUILD_opencv_photo	<input checked="" type="checkbox"/>
BUILD_PROTOBUF	<input checked="" type="checkbox"/>	BUILD_opencv_python_bindings_generator	<input checked="" type="checkbox"/>
BUILD_SHARED_LIBS	<input checked="" type="checkbox"/>	BUILD_opencv_shape	<input checked="" type="checkbox"/>
BUILD_TBB	<input type="checkbox"/>	BUILD_opencv_stitching	<input checked="" type="checkbox"/>
BUILD_TESTS	<input checked="" type="checkbox"/>	BUILD_opencv_superres	<input checked="" type="checkbox"/>
BUILD_TIFF	<input checked="" type="checkbox"/>	BUILD_opencv_ts	<input checked="" type="checkbox"/>
BUILD_USE_SYMLINKS	<input type="checkbox"/>	BUILD_opencv_video	<input checked="" type="checkbox"/>
BUILD_WEBP	<input checked="" type="checkbox"/>	BUILD_opencv_videoio	<input checked="" type="checkbox"/>
BUILD_WITH_DEBUG_INFO	<input checked="" type="checkbox"/>	BUILD_opencv_videostab	<input checked="" type="checkbox"/>
BUILD_WITH_DYNAMIC_IPP	<input checked="" type="checkbox"/>	BUILD_opencv_world	<input type="checkbox"/>
BUILD_WITH_STATIC_CRT	<input checked="" type="checkbox"/>		
BUILD_ZLIB	<input checked="" type="checkbox"/>		

Step 9: Now we Click on **ENABLE** field.

EIGEN_INCLUDE_PATH		EIGEN_INCLUDE_PATH-NOTFOUND
ENABLE_BUILD_HARDENING	<input type="checkbox"/>	
ENABLE_CCACHE	<input type="checkbox"/>	
ENABLE_IMPL_COLLECTION	<input type="checkbox"/>	
ENABLE_INSTRUMENTATION	<input type="checkbox"/>	
ENABLE_LTO	<input type="checkbox"/>	
ENABLE_NOISY_WARNINGS	<input type="checkbox"/>	
ENABLE_PIC	<input checked="" type="checkbox"/>	
ENABLE_PRECOMPILED_HEADERS	<input checked="" type="checkbox"/>	
ENABLE_PYLINT	<input type="checkbox"/>	
ENABLE_SOLUTION_FOLDERS	<input checked="" type="checkbox"/>	
EXECUTABLE_OUTPUT_PATH		C:/Users/Ramsee/Downloads/opencv-master/opencv-master/build/bin

Step 10: Also make sure that in the **PYTHON** field, everything is filled. (Ignore PYTHON_DEBUG_LIBRARY).

PROTOBUF_UPDATE_FILES	<input type="checkbox"/>	
PYTHON2_EXECUTABLE		C:/Python27/python.exe
PYTHON2_INCLUDE_DIR		C:/Python27/include
PYTHON2_INCLUDE_DIR2		
PYTHON2_LIBRARY		C:/Python27/libs/python27.lib
PYTHON2_LIBRARY_DEBUG		
PYTHON2_NUMPY_INCLUDE_DIRS		C:/Python27/lib/site-packages/numpy/core/include
PYTHON2_PACKAGES_PATH		C:/Python27/Lib/site-packages
PYTHON3_EXECUTABLE		
PYTHON3_INCLUDE_DIR		
PYTHON3_INCLUDE_DIR2		
PYTHON3_LIBRARY		
PYTHON3_LIBRARY_DEBUG		
PYTHON3_NUMPY_INCLUDE_DIRS		
PYTHON3_PACKAGES_PATH		

Step 11: Finally click the **Generate** button.

Step 12: Now go to our **OpenCV/build** folder. There you will find **OpenCV.sln** file. Open it with Visual Studio.

Step 13: Check build mode as **Release** instead of **Debug**.

Step 14: In the solution explorer, right-click on the **Solution** (or **ALL_BUILD**) and build it. It will take some time to finish.

Step 15: Again, right-click on **INSTALL** and build it. Now OpenCV-Python will be installed.

Step 16: Open Python IDLE and enter import cv2. If no error, it is installed correctly.

Reading and Writing an IMAGE in PYTHON

READING: Use the function `cv2.imread ()` to read an image. The image should be in the working directory or a full path of image should be given.

We can simply pass integers -1,0,1 respectively for transparency of image will be neglected, image will be loaded in greyscale and load image as alpha channel respectively

WRITING: Use the function `cv2.imwrite ()` to save an image.

First argument is the filename; second argument is the image you want to save. This will save the image in working directory

```
>>> img=cv2.imread('C:/imaclo.jpg',0)
>>> print img
[[255 255 255 ..., 144 149 170]
 [255 255 255 ..., 137 139 187]
 [255 255 255 ..., 138 137 221]
 ...,
 [255 255 255 ..., 255 254 254]
 [255 255 255 ..., 255 255 253]
 [255 255 255 ..., 250 248 255]]
>>> cv2.imwrite('ya.jpg',img)
True
>>> cv2.imwrite('yash.jpg',img)
True
```

Imaclo.jpg:



Yash.jpg



Capture Video from Camera

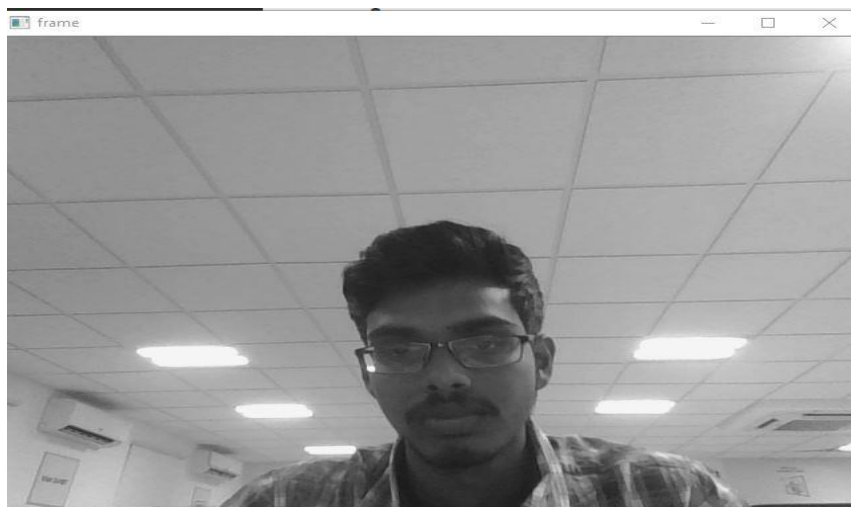
We need to create a **VideoCapture ()** object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected. So I simply pass 0 (or -1).

```
>>> import numpy
>>> import cv2
>>> cap=cv2.VideoCapture(0)
>>> while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
Traceback (most recent call last):
  File "<pyshell#3>", line 3, in <module>
    ret, frame = cap.read()
KeyboardInterrupt
>>> cap.release()
>>> cv2.destroyAllWindows()
```



❖ DRAWING FUNCTIONS IN OPENCV:

Drawing Line

To draw a line, you need to pass starting and ending coordinates of line. We will create a black image and draw a blue line on it from top-left to bottom-right corners.

Drawing Rectangle

To draw a rectangle, you need top-left corner and bottom-right corner of rectangle. This time we will draw a green rectangle at the top-right corner of image.

Drawing Circle

To draw a circle, you need its center coordinates and radius. A circle inside the rectangle drawn

Drawing Ellipse

To draw the ellipse, we need to pass several arguments. One argument is the center location (x,y). Next argument is axes lengths (major axis length, minor axis length). angle is the angle of rotation of ellipse in anti-clockwise direction. StartAngle and endAngle denotes the starting and ending of ellipse arc measured in clockwise direction from major axis. i.e. giving values 0 and 360 gives the full ellipse. For more details, check the documentation of [cv2.ellipse\(\)](#). Below example draws a half ellipse at the center of the image.

Drawing Polygon

To draw a polygon, first you need coordinates of vertices. Make those points into an array of shape ROWS. Here it is small polygon of with four vertices in yellow color.

Adding Text to Images:

To put texts in images, you need specify following things.

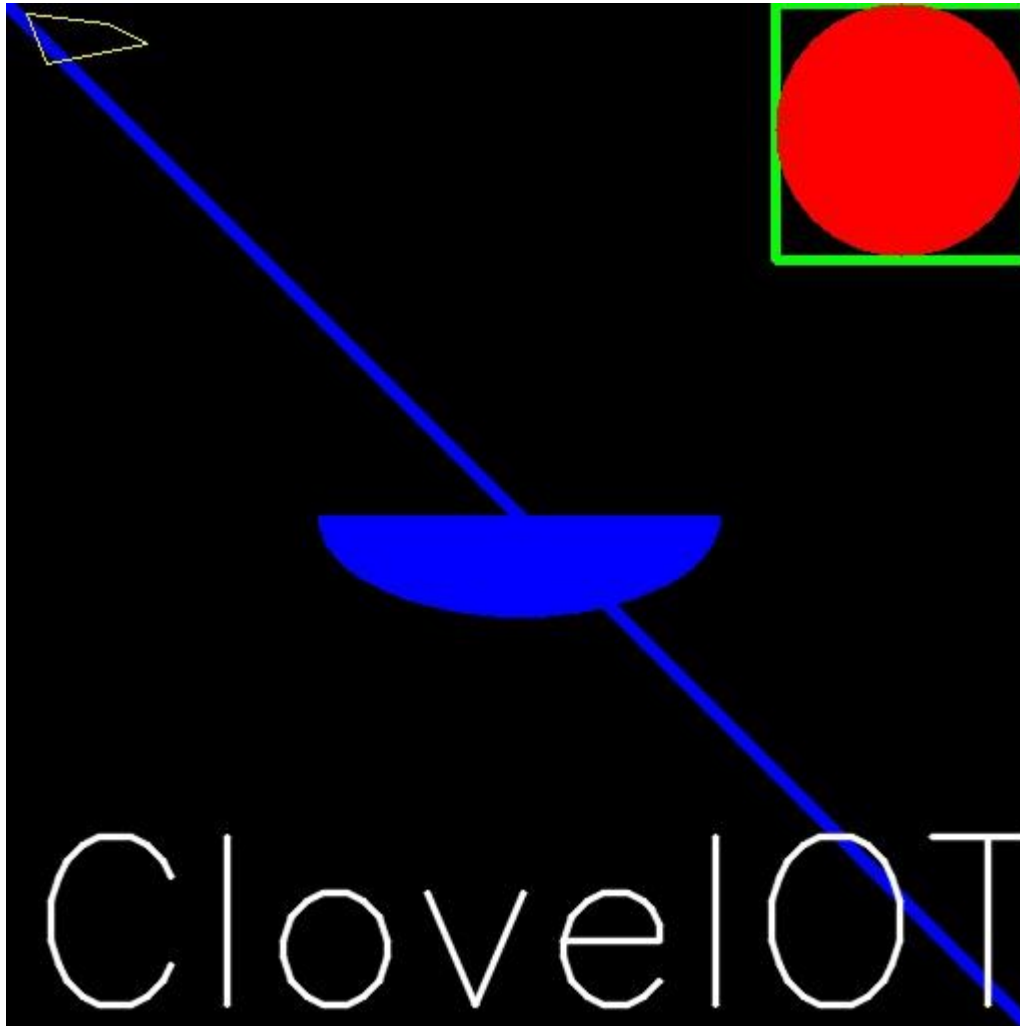
- Text data that you want to write
- Position coordinates of where you want put it (i.e. bottom-left corner where data starts).
- Font type (Check **cv2.putText()** docs for supported fonts)
- Font Scale (specifies the size of font)
- regular things like color, thickness, lineType etc. For better look

lineType=cv2.LINE is recommended.

```

import numpy as np
img = np.zeros((512,512,3), np.uint8)
img = cv2.line(img, (0,0), (511,511), (255,0,0), 5)
img = cv2.rectangle(img, (384,0), (510,128), (0,255,0), 3)
img = cv2.circle(img, (447,63), 63, (0,0,255), -1)
img = cv2.ellipse(img, (256,256), (100,50), 0,0,180,255,-1)
pts = np.array([[10,5],[20,30],[70,20],[50,10]], np.int32)
pts = pts.reshape((-1,1,2))
img = cv2.polylines(img, [pts], True, (0,255,255))
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img, 'CloveIoT', (10,500), font, 4, (255,255,255), 2, cv2.LINE_AA)

```



Mouse as a Paint-Brush:

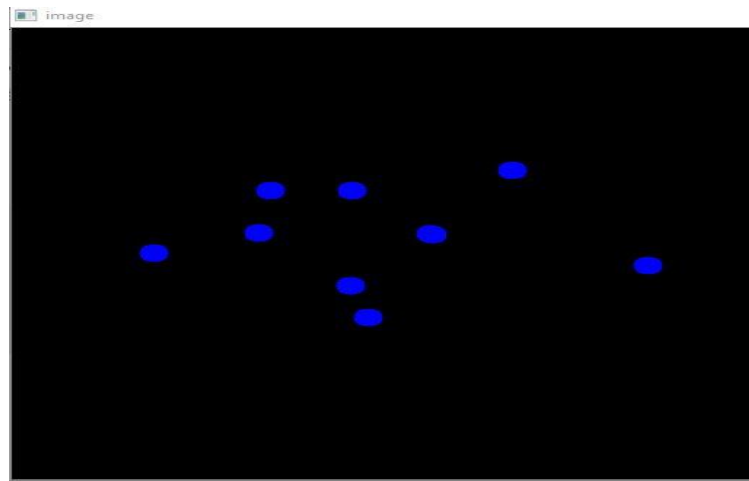
We create a simple application which draws a circle on an image wherever we double-click on it.

We create a mouse callback function which is executed when a mouse event take place. Mouse event can be anything related to mouse like left-button down, left-button up, left-button double-click etc. It gives us the coordinates (x,y) for every mouse event. With this event and location, we can do whatever we like.

Creating mouse callback function has a specific format which is same everywhere. It differs only in what the function does. So our mouse callback function does one thing, it draws a circle where we double-click.

```
>>> import cv2
>>> import numpy as np
>>> def draw_circle(event,x,y,flags,param):
>>>     if event==cv2.EVENT_LBUTTONDBLCLK:
>>>         cv2.circle(img,(x,y),10,(255,0,0),-1)

>>> img=np.zeros((512,512,3),np.uint8)
>>> cv2.namedWindow('image')
>>> cv2.setMouseCallback('image',draw_circle)
>>> while(1):
>>>     cv2.imshow('image',img)
>>>     if cv2.waitKey(20) & 0xFF == 27:
>>>         break
```



ACCESSING PIXEL VALUES:

We can access a pixel value by its row and column coordinates. For BGR image, it returns an array of Blue, Green, Red values.

```
===== RESTART: Shell =
>>> import cv2
>>> import numpy as np
>>> img=cv2.imread('untitled.png')

>>> px=img[10,10]
>>> print px
[14  0 82]
>>> |
```

ACCESSING USING ITEM FUNCTION:

```
>>> img.item(30,40,1)
0
```

MODIFYING USING ITEMSET FUNCTION:

```
>>> img.itemset((50,50,2),250)
>>> img.item(50,50,2)
250
|
```

SHAPE OF AN IMAGE:

It returns a tuple of number of rows, columns and channels (if image is color):

```
>>> img.shape
(225, 225, 3)
... |
```

IMAGE SIZE:

It gives total no of pixels

```
>>> img.size
151875
... |
```

IMAGE ROI:

ROI is obtained using Numpy indexing

```
=====
>>> import numpy as np
>>> import cv2
>>> img=cv2.imread('r.png')

>>> img.shape
(194, 259, 3)
>>> ball = img[150:170, 130:140]
>>> img[80:100, 110:120] = ball
>>> cv2.imwrite('roil.png',img)
True
>>> |
```

r.png

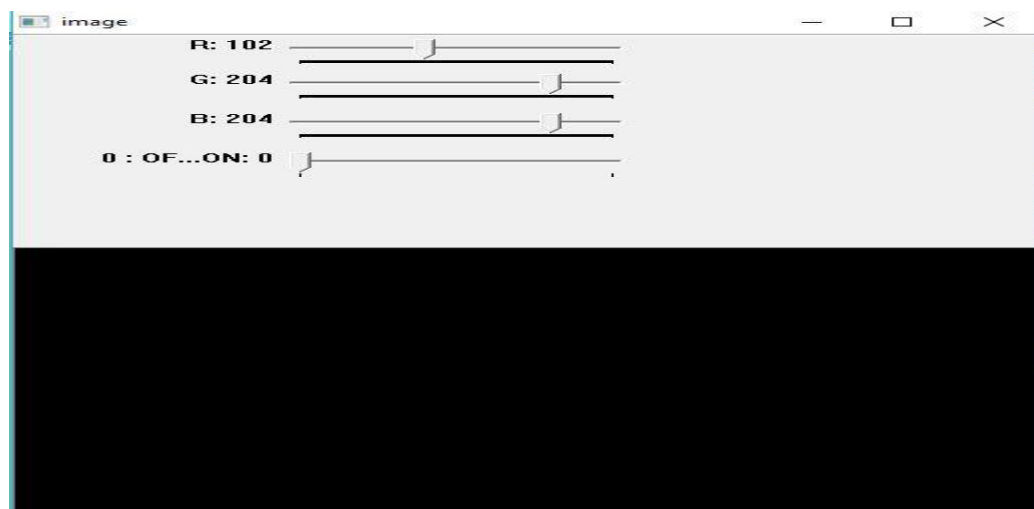


roil.png



TRACKBAR AS THE COLOR PALETTE:

In `cv2.getTrackbarPos()` function, first argument is the trackbar name, second one is the window name to which it is attached, third argument is the default value, fourth one is the maximum value and fifth one is the callback function which is executed everytime trackbar value changes. The callback function always has a default argument which is the trackbar position.



Splitting and merging Image channels:

The B,G,R channels of an image can be split into their individual planes when needed. Then, the individual channels can be merged back together to form a BGR image again.

```
>>> import numpy
>>> import cv2
>>> img=cv2.imread('gitam.png')
>>> px=img[100,100]
>>> print px
[ 44  32 153]
>>> b,g,r=cv2.split(img)
>>> img=cv2.merge((b,g,r))
>>> cv2.imwrite('merge.png',img)
True
>>> img[:, :, 2]=0
>>> cv2.imwrite('onlyred.png',img)
True
```

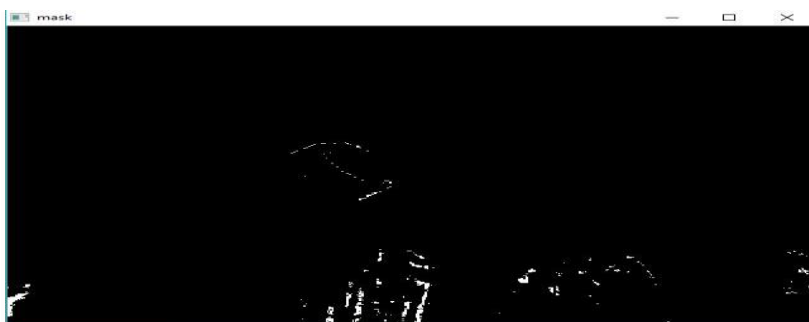


1.Changing Colorspaces:

Learn to change images between different color spaces. Also how to track a Colored Object

we can use this to extract a colored object. In HSV, it is more easier to represent a color than RGB color-space. In our application, we will try to extract a blue colored object. So here is the method:

- Take each frame of the video
- Convert from BGR to HSV color-space
- We threshold the HSV image for a range of blue color
- Extract the blue object alone, we can do whatever on that image we want.



2. Geometric Transformation:

In this concept we learned about how to do Scaling, Translation,

Affine Transformation, Rotation of images

Scaling uses re-sizing of image with Interpolation Methods like **cv2.INTER_AREA** for shrinking and **cv2.INTER_CUBIC** (slow) & **cv2.INTER_LINEAR** for zooming.

In Affine Transformation, all parallel lines in the original image will still be parallel in the output image. To find the transformation matrix, we need three points from input image and their corresponding locations in output image. **cv2.getAffineTransform** will create a 2x3 matrix which is to be passed to **cv2.warpAffine**.

Translation is moving Object Location while Rotation is flipping the image using degree

3. Smoothing Images:

Blur images with various low pass filters. Apply custom-made filters to images.

We have various Methods like Averaging, Gaussian Filtering, and Median Filtering of Images

Image Blurring is used to to remove noise from the images which reduce the quality of it. achieved by convolving the image with a low-pass filter kernel. It actually removes high frequency content (e.g.: noise, edges) from the image resulting in edges being blurred when this is filter is applied.

4. Morphological Transformations:

We will learn different morphological operations like Erosion, Dilation, Opening, Closing using different functions like:

cv2.erode (), **cv2.dilate ()**, **cv2.morphologyEx ()**

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, and second one is called **structuring element** or **kernel** which decides the nature of

Operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play.

In Erosion the process is that it removes all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases

or simply white region decreases in the image. It is useful for removing small white noises , detach two connected objects.

Dilation is exactly opposite of Erosion. It increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.

Opening is just another name of **erosion followed by dilation**.

Closing is just another name of **Dilation followed by erosion**.

❖ **Face Detection:**

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene.

Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

This technology has been available for some years now and is being used all over the place.

From cameras that make sure faces are focused before you take a picture, to Facebook when it tags people automatically once you upload a picture.

❖ **FACE DETECTION CLASSIFIERS:**

A computer program that decides whether an image is a positive image (face image) or negative image (non-face image) is called a **classifier**. A classifier is trained on hundreds of thousands of face and non-face images to learn how to classify a new image correctly. OpenCV provides us with two pre-trained and ready to be used for face detection classifiers:

A).Haar Classifier

B).LBP Classifier

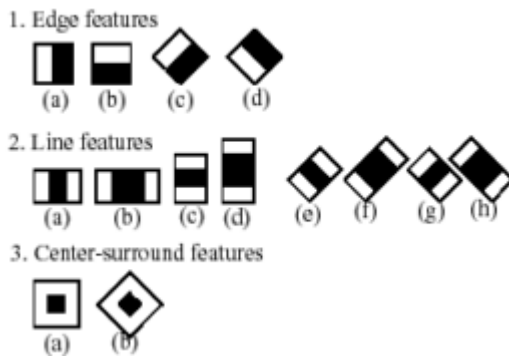
Both of these classifiers process images in gray scales, basically because we don't need color information to decide if a picture has a face or not (we'll talk more about this later on). As these are pre-trained in OpenCV, their learned knowledge files also come bundled with OpenCV opencv/data/.

To run a classifier, we need to load the knowledge files first, as if it had no knowledge, just like a newly born baby (stupid babies).

Each file starts with the name of the classifier it belongs to. For example, a **Haar cascade classifier** starts off as **haarcascade_frontalface_alt.xml**.

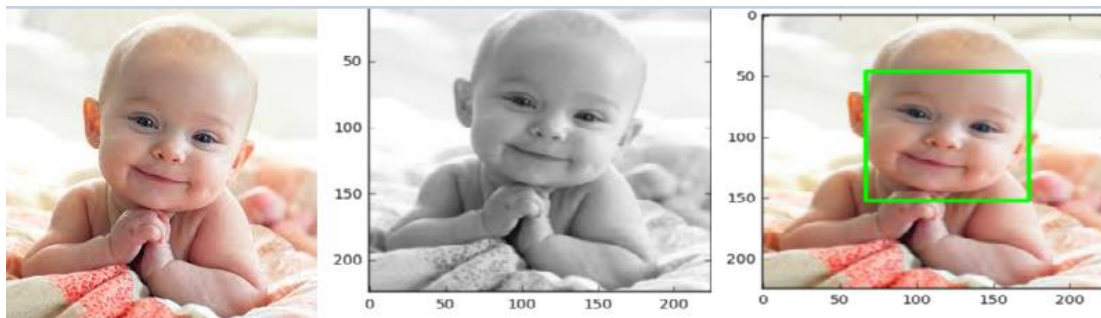
❖ HAAR CLASSIFIER:

The **Haar Classifier** is a machine learning based approach, an algorithm which (as mentioned before) are trained from many many positive images (with faces) and negatives images (without faces). It starts by extracting Haar features from each image by the windows.



OpenCV provides us with a class **cv2.CascadeClassifier** which takes as input the training file of the classifier we want to load and loads it for us.

Since we want to load **the Haar classifier**, its XML training files are stored in the `opencv/data/haarcascades/` folder.

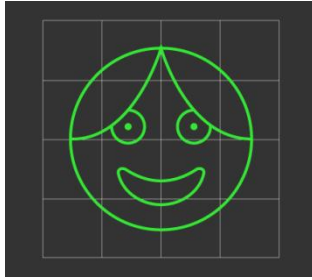


❖ LBP CASCADE CLASSIFIER:

As any other classifier, the **Local Binary Patterns**, or LBP in short, also needs to be trained on hundreds of images. LBP is a visual/texture descriptor, and thankfully, our faces are also composed of micro visual patterns.

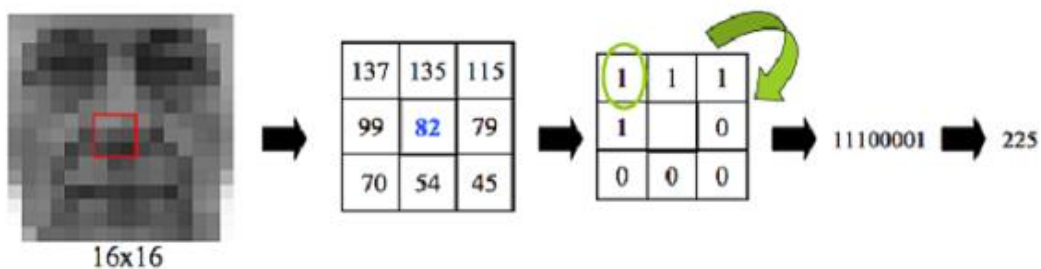
So, LBP features are extracted to form a feature vector that classifies a face from a non-face.

Each training image is divided into some blocks as shown in the picture below.



LBP Windows (disregard the first grader drawing)

For each block, LBP looks at 9 pixels (3×3 window) at a time, and with a particular interest in the pixel located in the center of the window.

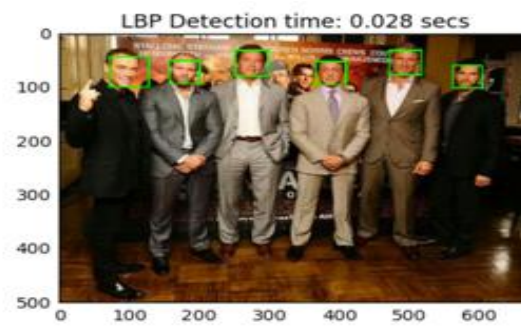


XML training files for LBP cascade are stored in the [opencv/data/lbpcascades/](#) folder, and their names start off with, as you might have guessed, **lbpcascade.**

From a coding perspective, you don't have to change anything in our face detection code except, instead of loading the Haar classifier training file you have to load the **LBP training file**, and rest of the system stays the same.

Differences Between Hass And LBP:

Algorithm	Advantages	Disadvantages
Haar	<ol style="list-style-type: none"> 1. High detection accuracy 2. Low false positive rate 	<ol style="list-style-type: none"> 1. Computationally complex and slow 2. Longer training time 3. Less accurate on black faces 4. Limitations in difficult lightening conditions 5. Less robust to occlusion
LBP	<ol style="list-style-type: none"> 1. Computationally simple and fast 2. Shorter training time 3. Robust to local illumination changes 4. Robust to occlusion 	<ol style="list-style-type: none"> 1. Less accurate 2. High false positive rate



❖ **Face Recognition:**

Face recognition consists of 4 important steps to be followed

Face Detection: Look at the picture and find a face in it.

Data Gathering: Extract unique characteristics of face that it can use to differentiate from another person, like eyes, mouth, nose, etc.

Data Comparison: Despite variations in light or expression, it will compare those unique features to all the features of all the people you know.

Face Recognition: It will determine exactly who you are.

Face Recognition through data set:

Here are three easy steps to computer coding facial recognition, which are similar to the steps that our brains use for recognizing faces. These steps are:

Data Gathering: Gather face data (face images in this case) of the persons you want to identify.

Train the Recognizer: Feed that face data and respective names of each face to the recognizer so that it can learn

Recognition: Feed new faces of that people and see if the face recognizer you just trained recognizes them.

OpenCV has three built-in face recognizers and thanks to its clean coding, you can use any of them just by changing a single line of code. Here are the names of those face recognizers and their OpenCV calls:

EigenFaces – `cv2.face.createEigenFaceRecognizer()`

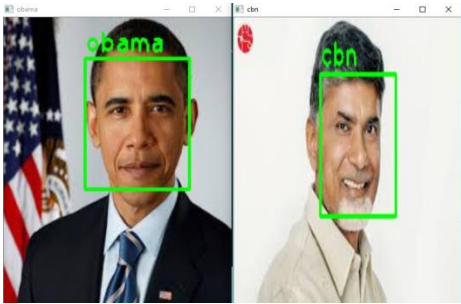
FisherFaces – `cv2.face.createFisherFaceRecognizer()`

Local Binary Patterns Histograms (LBPH) – `cv2.face.createLBPHFaceRecognizer()`

OpenCV provides us with two pre-trained and ready to be used for face detection classifiers:

Haar Classifier-**`haarcascade_frontalface_alt.xml`**.

LBP Classifier - **`lbpcascade_frontalface.xml`**.



Automatic face recognition is all about extracting those meaningful features from an image, putting them into a useful representation and performing some kind of classification on them.

Face recognition based on the geometric features of a face is probably the most intuitive approach to face recognition.

Face recognition from camera after creating database:

After reading the data from camera it creates the negative images by learning the features and then return how many faces it learnt and when it opens the camera again after execution it predicts and recognizes the facial expressions and detect the person.

Data set after training:



Execution:



❖ **Object tracking:**

Locating an object in successive frames of a video is called **tracking**.

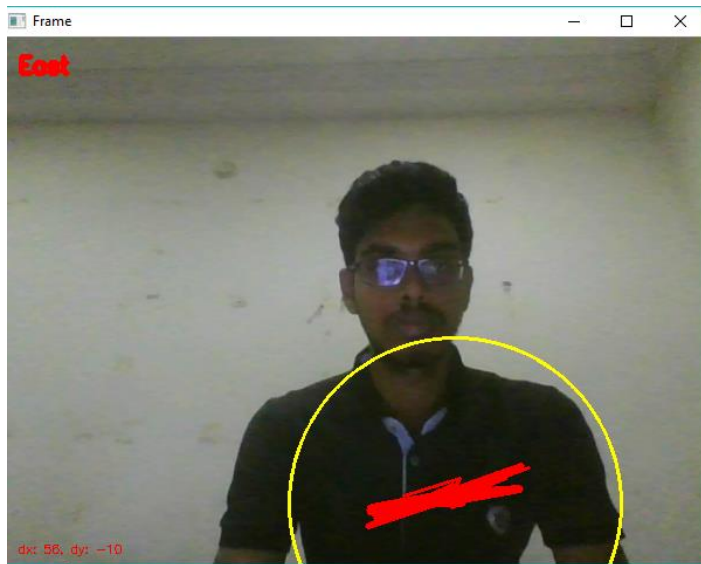
The definition sounds straight forward but in computer vision and machine learning, tracking is a very broad term that encompasses conceptually similar but technically different ideas.

1. **Dense Optical flow:** These algorithms help estimate the motion vector of every pixel in a video frame.
2. **Sparse optical flow:** These algorithms, like the Kanade-Lucas-Tomashi (KLT) feature tracker, track the location of a few feature points in an image.
3. **Kalman Filtering:** A very popular signal processing algorithm used to predict the location of a moving object based on prior motion information.
4. **Meanshift and Camshift:** These are algorithms for locating the maxima of a density function. They are also used for tracking.
5. **Single object trackers:** In this class of trackers, the first frame is marked using a rectangle to indicate the location of the object we want to track. The object is then tracked in subsequent frames using the tracking algorithm. In most real life applications, these trackers are used in conjunction with an object detector.
6. **Multiple object track finding algorithms:** In cases when we have a fast object detector, it makes sense to detect multiple objects in each frame and then run a track finding algorithm that identifies which rectangle in one frame corresponds to a rectangle in the next frame.

Tracking vs. Detection

1. **Tracking is faster than Detection:** Usually tracking algorithms are faster than detection algorithms. The reason is simple. When you are tracking an object that was detected in the previous frame, you know a lot about the appearance of the object. You also know the location in the previous frame and the direction and speed of its motion. So in the next frame, you can use all this information to predict the location of the object in the next frame and do a small search around the expected location of the object to accurately locate the object. A good tracking algorithm will use all information it has about the object up to that point while a detection algorithm always starts from scratch.

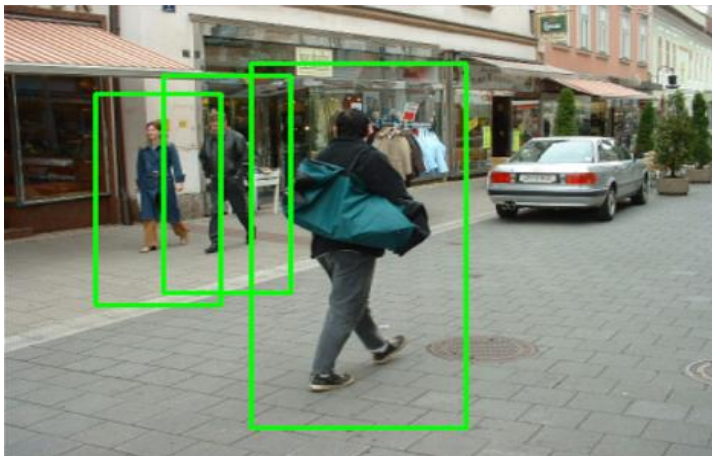
2. **Tracking can help when detection fails:** If you are running a face detector on a video and the person's face gets occluded by an object, the face detector will most likely fail. A good tracking algorithm, on the other hand, will handle some level of occlusion.



❖ Pedestrian Tracking:

We can use computer vision to exploit this semi-rigid structure and extract features to quantify the human body. These features can be passed on to machine learning models that when trained can be used to **detect** and **track** humans in images and video streams. This is especially useful for the task of **pedestrian detection**

OpenCV ships with a pre-trained HOG + Linear SVM model that can be used to perform pedestrian detection in both images and video streams. If you're not familiar with the Histogram of Oriented Gradients and Linear SVM method



❖ VIRTUAL FENCING:-

A virtual fence can be defined as a structure serving as an enclosure, a barrier, or a boundary without a physical barrier. The concept of virtual fencing occurs increasingly in discussions wherever free-ranging livestock is managed. It is especially interesting because of its potential to initially enhance ecological management, improve management by turning manual labor into cognitive labor as well as improve the life-style of livestock managers. All of these have the

Potential to reduce costs. Moreover, it opens up the possibility of managing areas that are not manageable at the moment. Virtual fencing systems offer potential benefits in animal management, including time and labor savings through the maintenance of standard and electric fences. There are also potential environmental benefits through the protection of sensitive areas, wildlife conservation, and reduction in overgrazing.

But now, we are creating a line as a fence so that if a person passes, the line changes its color from green to red.

The steps required to do the process are:-

- 1.) First we should create a line which acts a fence byusing cv2.line function
- 2.) Next, we should detect the whole pedestrian
- 3.)After detecting the whole pedestrian, we should change the color of the line if the pedestrian passes by the line(fence).

STEP-1:- Drawing a line

This is done by using the statement:-

```
cv2.line(img=img, pt1=(400, 500), pt2=(400,100),
color=(0, 255, 0), thickness=2, lineType=8, shift=0)
```

STEP 2:- Detecting the pedestrian

This is done by using hog descriptor:-

```
hog = cv2.HOGDescriptor()
```

```
hog.setSVMDetector (cv2.HOGDescriptor.getDefaultPeopleDetector ())
```

And we can find the coordinates of the detector by using

```
rects = hog.detectMultiScale(img, winStride=(4, 4),
```

```
Padding=(8, 8), scale=1.05)
```

STEP 3:-Changing the color of the line

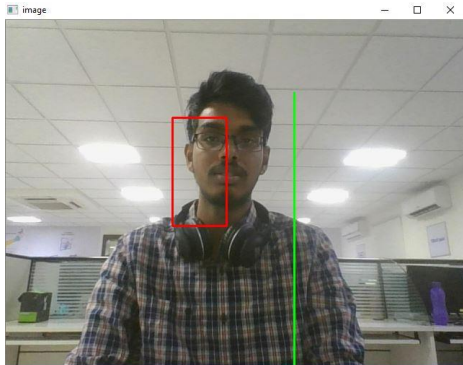
Here if the edges of the rectangle (which detects the Pedestrian) passes the coordinates of the line, then we can change the color of the line.

We can give the condition in the “if” part

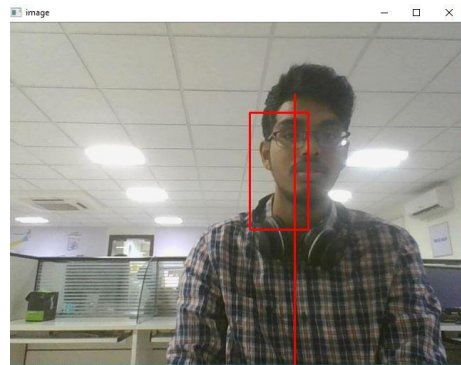
`if(x>400 or x+w>400):`

`cv2.line(img=img, pt1=(400, 500),
pt2=(400,100), color=(0, 0, 255),thickness=2,lineType=8, shift=0)`

Rectifying before entering:



After detecting when passes the line:



CREATING A 3D MESH:-

Here we draw a parallelogram (3D). Where when body gets detected in parallelogram red line appears in the parallelogram. So first we need to detect the body by using hog files. So that when face enters the parallelogram red lines appear in the middle.

Here lines are drawn using the function:-

`cv2.polylines (img,[pts],True,(0,255,0))`

We can detect whether the face enters the region by

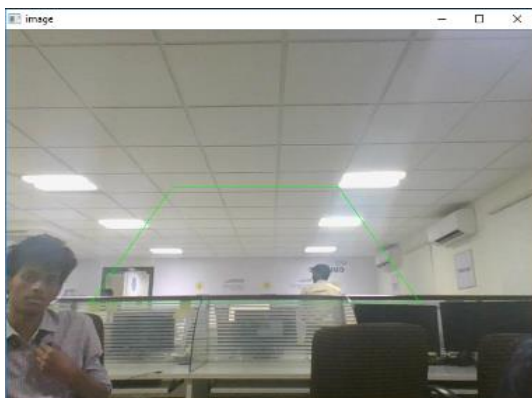
Using this if condition:-

`if(x>300 or x+w>300)`

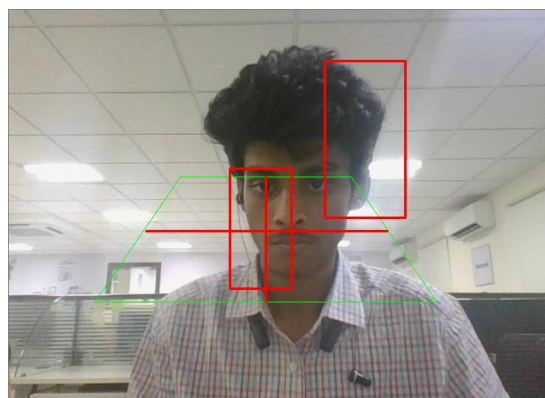
We can detect the body by using the hog descriptor:-

`hog.detectMultiScale (img, winStride=(4, 4), padding=(8, 8), scale=1.05)`

BEFORE DETECTION: -



AFTER DETECTION:-



❖ MQTT:

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with health-care providers, and in a range of home automation and small device scenarios.

Installing mqtt:

Pip install paho -mqtt

This document describes the source code for the Eclipse **Paho MQTT Python client** library, which implements versions of the **MQTT** protocol. This code provides a **client** class which enable applications to connect to an **MQTT** broker to publish messages, and to subscribe to topics and receive published messages.

Client

You can use the client class as an instance, within a class or by sub classing. The general usage flow is as follows:

- Create a client instance
- Connect to a broker using one of the connect*() functions
- Call one of the loop*() functions to maintain network traffic flow with the broker
- Use subscribe() to subscribe to a topic and receive messages
- Use publish() to publish messages to the broker
- Use disconnect() to disconnect from the broker

Publishing: Send a message from the client to the broker.

PUBLISH ()

Publish (topic, payload=None, retain=false,qos=0)

This causes a message to be sent to the broker and subsequently from the broker to any clients subscribing to matching topics. It takes the following arguments:

Topic:-the topic that the message should be published on

Payload:-

The actual message to send. If not given, or set to `none` a zero length message will be used. Passing an int or float will result in the payload being converted to a string representing that number. If you wish to send a true int/float, use `struct.pack()` to create the payload you require

Qos: -the quality of service level to use

Retain:-if set to True, the message will be set as the “last known good”/retained message for the topic.

SUBSCRIBE ():

Subscribe (topic,qos=0)

Subscribe the client to one or more topics.

This function may be called in three different ways:

Simple string and integer:-e.g. `subscribe ("my/topic", 2)`

Topic:-a string specifying the subscription topic to subscribe to.

Qos:-the desired quality of service level for the subscription. Defaults to 0.

List of string and integer tuples:

e.g. `subscribe([("my/topic", 0), ("another/topic", 2)])`

This allows multiple topic subscriptions in a single SUBSCRIPTION command, which is more efficient than using multiple calls to `subscribe()`.

Topic:-a list of tuple of format `(topic, qos)`. Both topic and qos must be present in all of the tuples.

Qos:-not used.

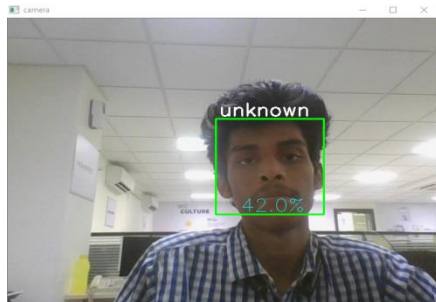
Explanation:

We made a slight change in the project where instead of hog descriptor we implemented the face Cascade detector where instead of body and objects, it identifies only the face of the person.

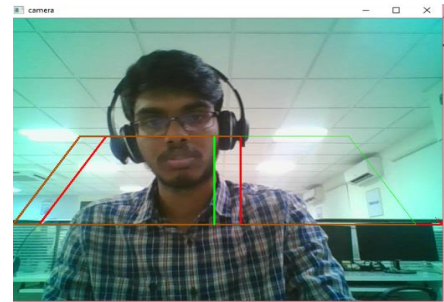
We created 2 regions in one machine of equal area and maintain the concept of facial detection and recognition on another machine and implemented the interface of MQTT with the concept of publish and subscribe.

E.g.; when the face on one machine is recognized on left region it publishes the instance to subscribed machine where the left region will be detected

Machine 1:



Machine2:



We created a parallelogram as a area and divided it into 2 parts so it could recognise that particular area only once where a unknown person is detected.

❖ BRESENHAM'S LINE ALGORITHM

Bresenham's line algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is commonly used to draw line primitives in a bitmap image (e.g. on a computer screen), as it uses only integer addition, subtraction and bit shifting.

```
function line(x0, y0, x1, y1)
  real deltax := x1 - x0
  real deltay := y1 - y0
  real deltaerr := abs(deltay / deltax) // Assume deltax != 0 (line is not vertical).
  // note that this division needs to be done in a way that preserves the fractional part
  real error := 0.0 // No error at start
  int y := y0
  for x from x0 to x1
    plot(x,y)
    error := error + deltaerr
    while error ≥ 0.5 then
      y := y + sign(deltay) * 1
      error := error - 1.0
```

How to fill the shape ():

cv2.fillPoly (img,[pts], color=(192,192,192,0.2), lineType=8, shift=0,)

❖ **APACHE KAAFKA:**

Apache Kafka is a distributed publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables you to pass messages from one end-point to another. Kafka is suitable for both offline and online message consumption. Kafka messages are persisted on the disk and replicated within the cluster to prevent data loss. Kafka is built on top of the Zoo Keeper synchronization service. It integrates very well with Apache Storm and Spark for real-time streaming data analysis.

Examples

Kafka can be used in many Use Cases.

METRICS, LOG AGGREGATION SOLUTION, STREAM PROCESSING

Kafka is a unified platform for handling all the real-time data feeds. Kafka supports low latency message delivery and gives guarantee for fault tolerance in the presence of machine failures. It has the ability to handle a large number of diverse consumers. Kafka is very fast, performs 2 million writes/sec. Kafka persists all data to the disk, which essentially means that all the writes go to the page cache of the OS (RAM). This makes it very efficient to transfer data from page cache to a network socket.

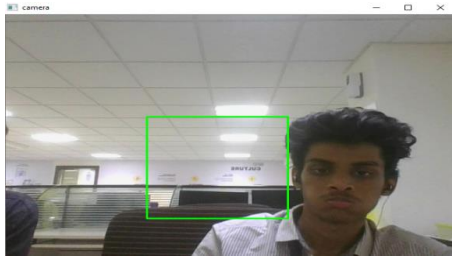
❖ **ZOOKEEPER:**

Zookeeper is a centralized open-source server for maintaining and managing configuration information, naming conventions and synchronization for distributed cluster environment. **Zookeeper** helps the distributed systems to reduce their management complexity by providing low latency and high availability. Zookeeper is a service for sure - to which clients can connect to. It provides access to clients to a tree like structure or a hierarchical name space as Zookeeper documentation says. So why we need this tree ? Of course for storing data and hence it is called a “**data tree**”. On that tree you can do the whole set of **CRUD** operations, Plus you can use **GET and SET operations** for data manipulations. It uses the standard UNIX notation for file system paths.

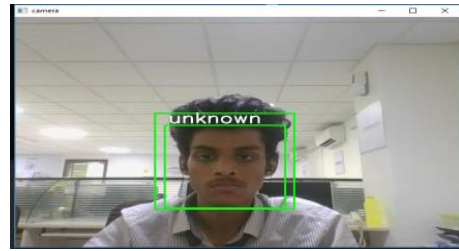
FACE RECOGNITION ON ROI (REGION OF INTEREST)

We again changed the program where we can recognize the face only inside a particular area (rectangle).

OUTSIDE THE RECTANGLE:-



INSIDE THE RECTANGLE:-



INTERN PERFORMANCE EVALUATION

Name of the Student: Pavuluri Yaswanth Chowdary

College or University Name: GITAM (Deemed to be University)

S.no.	Components To Be Evaluated	Maximum marks	Marks Obtained
1	Quality of Project	10	10
2	Attendance	5	5
3	Training	5	5
4	Communication Skills	5	5
5	Presentation	5	5
	Grand Total		30

Duration of Internship: From 30th April 2018 to 8th June 2018

Name of Evaluator: RAJNEESH RAJU. V

Designation: DIRECTOR (SYSTEMS & IOT)

REFERENCES

7.1 Opencv

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

7.2 BASIC OPENCV, NUMPY INSTALLATION (IDE USAGE AND TOOLS)

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html

7.3 BASIC UNDERSTANDING OF IMAGE PROCESSING BY VARIOUS JOURNALS AND PAPER PUBLISHERS

Burton, A.M., Bruce, V., and Hancock, P.J.B. (January 1999).
From pixels to people: A model of familiar face recognition

Burton, A.M., Bruce, V., and Johnston, R.A. (1990).
Understanding face recognition with an interactive activation model

Chellappa, R., Wilson, C.L., and Sirohey, S. (May 1995).
Human and Machine Recognition of Faces: A Survey

Chen, C.W. and Huang, C.L.
Human face recognition from a single front view
IEEE Transactions on Pattern Analysis and Machine Intelligence

<https://www.readbyqxsd.com/journal/34155>

7.4 BASIC FACE RECOGNITION CREATION CODE

https://github.com/ageitgey/face_recognition

7.5 BASIC FACE DETECTION UNDERSTANDING

<https://realpython.com/face-recognition-with-python/>

7.6 APPLICATIONS AND EXAMPLE ILLUSTRATIONS

<https://www.superdatascience.com/opencv-face-recognition/>

<http://shervinemami.info/faceRecognition.html>

<https://www.pyimagesearch.com/author/adrian/>

7.7 MQTT, BRESENHAMS ALGORITHM, KAAFKA AND ZOOKEEPER

<http://www.steves-internet-guide.com/python-mqtt-publish-subscribe/>

<https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html>

<https://data-flair.training/blogs/zookeeper-in-kafka/>

<https://kafka.apache.org/documentation/>