MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.)
A Govt. Aided UGC Autonomous & NAAC Accredited Institute, Affiliated to R.G.P.V, Bhopal

DEPARTMENT OF CSE AND IT

PROJECT REPORT

ON

FACE DETECTION SYSTEM

SUBMITTED TO:-

PROF. VIKAS SEJWAR
Dr. YOGESHWAR SINGH

SUBMITTED BY:-

1) MAITREY SHUKLA
(0901IT191033)

2) HIMANSHU PURTE
(0901IT191027)

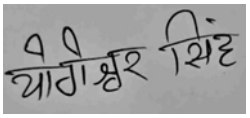
MADHAV INSTITUDE OF TECHNOLOGY AND SCIENCE GWALIOR M.P.

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that Maitrey Shukla, Himanshu Purte and working in a team have satisfactorily completed the project entitled "Face Detection System" under the guidance of Prof vikas sejwar in the partial fulfillment of the degree of Bachelor of Technology in Information Technology awarded by Madhav Institude of Technology and Science Gwalior M.p. during the academic year 2021

(Prof.Vikas Sejwar)
Mentor



(Prof. YOGESHWAR SINGH)
Mentor

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.)
A Govt. Aided UGC Autonomous & NAAC Accredited Institute, Affiliated to R.G.P.V, Bhopal

CERTIFICATE

This is to certify that the project work entitled as " ***FACE DETECTION SYSTEM***" is being Submitted by ***MAITREY SHUKLA 0901IT191033 AND HIMANSHU PURTE 0901IT191027*** in the fulfillment for the award of the Degree of Bachelor of Technology in "INFORMATION AND TECHNOLOGY" in the academic during 2019-2023 as MINOR PROJECT in 5th SEMESTER.

PROF. VIKAS SEJWAR

DR. YOGESHWAR SINGH

DEPT. OF INFORMATION AND TECHNOLOGY, MITS GWALIOR

UNDERTAKING

I declare that the work presented in this project titled " ***FACE DETECTION SYSTEM***", submitted to the DEPT OF CSE AND IT, Faculty of Madhav institute of technology and science, Gwalior, M.P for the award of the ***Bachelor of Technology*** degree in ***INFORMATION TECHNOLOGY***, is our original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

MAITREY SHUKLA 0901IT191033

HIMANSHU PURTE 0901IT191027

ACKNOWLEDGMENT

I thank the almighty for giving us the courage and perseverance in completing the Minor project. This project itself is acknowledgements for all those people who have give us their heartfelt co-operation in making this project a grand success.

I extend our sincere thanks To Director and faculty members, for providing sufficient infrastructure and good environment in the College to complete our course.

Our special thanks to all IT DEPARTMENT and peers for their valuable advises at every stage of this work

Last but not least, we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation in doing the main project.

FACE DETECTION SYSTEM

ABSTRACT

Object identification and face detection are probably the most popular applications of computer vision. This technology finds applications in various industries, such as security and social media. So we're building a face detection project through C++

Note that you should be familiar with programming in C++ and OpenCV, .We have written our code on Visual Studio code. It will ensure that you don't get confused while working on this project.

In this project, we've performed face detection by using OpenCV . We've used **Haar cascade Method**, but we can also use other systems.

EXTENSION:- THIS PROJECT CAN FUTHER BE INCREASED TO VIRTUAL PAINT ,DOCUMENT SCANNER AND LICENSE PLATE DETECTOR.

INDEX

PAGE.NO.

INTRODUCTION

Face detection can consider a substantial part of face recognition operations. According to its strength to focus computational resources on the section of an image holding a face. The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin colour, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution.

Object detection is one of the computer technologies, which connected to the image processing and computer vision and it interacts with detecting instances of an object such as human faces, building, tree, car, etc. The primary aim of face detection algorithms is to determine whether there is any face in an image or not.

In recent times, a lot of study work proposed in the field of Face Recognition and Face Detection to make it more advanced and accurate, but it makes a revolution in this field when Viola-Jones comes with its Real-Time Face Detector, which is capable of detecting the faces in real-time with high accuracy.

Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc.

Face Detection widely popular subject with a huge range of applications. Modern day Smartphone's and Laptops come with in-built face detection

software's, which can authenticate the identity of the user. There are numerous apps that can capture, detect and process a face in real time, can identify the age and the gender of the user, and also can apply some really cool filters. The list is not limited to these mobile apps, as Face Detection also has a wide range of applications in Surveillance, Security and Biometrics as well. But the origin of its Success stories dates back to ***2001***, when ***Viola and Jones*** proposed the first ever Object Detection Framework for Real Time Face Detection in Video Footage.

OpenCV:-

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing; video capture and analysis including features like face detection and object detection.

Computer Vision:-

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields −

- **Image Processing** − It focuses on image manipulation.
- **Pattern Recognition** − It explains various techniques to classify patterns.
- **Photogrammetry** − It is concerned with obtaining accurate measurements from images.

Features of OpenCV Library:-

Using OpenCV library, you can −

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc

OpenCV Library Modules

Following are the main library modules of the OpenCV library.

Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array **Mat**, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.core**.

Image Processing

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.imgproc**.

Video

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.video**.

Video I/O

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.videoio**.

calib3d

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.calib3d**.

features2d

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.features2d**.

Objdetect

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.objdetect**.

Highgui

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the features of this module is included in two different packages namely, **org.opencv.imgcodecs** and **org.opencv.videoio**.

STORING IMAGE:-

To capture an image, we use devices like cameras and scanners. These devices record numerical values of the image (Ex: pixel values). OpenCV is a library which processes the digital images, therefore we need to store these images for processing.

The **Mat** class of OpenCV library is used to store the values of an image. It represents an n-dimensional array and is used to store image data of grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms, etc.

This class comprises of two data parts: the **header** and a **pointer**

- **Header** − Contains information like size, method used for storing, and the address of the matrix (constant in size).
- **Pointer** − Stores the pixel values of the image (Keeps on varying).

READING IMAGES:-

The **Imgcodecs** class of the package **org.opencv.imgcodecs** provides methods to read and write images. Using OpenCV, you can read an image and store it in a matrix (perform transformations on the matrix if needed). Later, you can write the processed matrix to a file.

The **read()** method of the **Imgcodecs** class is used to read an image using OpenCV.

WRITING IMAGES;-

The **write()** method of the **Imgcodecs** class is used to write an image using OpenCV..

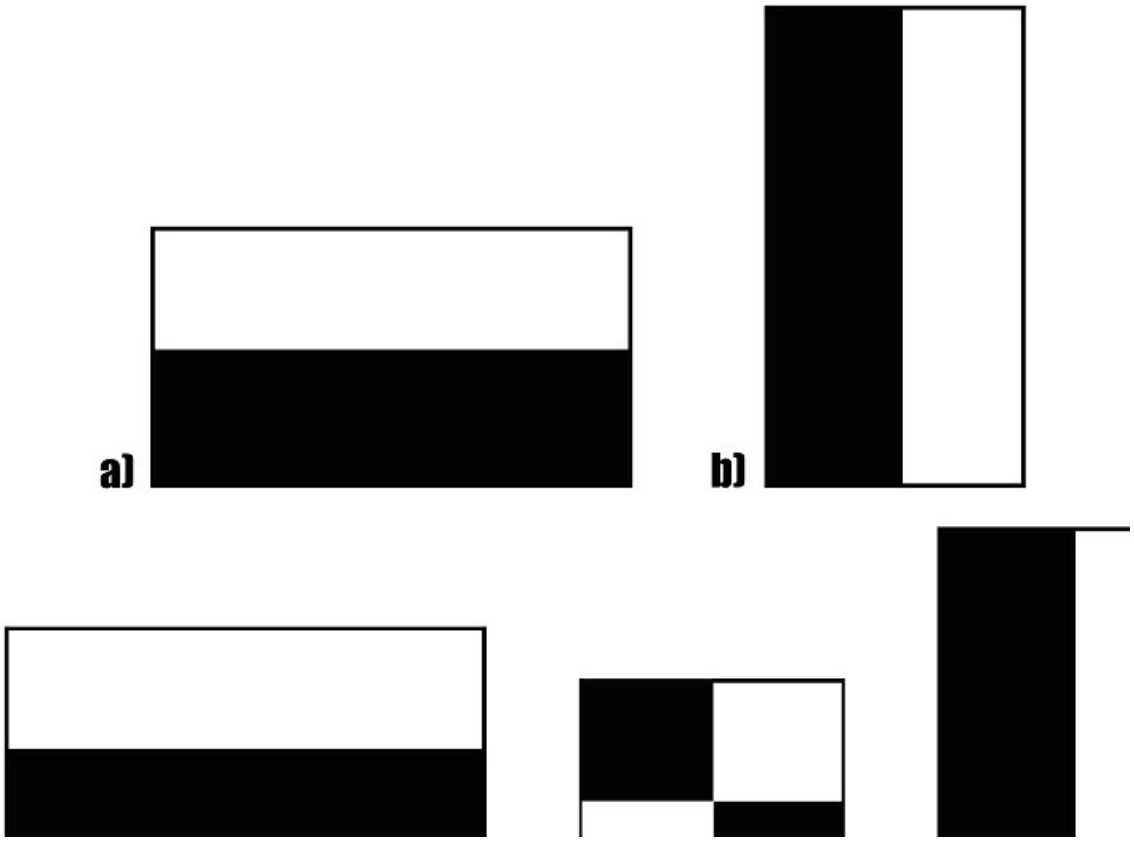
To write an image, you need to invoke the **imwrite()** method of the **Imgcodecs** class.

Haar-cascade Detection in OpenCV:-

It is an Object Detection Algorithm used to identify faces in an image or a real time video**.** The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them.

The repository has the models stored in XML files, and can be read with the OpenCV methods. These include models for face detection, eye detection, upper body and lower body detection, license plate detection etc.

FEATURES:-



The first contribution to the research was the introduction of the Haar features shown above. These features on the image make it easy to find out the edges or the lines in the image, or to pick areas where there is a sudden change in the intensities of the pixels.

The objective here is to find out *the sum of all the image pixels lying in the darker area of the Haar feature* and *the sum of all the image pixels lying in the lighter area of the Haar feature***.** And then find out their difference. Now if the image has an edge separating dark pixels on the right and light pixels on the left, then the Haar value will be closer to 1. That means, we say that there is an edge detected if the Haar value is closer to 1.

CASCADING:-

The subset of all 6000 features will again run on the training images to detect if there's a facial feature present or not. Now if we have taken a standard window size of 24x24 within which the feature detection will be running. It's again a tiresome task.

In the Viola — Jones research, they had a total of 38 stages for something around 6000 features. The number of features in the first five stages is 1, 10, 25, 25, and 50, and this increased in the subsequent stages. The initial stages with simpler and lesser number of features removed most of the windows not having any facial features, thereby reducing the false negative ratio, whereas the later stages with complex and more number of features can focus on reducing the error detection rate, hence achieving low false positive ratio.

So this is how the detection of features takes place in stages. You can notice that, when the window is at a non-face region, only the first stage with two rectangle features are running, and as they discard the window before the second stage starts. Only one window which actually contains a face, runs both the stages and detects the face.

What OpenCV provides?:-

OpenCV provides a training method (Cascade Classifier Training) or pertained models, that can be read using
the cv::CascadeClassifier::load method. The pertained models are located in the data folder in the OpenCV installation.

The following code example will use pertained Haar cascade models to detect faces and eyes in an image. First, a cv::CascadeClassifier is created and the necessary XML file is loaded using

the **cv::CascadeClassifier::load** method. Afterwards, the detection is done using the **cv::CascadeClassifier::detectMultiScale** method, which returns boundary rectangles for the detected faces or eyes.

DIFFERENT CHAPTERS INVOLVED:-

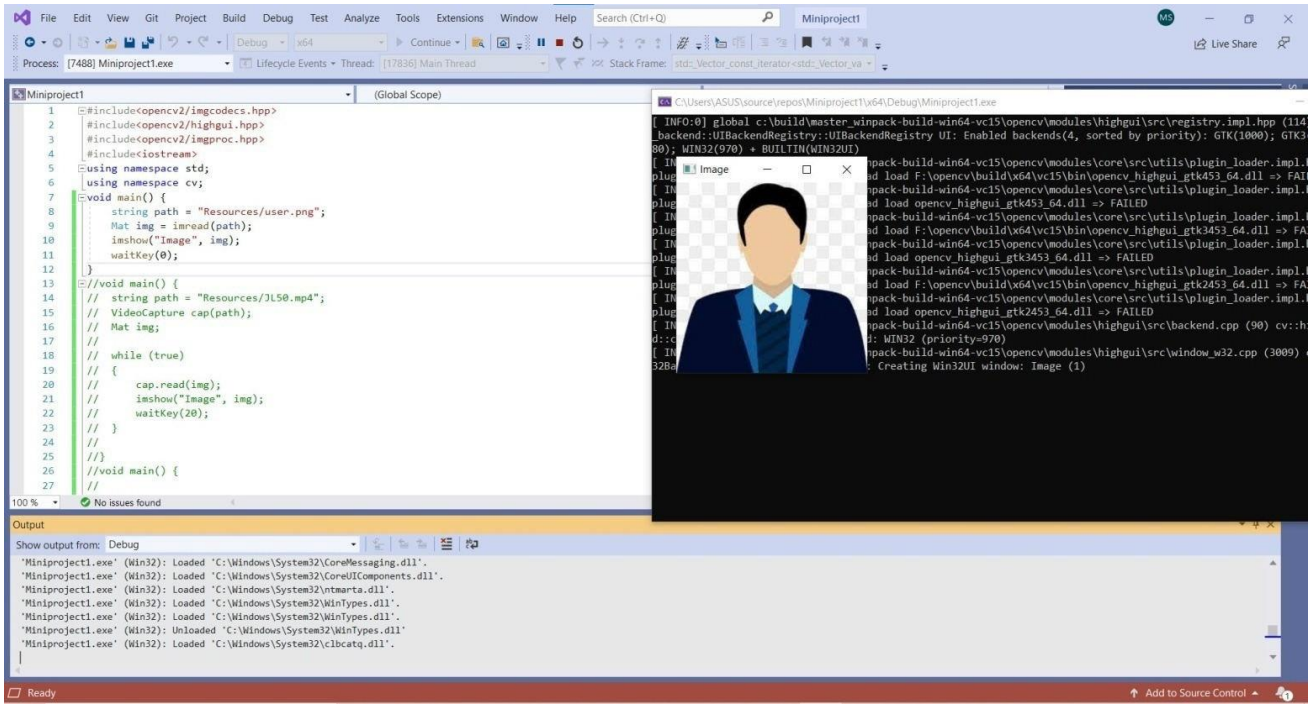
1) READ IMAGES, VIDEOS AND WEBCAME.

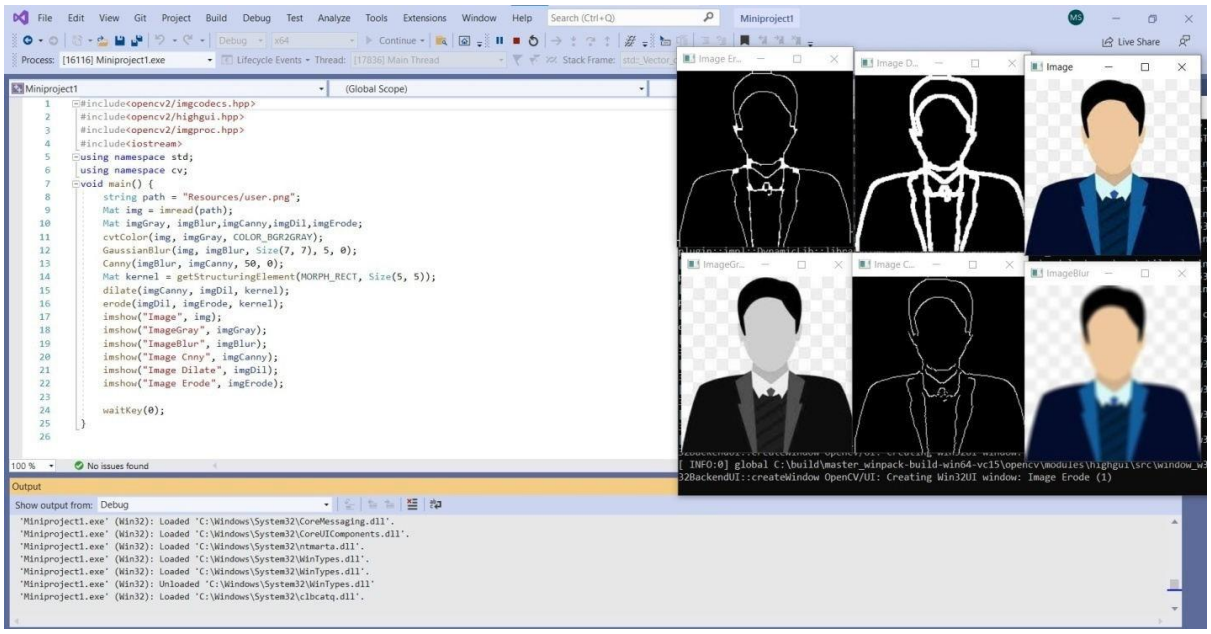
2) RESIZE AND CROP.

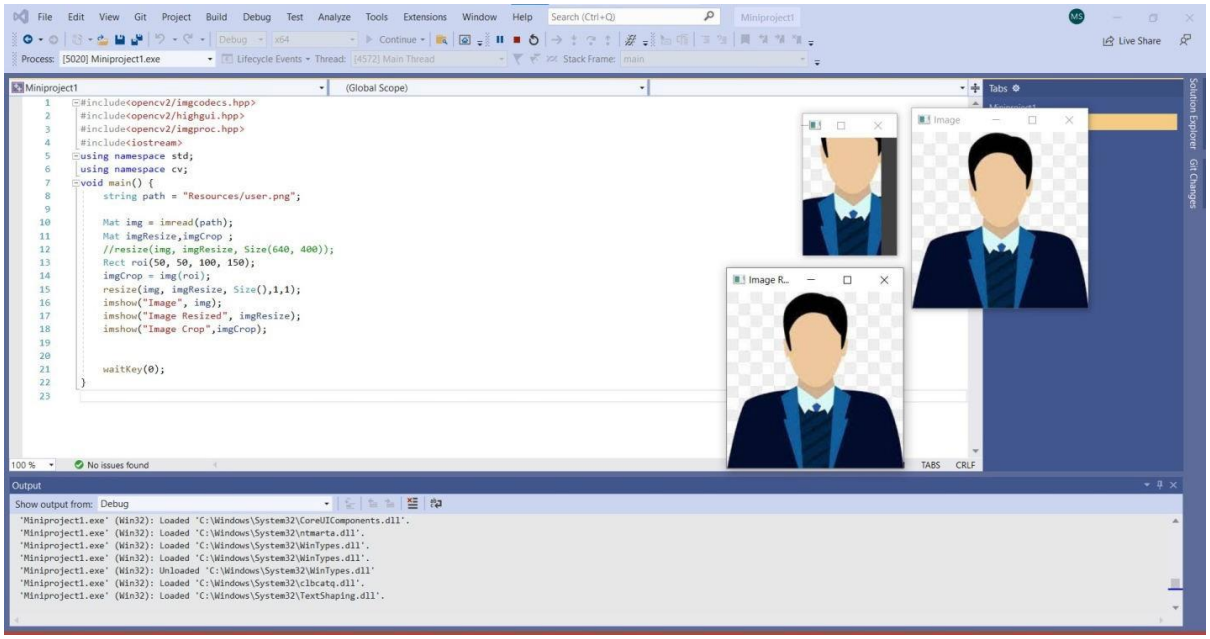
3) DRAWING SHAPES AND TEXTS.

4) COLOR DETECTION.

5) SHAPES AND CONTOUR DETECTION.


Miniproject1
(Global Scope)
#include<opencv2/imgcodecs.hpp>
#include<opencv2/highgui.hpp>
#include<opencv2/imgproc.hpp>
#include<iostream>
using namespace std;
using namespace cv;
void main() {
string path = "Resources/user.png";
Mat img = imread(path);
imshow("Image", img);
waitKey(0);
}
//void main() {
// string path = "Resources/JL50.mp4";
// VideoCapture cap(path);
// Mat img;
//
// while (true)
// {
// cap.read(img);
// imshow("Image", img);
// waitKey(20);
// }
//
//}
//void main() {
//
100 %
No issues found
Image
Output
Show output from: Debug
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\CoreMessaging.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\CoreUIComponents.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\ntmarta.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
'Miniproject1.exe' (Win32): Unloaded 'C:\Windows\System32\WinTypes.dll'
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\clbcatq.dll'.
Ready
Add to Source Control


Miniproject1
(Global Scope)
#include<opencv2/imgcodecs.hpp>
#include<opencv2/highgui.hpp>
#include<opencv2/imgproc.hpp>
#include<iostream>
using namespace std;
using namespace cv;
void main() {
string path = "Resources/user.png";
Mat img = imread(path);
Mat imgGray, imgBlur,imgCanny,imgDil,imgErode;
cvtColor(img, imgGray, COLOR_BGR2GRAY);
GaussianBlur(img, imgBlur, Size(7, 7), 5, 0);
Canny(imgBlur, imgCanny, 50, 0);
Mat kernel = getStructuringElement(MORPH_RECT, Size(5, 5));
dilate(imgCanny, imgDil, kernel);
erode(imgDil, imgErode, kernel);
imshow("Image", img);
imshow("ImageGray", imgGray);
imshow("ImageBlur", imgBlur);
imshow("Image Cnny", imgCanny);
imshow("Image Dilate", imgDil);
imshow("Image Erode", imgErode);
waitKey(0);
}
100 %
No issues found
Image
ImageBlur
Output
Show output from: Debug
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\CoreMessaging.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\CoreUIComponents.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\ntmarta.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
'Miniproject1.exe' (Win32): Unloaded 'C:\Windows\System32\WinTypes.dll'
'Miniproject1.exe' (Win32): Loaded 'C:\Windows\System32\clbcatq.dll'.



*MAIN CODE FOR FACE DETECTION:-

```
#include "opencv2/objdetect.hpp"

#include "opencv2/highgui.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/videoio.hpp"

#include <iostream>
```

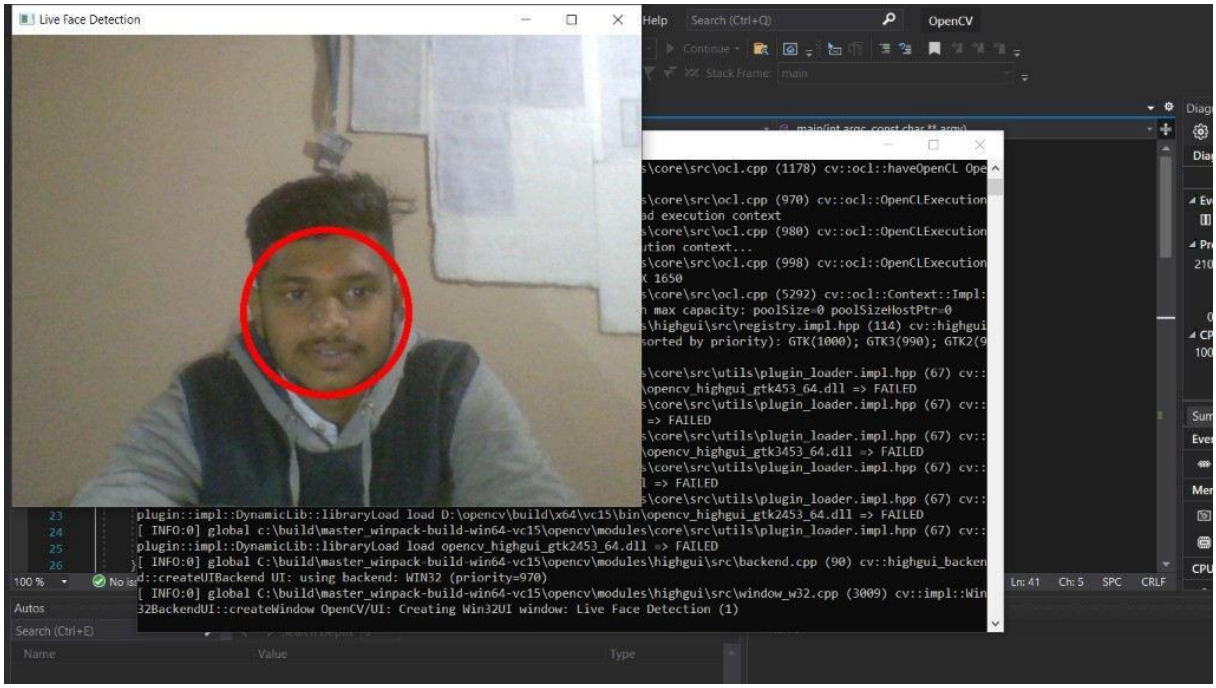
```
using namespace std;
using namespace cv;

CascadeClassifier face_cascade;

void faceDetection(Mat frame)
{
   Mat frame_gray;
   cvtColor(frame, frame_gray, COLOR_BGR2GRAY);

   // Detect faces
   std::vector<Rect> faces;
   face_cascade.detectMultiScale(frame_gray, faces);

   for (size_t i = 0; i < faces.size(); i++)
   {
      Point center(faces[i].x + faces[i].width / 2, faces[i].y + faces[i].height / 2);
```

```
        ellipse(frame, center, Size(faces[i].width / 2, faces[i].height / 2), 0,
0, 360, Scalar(0, 0, 255), 6);
        Mat faceROI = frame_gray(faces[i]);
    }

    imshow("Live Face Detection", frame);
}

int main(int argc, const char** argv)
{
    string                    faceClassifier                    =
"C:/Users/91700/source/repos/OpenCV/OpenCV/resources/haarcascad
e_frontalface_default.xml";

    if (!face_cascade.load(faceClassifier))
    {
        cout << "Could not load the classifier";
        return -1;
    };
```

```
cout << "Classifier Loaded!" << endl;

// Read the video stream from camera
VideoCapture capture(0);

if (!capture.isOpened())
{
   cout << "Could not open video capture";
   return -1;
}

Mat frame;
while (capture.read(frame))
{
   if (frame.empty())
   {
      cout << "No frame captured from camera";
      break;
   }
```

```
        // Apply the face detection with the haar cascade classifier
        faceDetection(frame);

        if (waitKey(10) == 'q')
        {
            break; // Terminate program if q pressed
        }
    }

    return 0;
}
```

OUTPUT:-


Live Face Detection
plugin::impl::DynamicLib::libraryLoad load D:\opencv\build\x64\vc15\bin\opencv_highgui_gtk2453_64.dll => FAILED
[INFO:0] global c:\build\master_winpack-build-win64-vc15\opencv\modules\core\src\utils\plugin_loader.impl.hpp (67) cv::
plugin::impl::DynamicLib::libraryLoad load opencv_highgui_gtk2453_64.dll => FAILED
[INFO:0] global C:\build\master_winpack-build-win64-vc15\opencv\modules\highgui\src\backend.cpp (90) cv::highgui_backen
d::createUIBackend UI: using backend: WIN32 (priority=970)
[INFO:0] global C:\build\master_winpack-build-win64-vc15\opencv\modules\highgui\src\window_w32.cpp (3009) cv::impl::Win
32BackendUI::createWindow OpenCV/UI: Creating Win32UI window: Live Face Detection (1)


Live Face Detection

CONCLUSION:-

Haar Cascade Detection is one of the oldest yet powerful face detection algorithms invented. It has been there since long, long before Deep Learning became famous. Haar Features were not only used to detect faces, but also for eyes, lips, license number plates etc.

Face detection technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc.

These applications usually work in controlled environments and detection algorithms can take advantage of the environmental constraints to obtain high detection accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

To achieve these goal computers must be able to reliably identify nearby people in a manner that fits naturally within the pattern of normal human interactions. They must not require special interactions and must conform to human intuitions about when recognition is likely. This implies that future smart environments should use the same modalities as humans, and have approximately the same limitations. These goals now appear in reach -- however, substantial research remains to be done in making person recognition technology work reliably, in widely varying conditions using information from single or multiple modalities.

REFERENCES:-

1) https://www.youtube.com/watch?v=2FYm3GOonhk&t=5933s
2) WIKIPEDIA.
3) https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.
4) https://docs.opencv.org/4.x/df/d65/tutorial_table_of_content_introduction.html.
5) https://www.tutorialspoint.com/opencv/opencv_blur_averaging.htm.

THANKING YOU