University of Colorado Boulder

EXERCISE 1

BY
Amreeta Sengupta

06/14/2019
Question 2,3,4,5 have been performed using Jetson Nano

QUESTION 1

- Video Analytics can broadly be defined as the technique of processing stored or previously captured sequence of images. Some major areas which video analytics encompasses are:
  - Image acquisition and encoding involves photometer technologies, digital format for arrays of pixels in frames and compression and decompression of images
  - Computer Vision which is basically emulation and replication of human vision with use of computers in an uncontrolled environment. Its primary focus is to augment human vision and it can be used in several applications including Robotics, Self-driving vehicles, Vision Prosthetics etc.
  - Machine Vision majorly focuses on automation of a process in a highly controlled environment. Its major application lies in Industrial Automation, Robotics, Sorting, Segmentation and recognition etc.
  - Image Processing involves processing and analysis of data from various detectors such as photometers and radiometers in order to study and learn the properties of the observed target.
  - Machine Learning is study of algorithms and training of data in order to automatically learn and improve the performance required for performing a specific task
  - Real-time and interactive systems are systems that are required to perform a task within a specified deadline or a system that involves interaction with the users.
  - Storage, networking, database and computing are needed for processing, storage and analysis of the data used in video analytics.
- Video Analytics has a wide array of applications which includes the following:
  - Augmented Reality views of scenes for improved understanding which for example could be used to observe an aircraft via a computer along with the estimates of the aircraft data such as attitude etc. This enables a smooth transition from frames from a digital camera to a view of the user in real time which allows a better and more visually enriching experience for the user. It can be used in Intelligent Transportation systems, Air Traffic safety etc.
  - Skeletal transformations to track the movement and estimate the intent and trajectory of an animal that might jump onto a highway which employs grey map transformation of the image and then thresholding of the image for foreground tracking and finally application of progressive thinning algorithms to derive the skeletal of the object. This is widely used in Gesture recognition systems for entertainment.

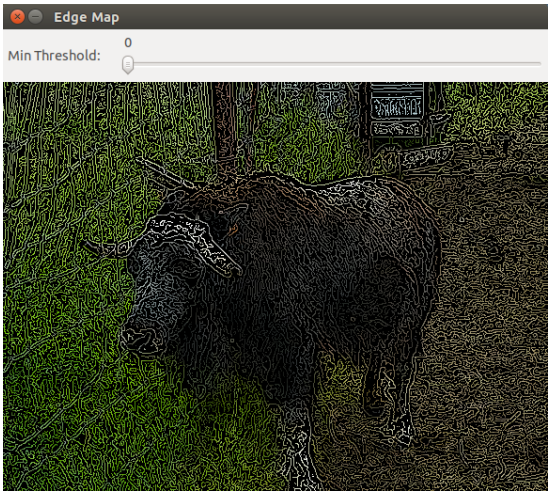
- ❖ Fully autonomous or mostly autonomous vehicles with human supervisory control only which can make the driving experience for customers much more comfortable with capabilities like cars which can parallel park themselves.
- ❖ Beyond face detection to reliable recognition and, perhaps more importantly, for expression feedback which can perhaps be used in semiautonomous vehicles to gauge the expression of the driver.
- ❖ Virtual shopping (AR to try products) which can make shopping a fun and more comfortable experience for the customers and entitle them to view themselves in that new suit.
- ❖ Signage that interacts with viewers is a way of communicating with the viewers based on the known data such as expressions, likes, dislikes etc.
- ❖ Two-way television and interactive digital cinema which will provide users with an entertainment experience that can be influenced and controlled by them as if they were the actors of the story.
- ❖ Interactive telemedicine which allows healthcare facilities available round the clock from medical experts throughout the world.

- Advanced video analytics has led to a revolution in today's technology allowing us to reach new heights in the field of image processing, analysis and data extraction which was not considered to be possible before. It can provide us with effective and improved solutions in various industries such as medical industry, entertainment industry, social network etc. Though it is quite a challenge, video analytics has given us the hope for emulation and replication of human vision and use of this for vision-based automation applications. This is rapidly growing technology with a promising future for a more efficient, intelligent system with improved public safety.

Overall, Video Analytics is a step towards an advanced intelligent computing system. It can be explored with the use of several easy to use tools such as FFmpeg for encoding and decoding of sequence of images or GNU Image Processing (GIMP) for still images. It allows us to perform image segmentation, processing, transformation, recognition and acquisition of useful data which can be used in real world applications. It's structure can be classified into two main segments which are embedded intelligent sensors and cloud-based processing for analysis. With the rapid progress in computing capabilities of system using the cloud and embedded sensors, the promise of inverse rendering can perhaps be realized soon. This opens a door full of opportunities for us in various sectors of the real world such as providing world-class health care facilities such as vision system prosthetics, improved entertainment for users with a personal touch and use of augmented reality for better understanding. However, this will be challenging in terms for the high-level image processing algorithms and machine learning as well as the highly intensive data centric computer architecture. But if it is achieved, then it can be beneficial in all areas of the industry for an efficient and intelligent system.

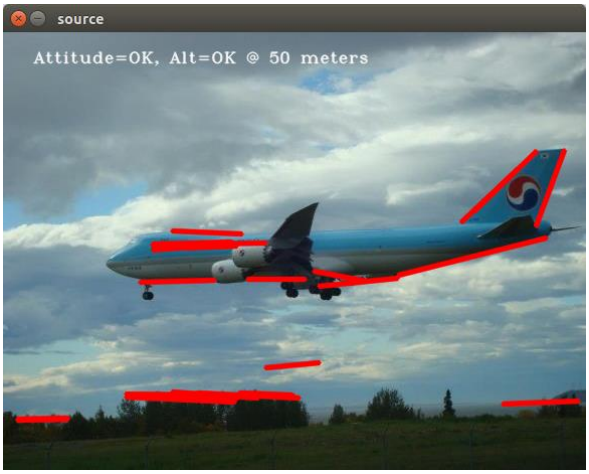
QUESTION 2

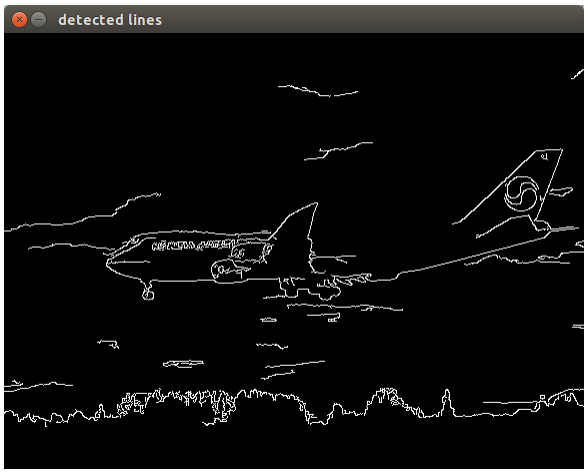
Canny Edge Detection Transform



- This code is used to apply canny transform on images which is basically an edge detection transform. createTrackbar API is used to create a trackbar to set the low threshold for the implementation of the canny algorithm.
- src is an object of Mat class which is initialized with the image frame. cvtColor API to convert the image to grey scale from RGB.
- Noise reduction is done using blur API with a filter of kernel size of 3X3. Canny API is used to return the edge detected canny mask.
- It takes lowthreshold and highthreshold as argument where highthreshold is coded as 3 times the lowthreshold value.
- If the variation in the gradient is higher than the highthreshold, then it is detected as a boundary of an edge and if it is between the lowthreshold and the high threshold, then it is detected as a continuation of an edge and if it is lower than the low threshold, then it is ignored. Scalar::all(0) is used to fill the image with zeros.
- copyTo function returns the original image masked using canny mask. imshow function is used to display the resultant image. The program will terminate when the user presses a key.

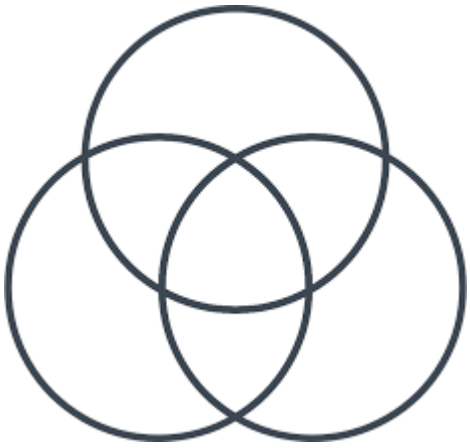
Hough-Line Tranform

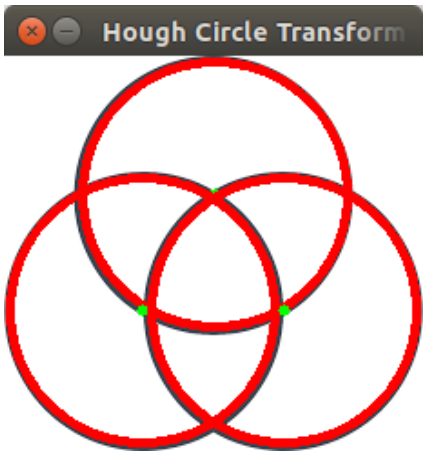




- This code is used to apply Hough-line transform on images which is basically a straight-line detection transform. src is an object of Mat class which is initialized with image frame.
- The canny transform is applied to detect the edges of this image. lines is a variable of vector type which stores the 4 intergers for the lines that are detected in the image.
- HoughLinesP function returns the endpoints of the detected lines.
- Line function is used draw the lines on the original image i.e. src. imshow is used to display the resultant image.
- The program will terminate when the user presses a key.

Hough-Circle Transform





- In the example images, there aren't any images with prominent circles and hence circles were not getting detected by the transform. Therefore, I have used an image downloaded from the net with prominent circles.
- This code is used to apply Hough-circle transform on images which is basically a circle detection transform.

- src is an object of Mat class which is initialized with image frame. cvtColor API to convert the image to grey scale from RGB.
- Noise reduction is done using blur API with a filter of kernel size of 2X2. circles is a variable of vector type which stores the 3 floats for the circles that are detected in the image.
- HoughCircles function returns the center points and radius of the detected circles.
- Circle function is used draw the detected circle on the original image i.e. src. imshow is used to display the resultant image.
- The program will terminate when the user presses a key.

Sobel description

```
amreeta@amreeta-desktop:~/Downloads/src/capture-transformer$ ./sobel ../../examp
le-images/Trees.jpg
Gtk-Message: 17:25:10.699: Failed to load module "canberra-gtk-module"
amreeta@amreeta-desktop:~/Downloads/src/capture-transformer$ ./sobel ../../examp
le-images/Trees.jpg
```

Sobel transform is a transform that is primarily used for emphasizing the edges of an image in both vertical and horizontal directions. At each pixel of the image, the gradient of the image intensity is determined which displays the even or sudden changes that occur at each pixel of the image which helps in establishing whether the pixel of interest represents an edge or not.

The figure on the left side shows the original image while the figure on the right shows us the image on application of Sobel Transform which enhances the edges. Sobel Transform calculates the gradient by basically placing the kernel matrix over each pixel of the source image.

For vertical derivative

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A$$ where A is the source image and $*$ represents the convolution operation

For horizontal derivative

$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A$ where A is the source image and $*$ represents the convolution operation

Here, while G_x increases in the right direction, G_y increases downwards. Thus, at each pixel of the image, approximation of the intensity gradient is calculated, and the gradient magnitude and direction is given by:

$$G = \sqrt{Gx^2 + Gy^2} \quad \text{and} \quad \theta = \operatorname{atan}\frac{Gx}{Gy}$$

The areas in an image where the intensity gradient is high is represented by white line in the resultant image on application of Sobel Transform. This indicates the likely edges in the image. Sobel Transform can be used for Image Segmentation.

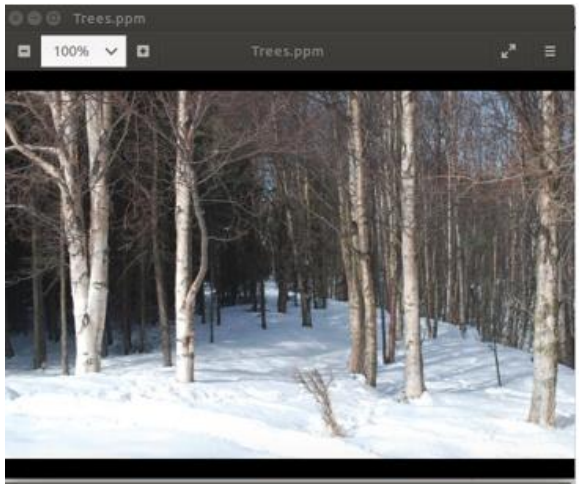
Code Explanation

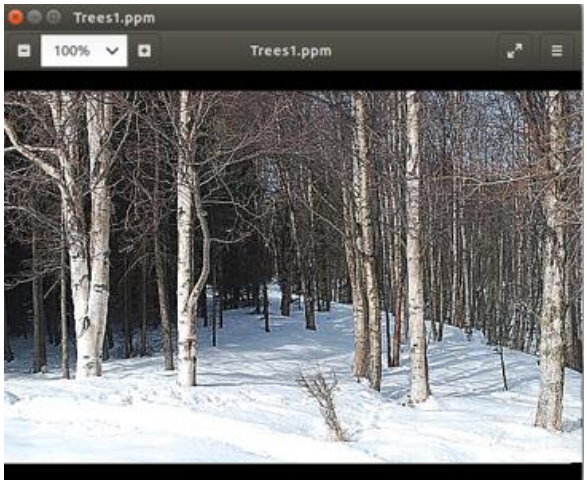
- This code reads the image as given by the user argument.
- To reduce the noise, Gaussian Blur Filter with a kernel size of 3 is used.
- The image is then converted from RGB to gray scale using cvtColor function.
- Sobel function is used to calculate the first order derivative of in x and y direction.
- In order to avoid overflow, the depth of the image we use is 16-bit unsigned and we use convertScaleAbs to convert it back to 8-bit.
- The addWeighted function is used to calculate the approximate total gradient.
- The final image is displayed using imshow() function.

QUESTION 3

Sharpen

```
amreeta@amreeta-desktop:~/Downloads/sharpen-psf$ make sharpen
gcc -O3   -c sharpen.c
gcc  -O3   -o sharpen sharpen.o -lpthread
amreeta@amreeta-desktop:~/Downloads/sharpen-psf$ ./sharpen Trees.ppm Trees1.ppm
amreeta@amreeta-desktop:~/Downloads/sharpen-psf$
```

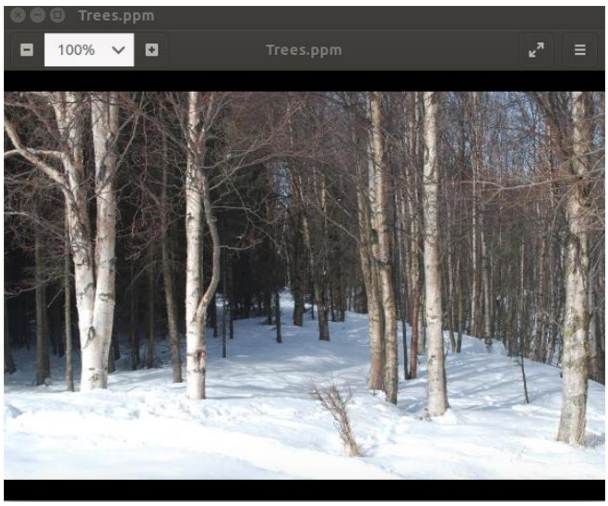


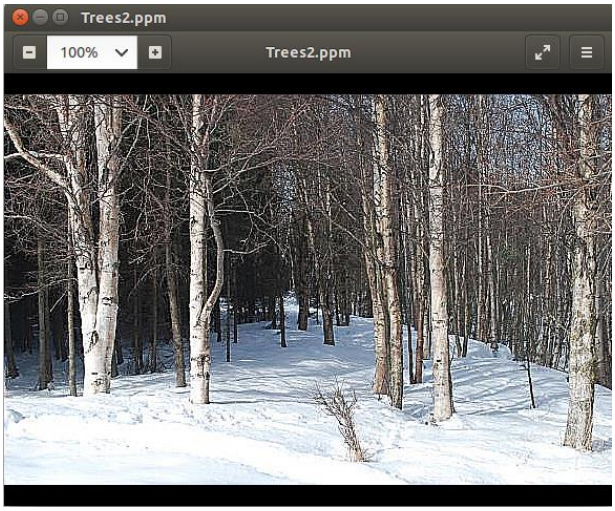


Sharpen_grid

```
amreeta@amreeta-desktop:~/Downloads/sharpen-psf$ ./sharpen_grid Trees.ppm Trees2
.ppm
  source file Trees.ppm read
frame 0 completed
frame 1 completed
frame 2 completed
frame 3 completed
frame 4 completed
frame 5 completed
frame 6 completed
frame 7 completed
frame 8 completed
frame 9 completed
frame 10 completed
frame 11 completed
frame 12 completed
frame 13 completed
frame 14 completed
frame 15 completed
frame 16 completed
frame 17 completed
frame 18 completed
```

```
amreeta@amreeta-desktop: ~/Downloads/sharpen-psf
frame 979 completed
frame 980 completed
frame 981 completed
frame 982 completed
frame 983 completed
frame 984 completed
frame 985 completed
frame 986 completed
frame 987 completed
frame 988 completed
frame 989 completed
frame 990 completed
frame 991 completed
frame 992 completed
frame 993 completed
frame 994 completed
frame 995 completed
frame 996 completed
frame 997 completed
frame 998 completed
frame 999 completed
starting sink file Trees2.ppm write
sink file Trees2.ppm written
amreeta@amreeta-desktop:~/Downloads/sharpen-psf$
```





Code Modifications

- The height and width of the given image was smaller and hence the transform was not applied to the entire image. Using GIMP, I found the size of the image to be 372x580 which I changed in the program.
- I also got a warning which said that the return value for the file read and write was not taken and hence I created to variable for the return value.
- In sharpen_grid code, multiple threads have been created and slices of the image is assigned to each thread for application of the transform. The loop used for allotting the image slices to each of the threads were disproportionate which resulted in application of the transform to some areas while the other areas were neglected. Hence the loop is modified to apply the transform uniformly.

Point Spread Function

- Point Spread Function is the impulse response of a linear space invariant system. It describes the response a specific pixel of an imaging system to an ideal point source.
- Output of an image is given by the convolution of the image with its Point Spread Function. In order to sharpen the edges of an image, we need to suppress the background while sharpening the edges.
- Since edges in spatial domain represent a sudden change which is equivalent to high frequency in the frequency domain, we need to use the point spread function of a high pass filter to get the desired output.
- Thus, the convolution of the image with the point spread function of a high pass filter will give us an output image with sharpened edges.

Code explanation

- Sharpen.c code takes the input image as an argument.
- It then defines a point spread function (PSF) which is the negative k-value divided by 8 of all the nearest neighbors of the pixel of interest.
- For each color channel i.e. R, G, B, these PSF values are summed up for all the nearest neighbors and multiplied with the pixel of interest in order to sharpen the image.
- Sharpen_grid code implements the same thing in a multi-threaded environment leading to faster results.

Sharpen_grid vs. sharpen

- In sharpen_grid code, multiple threads have been created and slices of the image is assigned to each thread for application of the transform while in the sharpen code threads have not been used.
- Hence on a multi-core system the transform will be applied to each slice of the image concurrently which makes it faster.
- Real-time image processing on embedded architecture which are data centric require huge amount of time and hence to make the system faster, the task can be split among multiple threads.

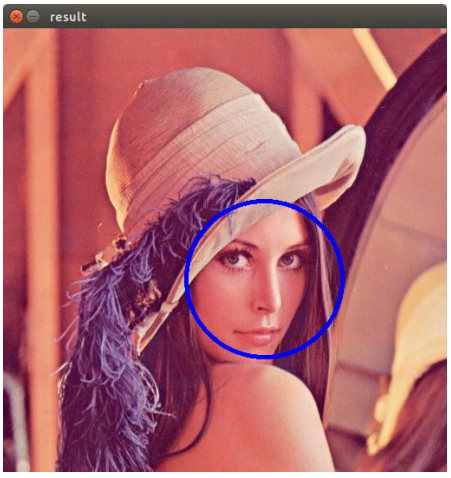
- This will allow the system to easily meet the real-time deadlines as the tasks work parallelly which will facilitate a faster approach.
- Thus, sharpen_grid is a better implementation for real-time frame transformation.

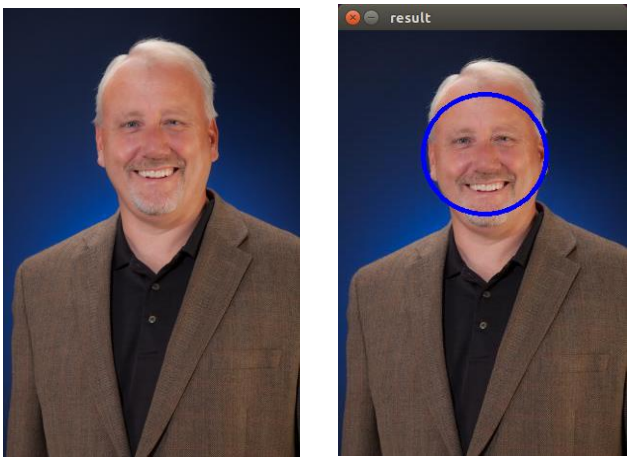
QUESTION 4

Build, run, test

```
amreeta@desktop:~/Downloads/src/faceDetect_sample$ make
g++ -O0 -pg -g   -c faceDetect.cpp
g++  -O0 -pg -g   -o faceDetect faceDetect.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/src/faceDetect_sample$ ./faceDetect --cascade="./haarcascade_frontalface_alt.xml" --nested-cascade="./haarcascade_eye.xml" --scale=1.3 lena.jpg

This program demonstrates the cascade recognizer. Now you can use Haar or LBP features.
This classifier can recognize many ~rigid objects, it's most known use is for faces.
Usage:
./facedetect [--cascade=<cascade_path> this is the primary trained classifier such as frontal face]
   [--nested-cascade[=nested_cascade_path this an optional secondary classifier such as eyes]]
   [--scale=<image scale greater or equal to 1, try 1.3 for example>
   [filename|camera_index]

see facedetect.cmd for one call:
./facedetect --cascade="../../data/haarcascades/haarcascade_frontalface_alt.xml" --nested-cascade="../../data/haarcascades/haarcascade_eye.xml" --scale=1.3
Hit any key to quit.
Using OpenCV version 3.3.1

Processing 1 --cascade=./haarcascade_frontalface_alt.xml
Processing 2 --nested-cascade=./haarcascade_eye.xml
Processing 3 --scale=1.3
Processing 4 lena.jpg
  from which we have cascadeName= haarcascade_frontalface_alt.xml
 from which we read scale = 1
before named window
Gtk-Message: 14:44:38.742: Failed to load module "canberra-gtk-module"
after
In image read
Before detectanddraw
bf detect multi
after
detection time = 197.548 ms
after
```

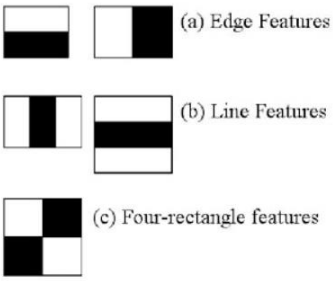
Picture example





Explanation of Haar Cascade

- Haar Cascade is basically an object detection algorithm that is employed to detect specified object in an image for which it has been trained.
- It utilizes machine learning based approach for which it requires a set of positive images i.e. images of object and a set of negative images i.e. images without the object which the cascade function uses to train itself and extract features of the object. It usually used in face detection algorithms where it detects the face and the eyes.
- Feature extraction is done in the following manner:
  The sum of pixel intensities under the white region are subtracted from the sum of pixel intensities under the black rectangle to obtain each feature which is a single value.



  However, among the huge set of the extracted features only a small subset of features is relevant. For example, for facial detection, the useful features are:
  - The region of the eye is often considered to be darker than the region of the nose and the cheeks.
  - Eyes are darker than the bridge of the nose.

  Other features of the face are not relevant in face detection.

- Hence, to select the useful features the concept of Adaboost is employed, wherein it determines the best threshold which will help in distinguishing between the objects of interest and the non-

objects. For quicker and more accurate face detection, a set of Haar features are grouped into a cascade classifier.

- These cascades of classifiers are applied one by one, where the first few stages will have lesser features and if an image fails this stage, it is discarded and if the images passes this stage, then the next stage is applied to it.
- The cascade classifier is trained with the use of a set of positive images i.e. images with faces and a set of negative images i.e. images without faces. For improved results, the number of stages can be increased, and better-quality images can be applied.
- Once the face is detected, it returns the position of the detected face.
- The advantage of this method of face detection is the improved speed and accuracy that is achieved.

Code explanation

- The code first loads the xml cascade classifiers which are used to detect the face and eyes.
- The image is then loaded using imread() function.
- The detectandDraw function is used to first detect the region of interest i.e. the face and the eyes.
- The detectMultiScale is used to first detect the face and draw a circle around it using the circle() function and then it is again used to detect the eyes and draw a circle around them.
- Finally the resultant image is displayed using imshow() function.

QUESTION 5

Build, run, test

```
amreeta@desktop:~/Downloads/simple-cv$ make
g++  -Wall   -c -o simplecv.o simplecv.cpp
g++ -Wall  simplecv.o -o simplecv `pkg-config --libs opencv` -L/usr/local/opencv/lib -lopencv_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/simple-cv$ ./simplecv
Gtk-Message: 18:03:41.007: Failed to load module "canberra-gtk-module"
hres=320, vres=180
```

180x320 Graphics

- In the code, resize function has been used for resizing the image resolution to 180x320.
- The drawMarker function has been used to draw the crosshair at the center of the image.
- The copyMakeBorder function has been used to draw a 4-pixel width border around the image

REFERENCES

- https://www.ibm.com/developerworks/cloud/library/cl-cloudscaling3-videoanalytics/#augmented-reality-example
- https://www.cs.auckland.ac.nz/compsci373s1c/PatricesLectures/Edge%20detection-Sobel_2up.pdf
- https://en.m.wikipedia.org/wiki/Sobel_operator
- https://en.m.wikipedia.org/wiki/Point_spread_function
- https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html
- http://www.willberger.org/cascade-haar-explained/
- http://www.cse.uaa.alaska.edu/~ssiewert/a485_code/
- https://docs.opencv.org
- RTES Exercise 4 by Amreeta Sengupta