University of Colorado Boulder

ECEN 5763 EMVIA Exercise 1

Dhiraj Bennadi

June 10th, 2022

Guided By: Professor Dr Sam Siewert

Code Repository

Github Link

The repository contains the source files, makefiles, output images and the report

1 Problem 1

The article focuses on the applications of video analytics and the resources needed for it to be successful.
I consider the following to be the main aspects of the article :-

1. Video analytics is defined as the analysis of digital video content from cameras or stored sequences of images and using of processing software such as OpenCV and similar tools to analyse the content.
2. Video analytics require a balanced data-centric compute architecture compared to the traditional systems due to the large amount of data and processing involved in functioning of the systems.
3. Video analytics is a very broad field of consisiting of various disciplines such as Image Acquisition and Encoding, CV, Machine Vision, Image Processing, Machine Learning, Real Time Interactive Systems, Storage, Networking, Database and Computing.
4. To understand and get an meaningful output from an image, video analytics should be considered as system design problem
5. The architecture of cloud based analytics can be divided into 2 segments, capturing using embedded intelligent sensors and cloud based analytics which is generally not possible on the sensor system.
6. The applications of video analytics are innumerable with major areas begin healthcare, transportation, retail, security,

Summary

Video analytics is an ever evolving field with emerging research, applications and adaptation in real world. The applications of it range from individual user to industry wide usage. More robust systems are being designed for better image capturing, processing and getting an output from the analysis. The whole ecosystem has to advance and keep upto to the trends of video analytics, such as algorithms, machine learning, sensor camera, processing hardware, cloud analytics and display. This field is a very promising one and that would benefit both the consumers and the suppliers.

2 Problem 2

2.1 Code Example: brighten.cpp

2.1.1 Output

Brighten factor = 2
Beta Contrast Factor = 90

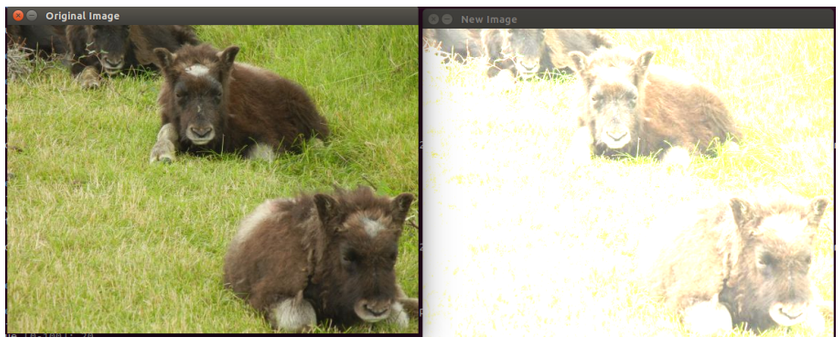


Figure 1: Brighten Image

2.1.2 Description

1. The example takes an image as input and displays a brightened image of the input image. Each pixel of the image is modified with values of alpha and beta provided as inputs from user. The alpha value corresponds to the brighten factor and the beta value corresponds to the contrast increase. Both of these would add to increasing the intensity of the image.

2. The following APIs of OpenCV have been used in the example:

   (a) **imread()** : Loads an image from the file system. It also specifies the color of the image.

   (b) **Mat::zeros()** : Returns a zero array of the specified size and type

   (c) **imshow()** : Displays an image in the specified window

3. The brighten example could be used to increase the intensity of images to get more pronounced features in the image. A real world example is the Night Mode in mobile phones which captures image in low light. It can also be used in capturing satellite images during night.

2.2 Code Example: capture.cpp

2.2.1 Output

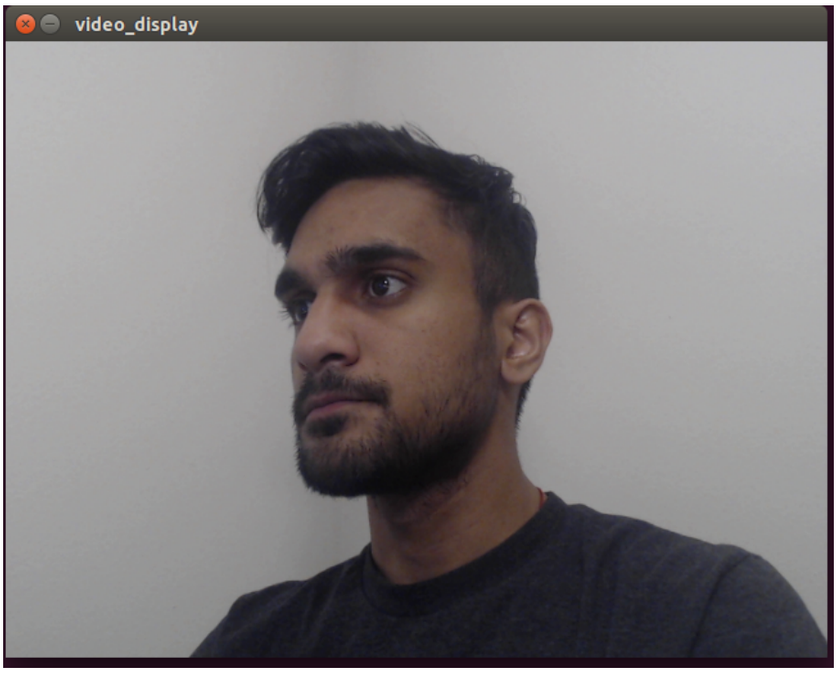


Figure 2: Capture Image

2.2.2 Description

1. The example involves code to capture frame from a live video stream. A video is individual frames being played at a certain rate usually described in fps. The frames are capture and are displayed on a window.

2. The following APIs of OpenCV have been used in the example:

   (a) **VideoCapture class** : Opens a camera for video capturing. The camera device is specified by an index on the system.

   (b) **Videocapture::isOpened()** Returns boolean value if the camera device is initialized

   (c) **Videocapture::read()** Grabs, decodes and returns the next video frame

3. The capture example can be used as a base example for any video capturing application. I used it for face recognition.

2.3 Code Example: diffcapture.cpp

2.3.1 Output

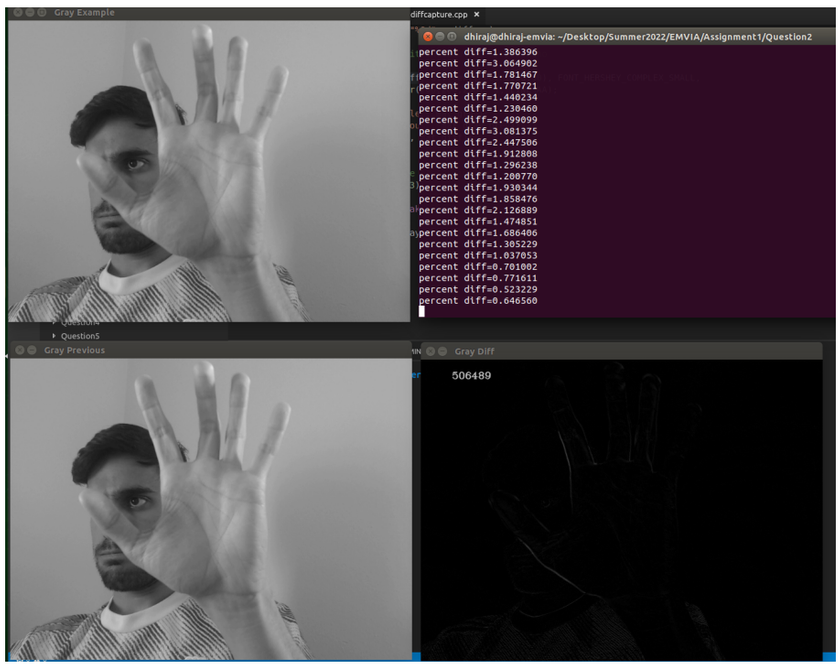


Figure 3: Capture Image

2.3.2 Description

1. The example captures the difference in image intensity and displays it in multiple windows. Initially an image is captured and then the image is compared with a newly captured image using the function absdiff. At all times, images are converted to grayscale for a single intensity value. The percentage of difference is also calculated and if it is greater than 0.5 then the value is displayed on the image window.

2. The following APIs of OpenCV have been used in the example:

   (a) **cvtColor()** : Converts an image from one color space to another

   (b) **clone()** : Creates a full copy of the image and the underlying data.

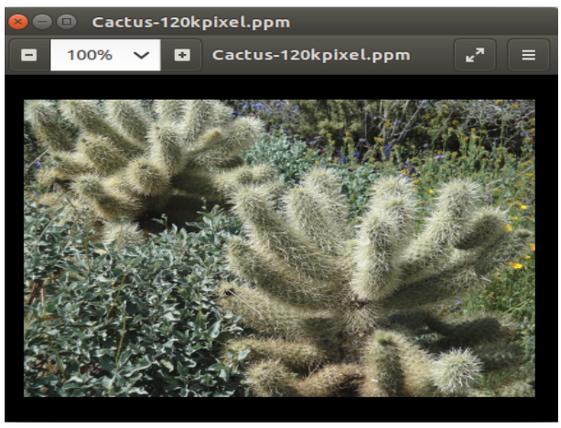
   (c) **absdiff** : Calculates the per-element absolute difference between two arrays or between an array and a scalar.

   (d) **putText** : Draws a text string on the image window

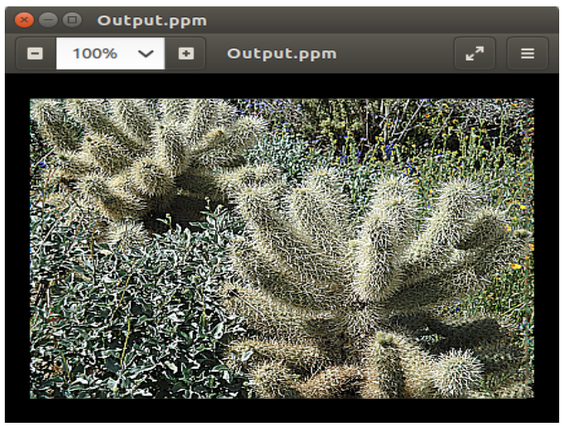
3. The example can be used in comparing photos and identify original and fake ones

3 Problem 3

3.1 Output of Sharpen on cactus Image



(a) Input Image for sharpen



(b) Output Image for sharpen

Figure 4: Sharpen PSF on Cactus Image

3.2 Output of Sharpen on Trees Image

The following changes were made on sharpen.c to work with the trees Image:

1. The Image Height and Width Macros were changes to accomodate the size of the Trees Image. This would enable the sharpen PSF to run on the whole Image
2. Makefile changes were made as the default one was for a different compiler

(a) Input Image for sharpen

(b) Output Image for sharpen

Figure 5: Sharpen PSF on Trees Image

3.3 Output of Sharpen_grid on Trees Image

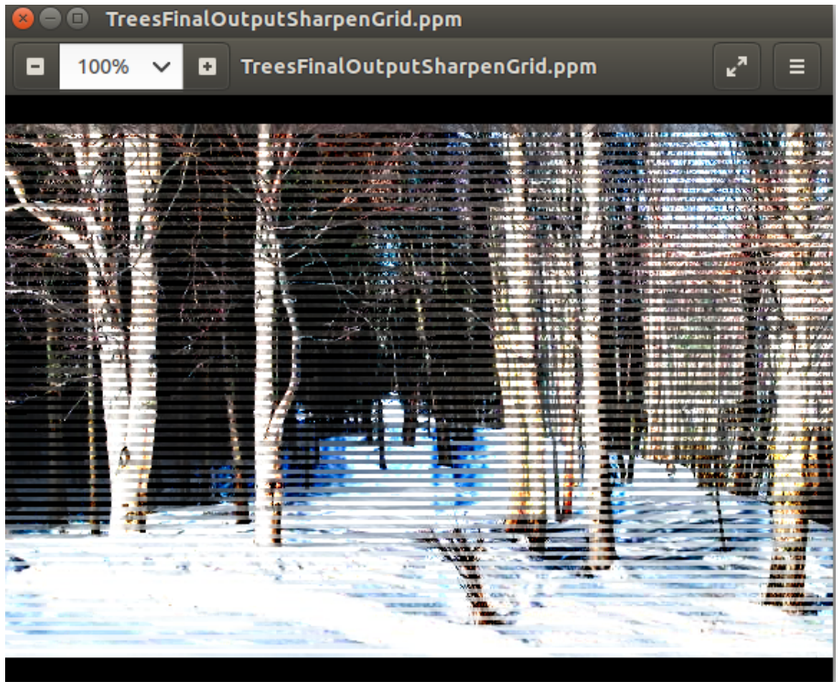


Figure 6: Crosshair Scale Image

3.4 Explanation of Point Spread Function

1. Point Spread Function: PSF is a response of an imaging system to a point source or a point object. When a delta function is passed through a linear system, the single non-zero point will be changed to different dimensional pattern, spreading the point and this impulse function is often referred to as PSF.

2. PSF can be designed by a variety of mathematical operations such pillbox, gaussian, square, edge enhancement, sinc

3. Image Convolution usually involves a large number of calcualtions. 3 approaches are used to speed things up, Small PSF usually 3*3, large PSF and FFT Convolution.

4. Based on the kernel used, various image processing techniques are yielded.

5. $$\begin{vmatrix} -k/8 & -k/8 & -k/8 \\ -k/8 & k+1 & -k/8 \\ -k/8 & -k/8 & -k/8 \end{vmatrix}$$
   The following PSF provides edge enhancement or sharpening operation. As the value of k is mage larger, the image shows better edge definition.
   Since the PSF function contains negative and positive values, the negative values are usually offset (shifted operation)

6. Humans usually distinguish one region from another based on the differences in brightness and color (Identifying its edges) and the PSF functions create an overshoot in the edge response, a situation of optical illusion.

3.5 sharpen vs sharpen_grid

3.5.1 sharpen.c

1. In the sharpen.c, the ppm image is read as a regular file capturing each pixel bytewise.
2. Each pixel, is altered using the PSF Array consisting of the edge enhancement values. If the pixel values are less than 0 or greater than 255 due to modification of PSF array values, the pixel values are set to the min and max value respectively.
3. The convoluted values are copied back to an output image provided by the user.
4. The input and the output images are treated as files and modified using file operations
5. A factor of 4 is used to sharpen the image

3.5.2 sharpen_grid.c

1. In the sharpen_grid.c, instead of considering the whole image as 2D array, image segments called slices are used depending on the number of row threads and columns threads provided by macros.
2. In this example a grid of approximately 100*50 pixel are worked upon and the PSF function is the operating function on all the image slices for about 1000 times.
3. A total of 48 threads are used for performing the convolution PSF.
4. On observing the output image, a sort of differentiation between the upper part of the image and lower part to the image is create. The upper part shows grid lines and the lower part is a smooth filtering option. This might also be due to the fact that the upper part of the image is has more elements compared to the lower part.

3.6 sharpen_grid.c Advantage

1. The technique is useful when working with a large number of images in a pipeline fashion. The scenario might be a stream of images (video) being sent to the processor for processing and there is limited memory for storing of images. The input images are sliced and are worked upon using threads and speeds up the process.
2. This technique might also mean images might be distorted if one of the threads doesn't function properly. Hence iterations of the PSF might be implemented to get a proper output.

4 Problem 4

4.1 Input

Figure 7: Face Detection Input Image

4.2 Output

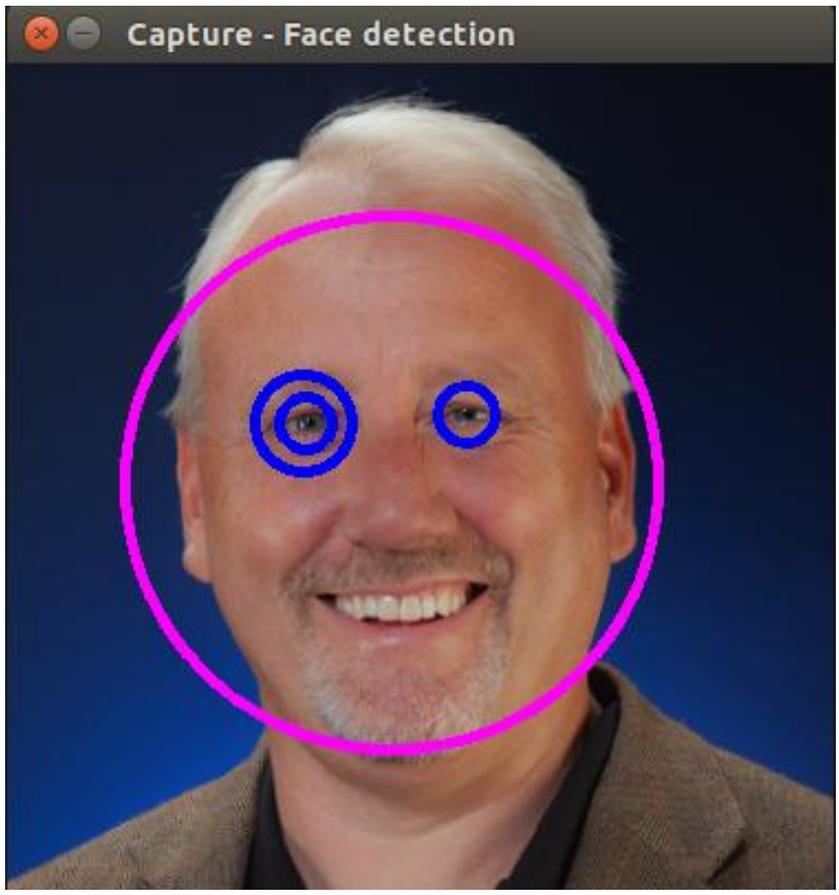


Figure 8: Face Detection Output Image

4.3 Explanation of HAAR Classifier

1. The main idea of identifying faces using HAAR classifiers is to use a set of features usually provided in the form of xml files and apply it to the image, and extract the features and classify it as a face or a non face.

2. The input image is scanned block by block in the form of a raster-scan fashion. At each pixel, the features are extracted and the features are classified as a face or a non-face

3. The HAAR features are square functions which are generally 2 valued filters (Binary). This is computationally very advantageous as the convolution technique involves only addition and subtraction which is cheaper than other mathematical functions.

4. The HAAR filters are applied at each pixel and a correlation values from the convolution is obtained. The higher the value of the correlation, it is more of a face feature. The filters range from variations in the x,y and Laplacian values.

5. Since the features are very large in number (6000), a cascade of classifiers is applied in the form of stages of classifiers reduced the number of calculations.

6. Once all the cascade of feature classifiers are passed, the coordinates of the rectangle of the face block is provided as output

7. The face can be used as an input image to the eyes classifier which runs a similar set of features to detect the eyes, and provide an output

5 Problem 5

5.1 Output

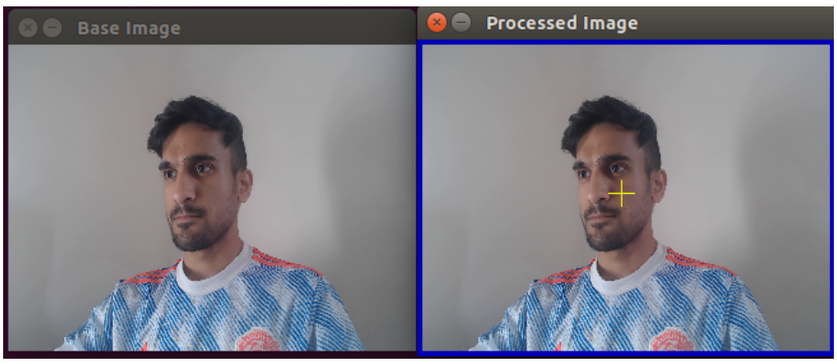


Figure 9: Crosshair Scale Image

References

[1] OpenCV Documentation

[2] Paper on Cloud Analytics

[3] Video on explanation of HAAR Classifiers

[4] Repository provided by Dr. Sam Siewert