Project Report

Shahzeb Naveed, Omar Ikram and Hamza Naqi

January 6, 2016

Project Title: Image Editor

1 Introduction

We are living in an era where whether it be social networking or it is online marketing, The businesses around the globe are trying to the best of their abilities to leverage online media in order to build and reinforce brands that assist in generating new avenues for lead generation and revenue.One more aspect they take into consideration is the cost of evolving media technologies, the use of images, especially in the cyber world, has witnessed a gradual increase over the years. It is not hard to guess why pictures have become an integral part of media in general and social media in particular. Photos give a boost to the content matter, It has become tremendously popular and trending that photos are integral part of multimedia message or web content or Facebook/twitter post. The kind of photos and pictures used in promotion or marketing, give a cue to the kind of organization or person involved and if marketing is involved then quality of image is something on which there is nothing to compromise. The true potential of a photo can be showcased to the viewers through image processing tools available through image editing software.

With the rapid development of new IT technologies around the globe, it has become an essential need in everybody's life. In the present times pictures/images are something that gives a new dimension to how you analyze things.Photos are vital to firms as well as individuals.

For individuals, photographs served as important medium in saving momentous memories of friends, families, peers as well as loved ones. Moreover, firms used photographs as tools for their advertising and marketing campaigns.Nowadays, consumers are more inclined to captivating photos, therefore companies ensure that their adverts has alluring photographs. As expected advertisement that does not have photos will not draw the attention of consumers because they don't like reading plain text only. Other than adverts, images are also needed for other purposes such as press releases, website creation, product launches and much more. To make sure that photographs efficiently draw the attention of customers, business enterprises should possess quality photos, therefore they should make use of photo editor software.What photo editor software actually is? Well With the use of photo editor, one can easily remove blemishes and flaws on photos like pimples and wrinkles. With photo editor, you can easily adjust the color, brightness and contrast, hence producing good quality photographs. You can even use the other functions in your photos such as sepia, blurred effects, night settings and etc. Apart from using photo editor, you can reap numerous benefits in getting the services of photo editing providers. Keeping in view the wide range of its application, We intend to develop an image editing software in which the user can input a picture and edit it using different image processing tools. The options we plan to include are zoom, crop, flip (horizontal/vertical), rotate, brightness, contrast and filters like grey-scale and sepia etc. We will make these by implementing suitable data structures and developing efficient algorithms. We also plan to design a user-friendly interface for the program.The idea of the project is to replicate the functions performed in existing image editing applications to the optimum efficiency. Some paid software programs are much complicated even for an experienced person. Amateurs will not be able to understand and perform most of the tasks or use necessary tools meant for the purpose. However, Our aim is to model a professional photo editor that can do wonders to your photograph using various techniques that you may not even have heard of. The purpose of this interim report is to implement the algorithms required to design the photo editor and analyze what problems might come along while designing it.

2 Implementation

2.1 What to Choose?

Initially we had two ways to go about it. The Open CV (Open Source Computer vision library) and the second one being File Handling. What to choose? Choosing file handling will result image file is being opened in txt format (the file contains integer values of all pixels for all the three channels). It also comes with a limitation that we can only operate through ppm format which is not commonly used as it does not have much practical applications as compared to png and jpeg which utilizes lossless compression, meaning no image data is lost when saving or viewing the image. Both of them are universal format that is recognized by the World Wide Web consortium, and supported by modern web browsers. In order to work with these format we will be opting for OpenCV to read and write the converted image. Therefore in order to read and write the converted images in an object of class 'Mat'.OpenCV is a computer vision library whose main focus is to process and manipulate the information. Therefore, the first thing we should be familiar with is how OpenCV stores and handles images.
As the described earlier the image is stored in the form of matrix. Hence each each element of the matrix is a structure containing three values i.e R,G,B ranging from 0 to 255. Therefore each element is in itself a pixel of the image. Something we really want to avoid is to decrease the speed of our program by making unnecessary copies of potentially large images. To tackle this issue OpenCV uses a reference counting system. The idea is that each Mat object has its own header, however the matrix may be shared between two instance of them by having their matrix pointers point to the same address. Moreover, the copy operators will only copy the headers and the pointer to the large matrix, not the data itself. One of the main problem we ecounterd while working along OpenCV was linking the OpenCV with visual studio. After researching for the possible solution on the internet. Since OpenCv was an external library so in

order to use in visual studio we have to link it in it's settings. There are four simple steps to go about it. They are described as below.

1. Specify to the compiler where the required header files are.
2. Specify to the compiler where the library files are.
3. the compiler which library files to use.
4. Copy DLLs to the same folder where exe is.

Initially there were errors like 'Cannot open the core.hpp' but it was easily removed by correctly linking the library folder but it was after that we started getting errors like 'x64 module conflicts the x86 target machine' So after hours of debugging and research it comes to our knowledge that this happens every time when you try to include wrong libraries (The error is explicit, you are trying to link libraries that were compiled with different CPU targets. An executable image can only contain pure x86 (32-bit) or pure x64 (64-bit) code. In order to rectify this we changed the configuration of visual studio and set it to produce a 64-bit application.
When we succeeded in solving this error, we encountered another error relating to some unresolved externals. We looked it up on the internet but we couldn't find any solution to it. It was then we decided to try an updated version of visual studio and it worked out fine. The reason of prolonged error was the fact that the OpenCV version we were using was incompatible with the version of visual studio.

2.2 Selection of Data Structure

Since the image is being stored in the form of a matrix. Each element of the matrix is actually storing the pixel of the image.There are two potential data Structures that can be used

- Multiple linked list
- 2-D Array

In order to traverse the matrix for various functions it is most efficient to use 2D arrays as it can access required pixel in O(1) time while if we would have used linked list the same operation would take O(row index + column index) hence making it less efficient. Using stack would be out of question as it would require all the preceding elements to popout for the using the required element/pixel. However in some functions stacks are more suitable then the others such as rotation of the image we will discuss it in detail later section **Therefore we used 2-D array to implement most of the above mentioned functions**

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

2.3 Pseudocode with Explanation of Algorithms

.

Our image editing software encompasses the processes of:

- Rotation (Clockwise and anticlockwise)
- Flipping (Horizontal and Vertical)
- Dither
- Noise Reduction
- Edge Filter
- Sharpen
- Blur
- Brightness
- Contrast
- Lightening of image
- Darkening of image
- Overlay
- Backlight
- Threshold
- Invert Color
- Grey-scale
- Instagram-style Effects
- **Sharpen:** In order to sharpen an image, we have to differentiate the pixels lying on edges from those who are farther away from the edges. So for this, what we do is that we scale the RGB values of each pixel by multiplying by a constant, preferably the integer 5 and then subtract the RGB values each of the adjacent pixels. We learnt this from a very useful book on Computer Vision and Algorithms by Richard Szeliski. As we have to visit every pixel only once in the process, hence the time complexity of the algorithm is O(n).
- **Blur:** In order to blur a given image we traverse through the matrix. As we arrive at each pixel, the RGB channels of the center pixel will take the average value of the corresponding channels of the adjacent pixels. Thus a smoothing effect is produced and consequently the image loses details and it gets blurred. As we have to visit every pixel only once in the process, hence the time complexity of the algorithm is O(n).

- **Contrast** When we apply contrast to an image, we actually lighten up the light areas and darken the light areas. It is also a special type of sharpening effect in which the edges become for distinguishable. We actually enhance the local gradients of all the pixels that eventually enhances the contrast. We calculate the quantity f = (259 * (contrast + 255)) / (255 * (259 - contrast)); The user sets the contrast and the we multiply the pixel values plus 128 and then minus 128 in order to get enchanced contrast.

- **Lightening of image** It is somewhat similar to brightness except that here, we 'add' a constant value to each pixel instead of multiplying it. Unlike brightness here the effect would be uniform on each pixel as addition would not depend on pixel's initial intensity. Hence the transformed image would appear as if there's a white overlay over the original image. As we have to visit every pixel only once in the process, hence the time complexity of the algorithm is O(n).

- **Darkening of image:** It is very much similar to lightening's algorithm. The only difference is that here we subtract a constant value from each pixel. Hence the transformed image would be darker then the original. As we visit one pixel only once in the process, hence the time complexity of the algorithm is O(n).

- **Backlight:** The concept of backlight is basically to lighten up the shadowy or darker areas while the lighter or brighter areas of the image remain the same. So for implementing this, we take average value of the RGB channels of each pixel and store the average in a variable. After doing this, the user sets a threshold value that decides what are the dark areas and what are the light areas, For instance, we set a threshold value of 50, so the pixel value having average RGB values less then 50 will be considered dark. Those pixel can be adjusted using brightness function according to user's desire. Time Complexity of this algorithm is also O(n).

- **Flipping and Rotating:** Changing the orientation of an image is very commonly used. The algorithm is quite basic. We simply read a pixel from the input image and place it according to the effect we want to see. As we visit one pixel only once in the process, hence the time complexity of the algorithm is O(n).

- **Overlay:** In overlay, we put a layer of one image over the other. It actually uses the concept of blending. What we do is that the user gives a value say x ranging between 0 and 1. The variable x is multiplied to one image and the expression (1-x) is multiplied to the other image. Thus, if the value of the variable is 1, the first image gets stored in the resultant matrix, if it's zero, the other image gets stored. If x is somewhere between 0 and 1, the resultant image will be an overlay of the two images. This is also the basics of Image Compositing. All the channels of every pixel are treated likewise in this algorithm.

- **Noise Reduction:** Noise is a very common problem in digital images. Noise is actually the unwanted patches of colors or artifacts that are not part of the actual image. So in order to remove these, we use an algorithm very similar to the algorithm used for blurring. But in this case, we put the adjacent pixels of every pixel into an array. We then apply quick-sort on the array and then find the median of the array. We then apply the median to the corresponding channel of the centre pixel. This is done for all the channels of the pixel. This magnificently reduces noise in an image. It also operates in O(n) time.

- **Vignette:** Vignette is a very common filter found in almost every photo-editing software these days. What it does is that it puts black shadowy effect near the boundries of the image giving a very cool look. The algorithm includes determining the maximum distance the centre of the image has from the edges. We then iterate through every pixel of the image and determine the ratio of its distance to the maximum distance of the centre. We then multiply each channel of the pixel by (1-(1-ratio)). This means that the area nearer to the centre of the image will be bright and there will be a shadowy effect the on the edges.

- **Dither:** Dithering is when the computer system tries to approximate a color from a bunch of other colors when the required color is unavailable. This happens when we want to convert a color from a 16-bit image to 8-bit or from 32-bit image to a 16-bit image. In such cases, all the previous colors are not necessarily available in the new format. So we have to develop an algorithm that approximates the color of the pixel. For this, we use a very well researched Kernel Matrix and apply it to every pixel. It also takes O(n) time to run.

- **Edge Filter:** In this case, we simply subtract the previous value in the row and add it to the next value in the same row (we also do this for the column) and assign it to the centre pixel. If the difference is large enough, then that means the pixel is lying on the edge and the difference would be very intense. Hence the centre pixel would be very bright while the pixels that do not lie on the edges will appear very dark. All the channels are treated likewise in this algorithm which takes O(n) time to run.

- **Threshold:** The threshold effect is basically to convert an image to a pure black and white image. As explained earlier that how we set a parameters from which we decide what are the light areas of an image and what are the dark areas of an image. We call these parameter values as threshold values. So if a pixel has average RGB value lesser then threshold value, we set its RGB value to 0. Similarly if an image has a value greater then the threshold, we set its value to 255. Since it is done by traversing the matrix where we visit one pixel only once. Its time complexity would be O(n).

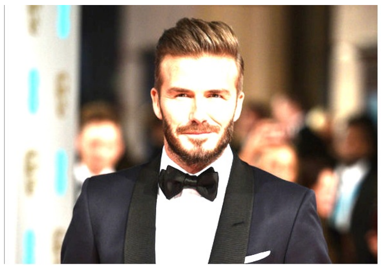
- **Grey-scale Filter** In grey-scale filter, what we do is that we actually convert a colored image and transform into a greyscale image. Since a colored image consists of three channels so we take the average of the three channels i.e R, G and B of each pixel and assign back that calculated average to each of the three channels of the same pixel. Consequently, all the channels have equal intensities and thus no color other than different shades of grey are produced. *Picture of algorithm Visual studio wali*

- **Instagram-style Effects** We have also successfully developed algorithms for different color filters somewhat close to what Instagram or Retrica offers. In this case, we simply use our own Vignette Filter and then vary the RGB channels of all the pixels with different RBG combinations that produce some very beatiful color effects like Sepia, Aden, Slumber, Moon, Nashville and many more.

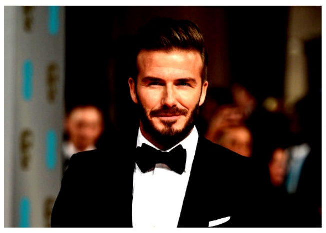
3 Results

In order to showcase the results of different algorithmic functions I have a taken a single image shown above which is in its unedited form , after applying different functions on it. Following are the results:

- Brightness



- Contrast



- Blur

- Darkness

- Sharpness

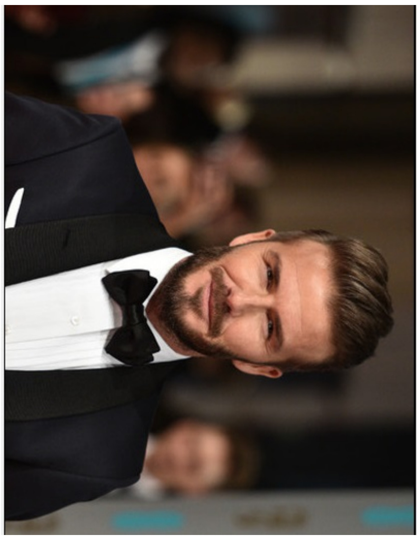
- Edge Filter

- Flip

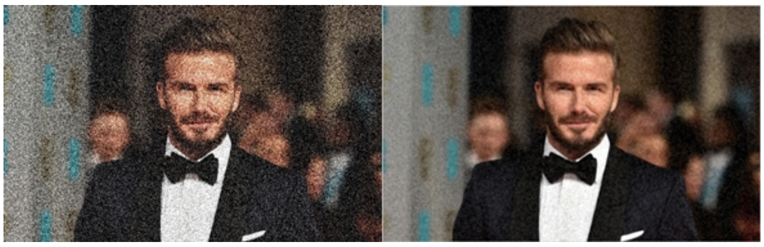
- Threshold

- Instagram-style Color-Filter

- Rotate



- Noise Reduction



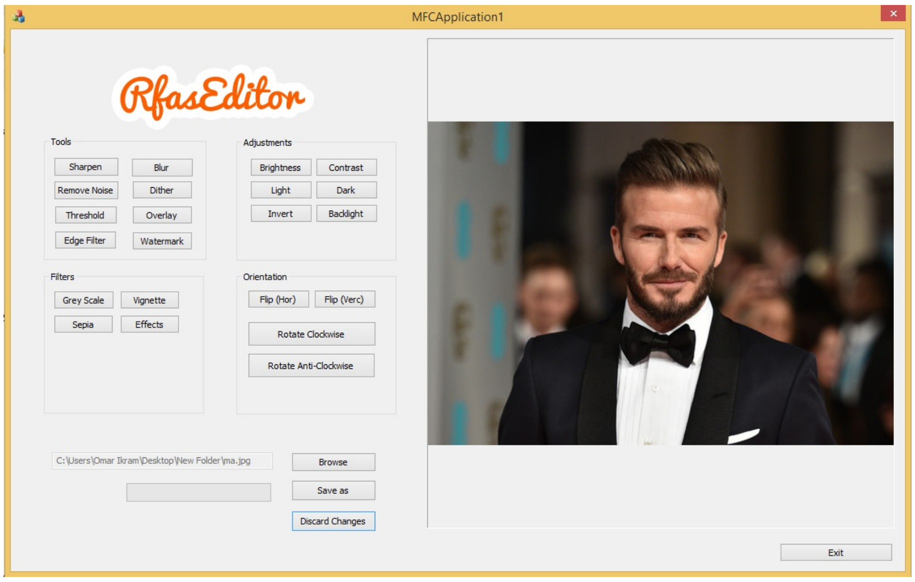
- Overlay

- Greyscale Filter

- Sepia Filter

- Vignette

We also made a very user-friendly interface for the appllication using **MFC Application**.



C:\Users\Omar Ikram\Desktop\New Folder\ma.jpg

4 Conclusion

This project was successfully implemented since the the accuracy and the quality of the results pretty much matches the results obtain by existing professional photo editors. The benchmark we set for ourselves also to replicate the behavior of these well known applications. To be very honest this project is more exciting then we anticipated it to be. Working with Open-CV was challenging but we learn quite a few things from it. Most of the test images produced very pleasing, readable images except the overlay function where we feel there is plenty of roam for improvement. Since all the algorithms runs in O(n) in all cases As one looks about it is not much disconcerting but in future we are intending to reduce this running time. The immediate future plan is to make a android application of this program.

References

[1] The Mat Basic Container.com

[2] Computer Vision and Algorithms by Richard Szeliski

[3] Greedy Algorithm for Local Contrast Enhancement of Images by Kartic Subr, Aditi Majumder and Sandy Irani

[4] An Empirical Identification Method of Gaussian Blur Parameter for Image Deblurring Fen Chen, Member, IEEE, and Jianglin Ma

[5] Image Sharpening - Making it real sharp by Jeff Schewe

[6] Deblured Gaussian Blurred Images Mr. Salem Saleh Al-Amri, Dr. N.V. Kalyankar and Dr. Khamitkar S.D

[7] EE3414: Image Filtering by Yao Wang Polytechnic University, Brooklyn, NY11201

[8] www.stackoverflow.com

[9] Dartmouth's document on Image Processing

[10] www.lodev.org

[11] http://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/