Photo Editor Mobile Application

Avram Georgiana Maria Image Processing Course, Student

Department of Computer Science

Technical University

Cluj-Napoca, Romania  
 g.avram.m@gmail.com

*Abstract*—The purpose of this application is to provide a user-friendly way to enhance or modify the parameters of a photo.

Keywords— image processing, image enhancement, mobile, application, algorithm, filters

# Introduction

Digital image processing is a field in computer science that puts a live effect on many other fields and has various applications from image sharpening and restoration, video processing to medical field, pattern recognition or even robot vision. Back in 1816, the idea of photos was revolutionary and since then there have been made impressive developments resulting in the technology we know today. The last decades were spent trying to bring hardware and software together in order to increase the quality of images, enhancements and much more than that, namely, extracting useful information.

In digital processing, images are matrices containing pixels, each pixel being described by three components: R (red), G (green) and B (blue). By modifying these components, we can obtain different effects on the image: conversion from RGB to Grayscale, from RGB to a different color model like HSV, changing the colors that give a different touch to the photo. However, algorithms can become more complex than that when putting Artificial Intelligence into equation, requiring powerful computer systems and may appear limitations.

This paper aims to present a mobile application that implements image processing algorithms that allow photo editing such as: contrast, brightness and color adjustments, applying different filters. The context of the developed system evolves around personal use, bringing functionalities to a regular user. Section I describes the simple introduction about image processing. Section II brings into light related work that is relevant to the presented subject. Then, section III contains the proposed solution to accomplish the above stated objective. In section IV the results are presented according to the specified solution, followed by the final conclusions in section V.

# Related work

## Image Filtering/Smoothing Algorithms

The purpose of smoothing is to reduce noise and improve the visual quality of the image. A variety of algorithms i.e. linear and non-linear – algorithms are used for filtering the images. Image filtering makes possible several useful tasks in image processing, like reduce the amount of unwanted noise in a particular image. Another type of filter can be used to reverse the effects of blurring on a particular picture. Non-linear filters have quite different behavior compared to linear filters, as the output does not obey the principles outlined earlier, particularly scaling and shift invariance.

Linear smoothing is the most common, simplest and fastest kind of filtering. The linear filter replaces each pixel with a linear combination of its neighbors and convolution kernel is used in prescription for the linear combination.

A box blur, also known as “moving average”, is a simple linear filter with a square kernel and it contains all the kernel coefficients equal. It is the quickest blur algorithm, but it has a drawback i.e. lacks smoothness of a Gaussian blur. The algorithm is based on a fact that sum S of elements in the rectangular window can be decomposed into sums C of columns of this window:

## Image Sharpening Using Unsharp Masking [1]

Image enhancement is considered as one of the most important techniques in image processing. Image sharpening is done using unsharp masking for contrast enhancement. The basic concept of UM is to blur the original image first, then subtract the blurred image from the original image itself. As the final stage add the difference to the original image. The linear unsharp filtering approach is used to enhance the noisy image using the high pass filter. Unsharp masks are very applicable for sharpening images. But too much sharpening can also lead to artificiality in the image losing its natural look. This method has two major drawbacks such as the contrast in the darker area is enhanced much deeper than the light area. Due to the issues the image loses its originality in most cases. Among all the image sharpening approaches used this UM approach is the easiest and simplest.

*sharpened = original + (original − blurred) × amount*

## Blending images

Algorithm for aligning images and stitching them into seamless photo-mosaic are among the oldest and most widely used in computer vision. Creating high resolution images by combining smaller images are popular since the beginning of the photography.

Even in medical imaging for better clinical diagnosis, a composite image needs to be formed starting from its component images.

This operator forms a blend of two images of the same size. Similar to pixel addition, the value of each pixel in the output image is a linear combination of the corresponding pixel values in the input images. The coefficients of the linear combination are user-predefined and they define the ratio by which to scale each image before combining them. These proportions are applied such that the output pixel values do not exceed the maximum pixel value.

P1 and P2 are the two input images. In some applications, P2 can also be a constant, thus allowing a constant offset value to be added to a single image.

X is the blending ratio which determines the influence of each input image in the output. X can either be a constant factor for all pixels in the image or can be determined for each pixel separately using a mask. The size of the mask must then be identical with the size of the images.

## Image Filters Algorithm

Image filtering allows to apply various effects on photos.

The type of image filtering described here uses a 2D filter similar to the one included in Photoshop as Custom Filter.

Image is divided into pixels or say 2D Array (Matrix) or can be defined as the mathematical function f(x, y) where x and y the two coordinates horizontally and vertically.

*Gray Filter*

The formula that must be computed is:

*Intensity = (oldRed + oldGreen + oldBlue) /3*

Then each component R, G, B is assigned this new intensity.

*Sepia Filter*

The formulas that must be computed is:

newRed = 0.393 × oldRed + 0.769 × oldGreen + 0.189 × old Blue

newGreen = 0.349 × oldRed + 0.686 × oldGreen + 0.168 × old Blue

newBlue = 0.272 × oldRed + 0.534 × oldGreen + 0.131 × old Blue

*Sketch Filter*

Here we have some random scale factor set to 100 and checked if intensity is greater than the scale factor then applies the color white, else apply color black.

## Resizing Of Binary And Grayscale Images Using a Logical Transform

Image interpolation, both down sampling and up sampling, is necessary when resizing the data to match either the specifics of the communication channel or the output display. While it is more efficient to transmit low-resolution versions (often combined with data compression) to the client, an approximation of the high-resolution original may be necessary when presenting the final visual data.

To resize data, several techniques have been developed. The most basic method commonly used is nearest neighbor, a procedure that is not only fast, but does not introduce any artificial data into the final output. However, despite the speed with which it can be calculated, this procedure suffers from the fact that the resulting image often contains block artifacts, which are not only very visually noticeable, but typically also can drastically negatively affect error calculations used to compare methods.

*Nearest Neighbor Emulation*

The following figure (figure 3) shows the block representations for a 2x2 window (input) and a 4x4 window (resulting output from input being scaled up by a factor of 2). The pixel values are labeled by the variables A, B, C, and D, showing how the values are related in each window during the nearest neighbor scaling.

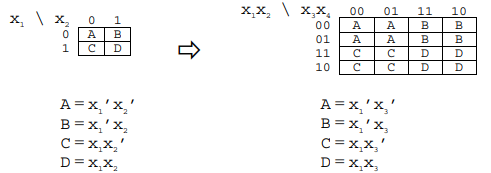


Figure 1 Scaling up by 2

Because the two window sizes are related, it is possible to develop a connection between the terms in the small window and the terms found in the larger nearest neighbor scaled window. For a factor of two (the same as in figure 3) and for a factor of four (2x2 window scaled to 8x8 window), this association is shown in the following diagram.

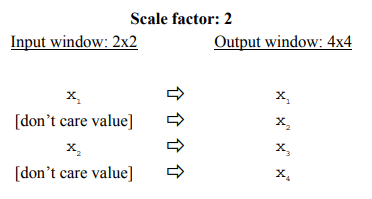


Figure 2 Association between literals in terms of input and output windows

Based on this relationship, the algorithm for performing nearest neighbor emulation using the logical transform is presented in figure.

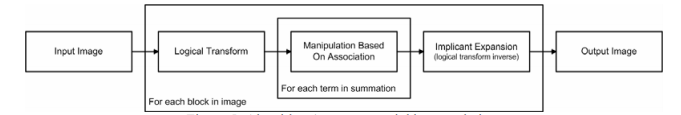


Figure Algorithm

# Proposed solution

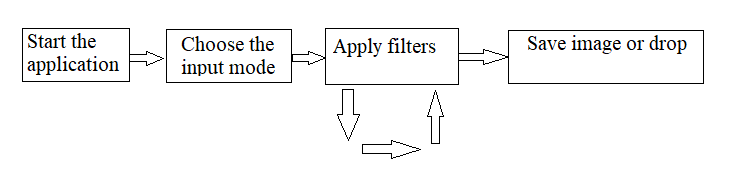
## Chosen method

The purpose is to implement a mobile application for the Android platform that is able to perform adjustments on images. The algorithms that perform the image processing part of the editor will be based on the OpenCV (Open Source Computer Vision) Library, which has more than 2500 optimized algorithms, including comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These can be used to remove red eyes from images taken using flash, find similar images from database, recognize faces, features and many more.

The software that I will be using to develop the code is Android Studio, primarily using the Java language. Being an application that is intended for regular users, the graphical interface is of great importance, because the image processing algorithms implemented in the back should be used very easily through the interface. My intention is to provide a clean and simple interface. When the application starts, the user is welcomed with a page that allows the choice between capturing a photo with the camera or just choosing one from the gallery. Afterwards, on the bottom of the page will appear the functionalities in categories. “Filters” section containing predefined filters that can be applied on any image, mainly filters that alter the colors. Next section is “Adjustments” that get an input from the user, a value, and give it to the methods that perform the algorithms on the input image. The implemented algorithms are the following: exposure, contrast, adjust (flipping, crop), sharpen, clarity, saturation, tone (highlights are dimmed, shadows are enlightened), white balance (temperature, tint), vignette, grain, fade, split tone (shadows tint, highlights tint). The section is Undo/Redo that cancels the last action or not.

The last operation would be to save the image in the gallery, or just drop it. Another functionality would be sharing on Facebook.

## Implementation flow



## Implemented algorithms

Images are defined as a two-dimensional function F(x, y), where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x, y) is called the intensity of the image at the point. In other words, an image can be defined by a two-dimensional array specifically arranged in rows and columns.

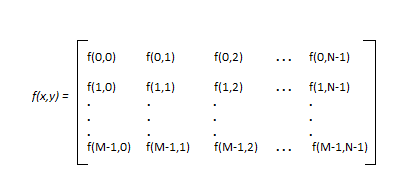


Figure Representation of an image

There are a few types of images:

* BINARY IMAGE - contains only two-pixel elements: 0 refers to black and 1 refers to 1, also known as Monochrome.
* BLACK AND WHITE IMAGE
* 8-bit COLOR FORMAT - the most common format, having 256 different
* 16-bit COLOR FORMAT

1. *Presets*

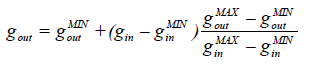
The presets are a combination of the algorithms below. Changing the light intensity, the colors we can achieve very different looking images.

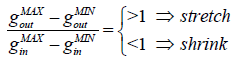
1. *Resize*

One of the simpler ways of increasing image size is nearest-neighbor interpolation, replacing every pixel with the nearest pixel in the output; for upscaling this means multiple pixels of the same color will be present. This can preserve sharp details in pixel art, but also introduce jaggedness in previously smooth images. 'Nearest' in nearest-neighbor doesn't have to be the mathematical nearest. One common implementation is to always round towards zero. Rounding this way produces fewer artifacts and is faster to calculate.

1. *Contrast*

Contrast enhancement processes adjust the relative brightness and darkness of objects in the scene to improve their visibility. The contrast and tone of the image can be changed by mapping the gray levels in the image to new values through a gray-level transform. The mapping function reassigns the current gray level GL to a new gray level GL

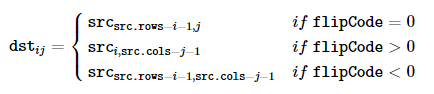




Stretch means high contrast, while shrink means fading the details.

1. *Flipping*

OpenCV provides the flip function that flips an image.



1. *Rotate*

OpenCV provides the rotate function that rotates an image in multiples of 90 degrees*.*

1. *Crop*

The algorithm needs two points: the upper left point and the bottom right point of the rectangular that defines the cropped area of the image. The result will contain all the pixels delimited by this area.

1. *Burring*

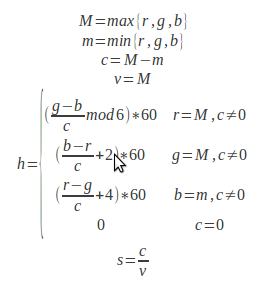
To blur an image, the color of each pixel is changed, setting the new color of the pixel per the average color of the pixel surrounding it.

Traverse through the pixel row-by-row to compute the average of the pixel with its horizontal neighboring pixel within the stated radius.

After that, use the result from first iteration, traverse through the pixel column-by-column to compute the average of the pixel with its vertical neighboring pixel within the stated radius.

1. *Saturation*

This algorithm implied to convert the RGB image into HSV. This color space tries to mimic the way the humans perceive color. The H component (hue) is the color itself, independent (invariant) of illumination, the S component (saturation) is the color’s “purity” (how well defined the color is), and V (value, or intensity) is the brightness. The final step is to convert back to RGB using an OpenCV function.



1. *Tone*

Gamma correction is a non-linear adjustment to individual pixel values. It can be used to correct the brightness of an image. The mid tones and the black tones are the most affected by this approach, briefly, it can brighten up dark spots without affecting the light patches.



Where, γ is a positive coefficient: <1 means encoding/compression and >1 means decoding, decompression.

1. *White balance*

This filter modifies the overall feeling that is give by the image. If the blue channel has the biggest value for every pixel, it means that the image has a cooler tone. The other way, if the red channel has the biggest value for every pixel, the image has a warmer tone.

1. *Vignette*

Vignetting is a reduction of an image’s brightness or saturation toward the periphery compared to the image center. The idea behind relies in creating a mask that will be applied on the image. This mask is defined by three elements: center point, radius and power. Center point is the brightest pixel in the mask, radius tells the maximum distance until the pixels get a lot darker and power gives the intensity of the effect. Then, iterating over the initial image, we multiply the blue channel with the value in the mask.

1. *Split tone*

The toning of a color image that allows to add a dominant color to the image. It is possible to tweak this dominant color in the image highlights and/or shadows.

# RESULTS

As was mentioned in the above sections, the application consists of a set of filters and edit functionalities.





Figure 8 Equalize + Negative Brightness + Rotate + Flip



Figure 9 HB1 + Saturation



Figure 10 Canny



Figure 11 Sepia + Contrast + Saturation + WB + Flip

The above figures are a few results after applying the proposed functions. As it can be seen, the results are quite pleasing and aesthetic.

Although, the algorithms are not so complex, the computations increase with the dimension of the picture. Taking into consideration that this is a mobile application, the device does not perform so satisfying in terms of speed. My approach was to integrate C language in Android. I managed to write the functions without causing the application to crash.

# Conclusions

After implementing this project, the conclusions are that image processing is a very vast domain. With only a few filters, image quality and aspect can be increased considerably. On the other hand, it requires computational power as dimensions of the pictures increase or the complexity of the algorithms applied.

# References

1. J.N, Archana. (2016). A Review on the Image Sharpening Algorithms Using Unsharp Masking. IJESC. 6. 8729-8733.
2. Parker, James. (1997). Algorithms for Image Processing and Computer Vision.
3. Gupta, Gaurav & chandel, ruchika. (2013). Image Filtering Algorithms and Techniques: A Review.
4. https://medium.com/better-programming/android-image-color-change-with-colormatrix-e927d7fb6eb4
5. Ethan E. Danahy, Sos S. Agaian Algorithms for the resizing of binary and grayscale image susing a logical transform
6. https://medium.com/better-programming/blurring-image-algorithm-example-in-android-cec81911cd5e
7. https://opencv.org/about/
8. https://android.jlelse.eu/a-beginners-guide-to-setting-up-opencv-android-library-on-android-studio-19794e220f3c
9. https://www.geeksforgeeks.org/digital-image-processing-basics/
10. Sergiu Nedevschi, Image Processing Course, Technical University Cluj-Napoca