ENSC 424 - Multimedia Communications Engineering Project Report

December 5, 2015

Lestley Gabo
Anmol Bhullar
Amandeep Singh Mand
Apassara Jurapan

Philter

Abstract

In this project, we are making an images filter Android app, Philter, that uses formulas and algorithm to manipulate digital images with straightforward operations and user interface. Images manipulation by adding filters is a popular application people use before punishing their images on social media. It is also playing a significant role in computer vision and pattern recognition since it will provide a clearer image for human vision to process the images. We implement Grayscale, Sepia, Gamma Correction, Changing Brightness, Erosion and Dilation filters for the apps using mathematical manipulation and of the RGB and applying algorithm.

Introduction

Our objective was to create an Android app that will allow the user to alter newly taken images and/or alter existing images inside their Android device. The inspiration for our project came from the manipulation of the Lena and Barb images during our class assignments. We were also curious as to how the filters worked or how they were done. As we created our project we kept in mind that our goal was to of gain a deeper understanding of multimedia communications through image manipulation.
When creating the framework of the Android app used several resources such as android-arsenal.com [1]. Using Android Arsenal we were provided with the necessary Android Libraries. Enabling us to simplify the user interface of our app instead of creating it from scratch. As for the creation of the filters we used several websites, one example is the website jhlabs.com [2]. JHLABS had tutorials on how to manipulate images and we could benefited from referencing those tutorials into our project. An example of a tutorial in the JHLABS website was an article labeled “Blurring for Beginners.” It contained explanations for blurring images and goes into detail about translating that into Java language. We used these two websites mentioned above and several more listed in the reference. We extensively researched for these websites and we were able benefit from them.

Project Description

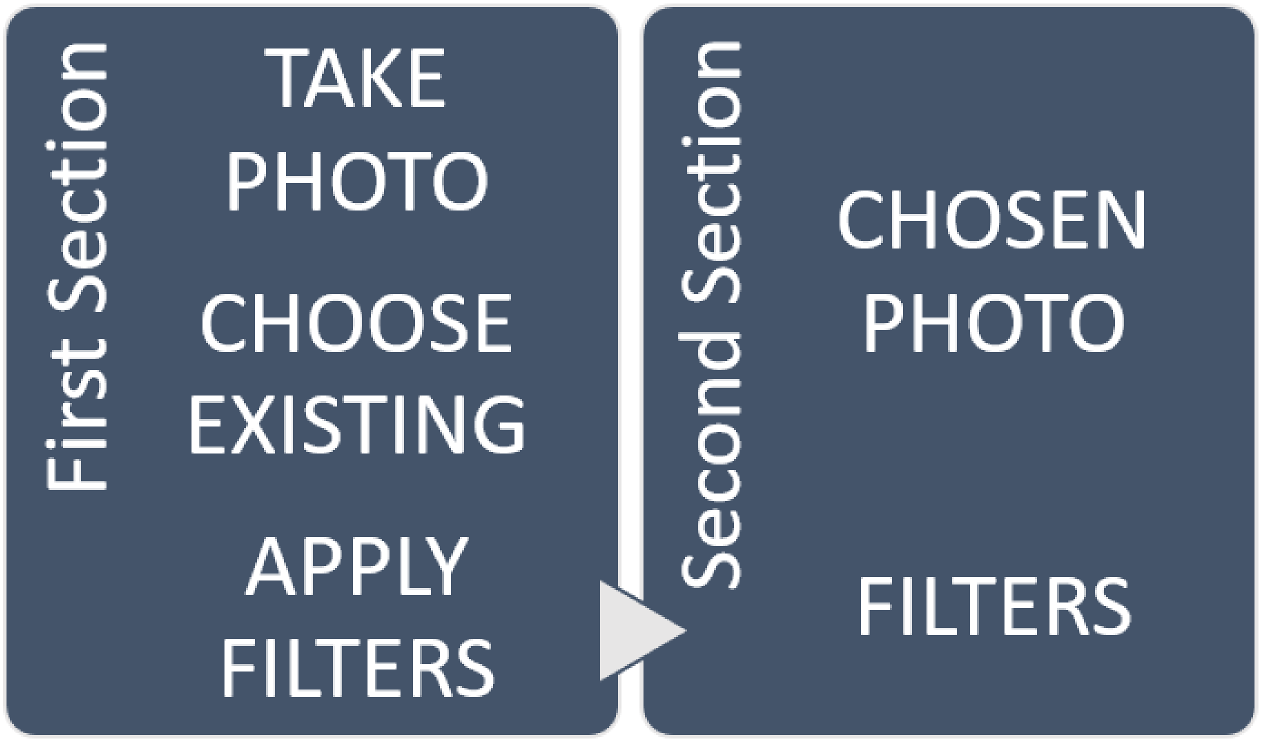

Figure 1: A logic diagram of our app Philter

Our project consists of an app that alters images using various filters. Figure 1 above shows the simple logic diagram. Opening the app, Philter, the user will be prompted to choose whether to create a new image from the phone’s camera or use an already existing image inside the phone. With the chosen image, the user will then be able to choose between the filters provided.

The app was created using Java and XML. These images chosen by the user are alterable in many different ways by the filters implemented on the app. We researched several filters to be used in the app and did as many as we could. The filters we put into the app are Gamma Correction, Changing Brightness, Sepia, Grayscale, Corrosion, Dilation, and Color Inversion.
When creating the filters we did not stay in one particular research paper or tutorial website. The filters were created through a combined effort by Lestley, Anmol, and Aman. Figuring out the codes and the algorithms for the filters was the challenge. In the end from the many papers and tutorials we combed through, we were able to create several filters.

Project Description: Application

The app part of the project has two sections. The first section is the main page seen below in Figure 2.

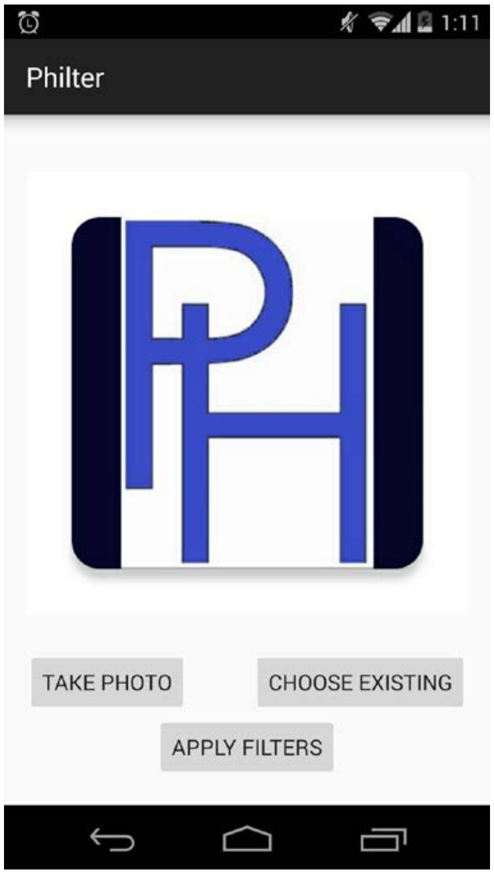

Figure 2: Philter app front page

The first section is the front page of the Philter app and this page will automatically show up when the app is opened. Figure 2 clearly shows the three options that Philter prompts the user to choose from. For the moment, the middle of the screen only shows the Philter logo, PH, because there is no photo indicated to be used. When there is no photo to be used the APPLY FILTERS button does nothing.

The user must first press either the button TAKE PHOTO or CHOOSE EXISTING. Pressing the TAKE PHOTO button leads to the access of the devices camera as shown in Figure 3.

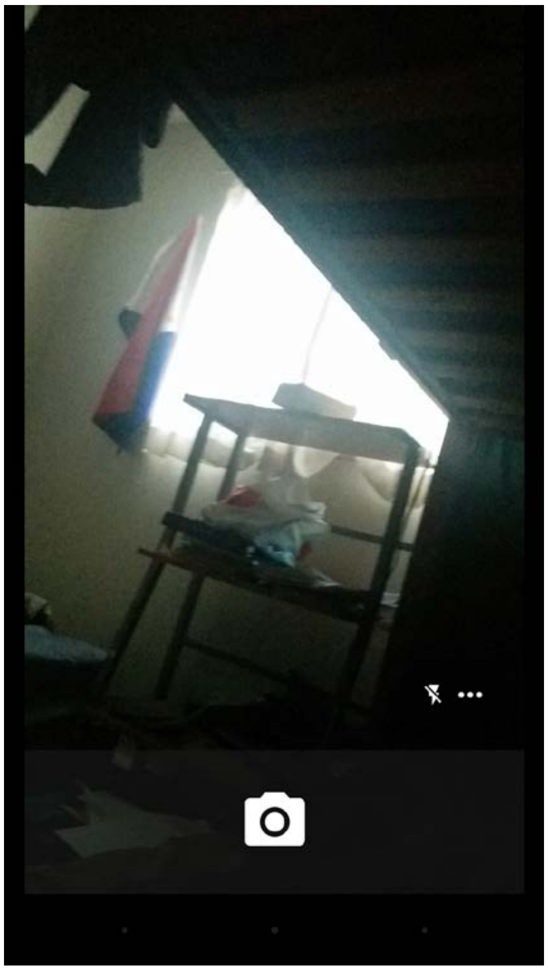
Figure 3: Accessing the camera of the device

Taking a photo just accesses the camera of the device, so the configurations of the photo just depends on the version the Android device's camera is (different versions of Androids have different camera versions). Figure 3 above is taken from the camera of an Android device on the version 4.4.4.

For the second button, CHOOSE EXISTING, this sends the user to the gallery of the device. The gallery of the device is the default built in application where any forms of media from images, screenshots, or videos are saved as seen in Figure 4.

After choosing either the TAKE PHOTO or the CHOOSE EXISTING buttons and then having a photo to use the APPLY FILTERS button should then be pressed.

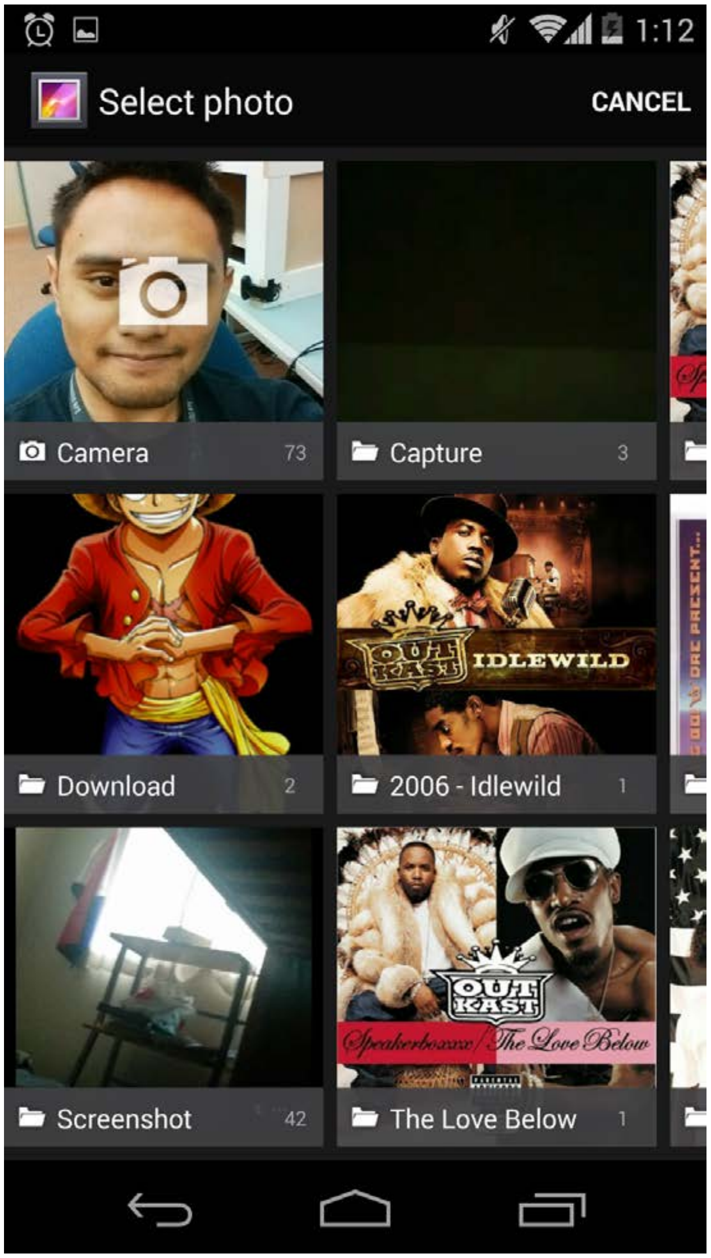

Figure 4: Inside the gallery

As mentioned above, there are two sections of the app we created. The first section was the front page while the second section is the APPLY FILTERS page. When the APPLY FILTERS button is pressed, the user is sent the second page of the app as seen in Figure 5. On this page, the photo used is in the middle, still an original image. The filters to be chosen are below it and can be scrolled through from left to right using a swiping motion. After choosing a filter, on the very right of the filters section, there is an orange button that will save the image to the gallery.

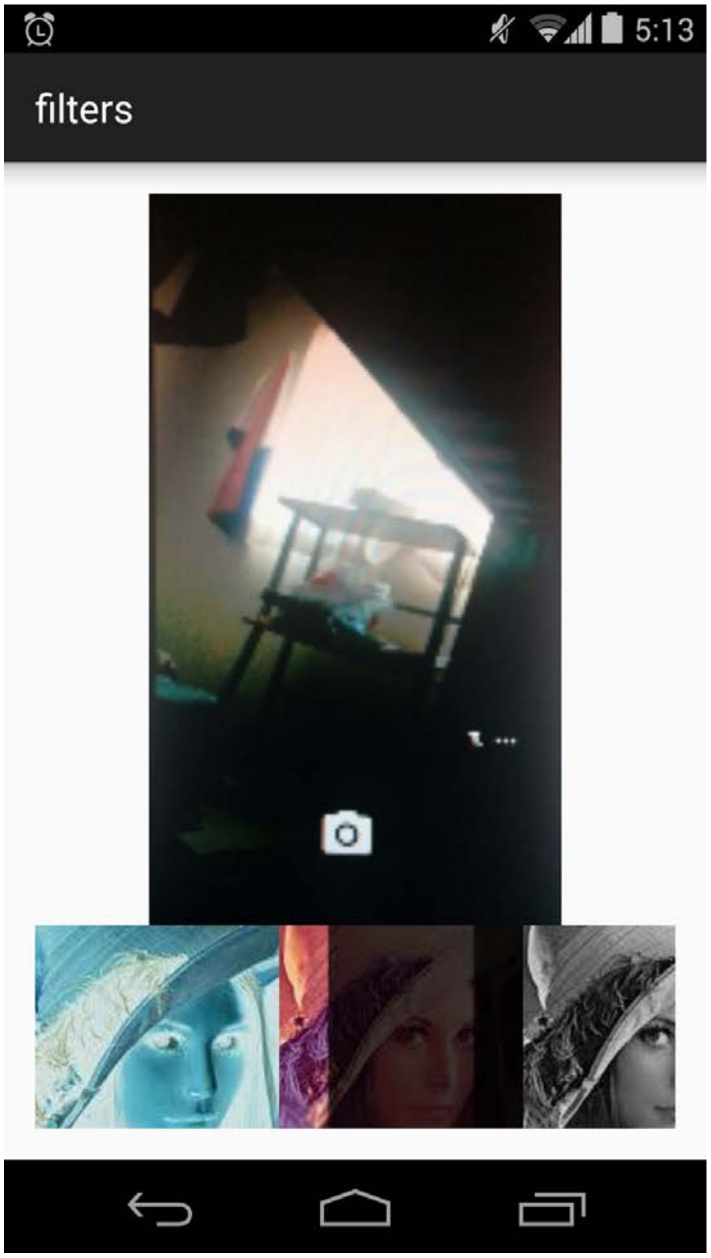

Figure 5: Photo chosen and filters to be chosen

Project Description: Filters

Figure 4 above shows that the filter effects can be seen on a Lena image that has been edited by a filter. This is so the user will know what will happen to the original image once a filter has been chosen.

Each filter on each different website we checked had small differences in their definitions, but the algorithm and its functions were mostly thoroughly explained. Even from just one website a filter could easily be explained. For example, the filter, Gamma Correction, can be understood right away with just the very first search result in Google.

However, this alone did not allow us to instantly have an epiphany of being able to create the filter just through learning what it is.
The more we understood a filter the easier it was for us to actually create them. So several websites that contained journals, blogs, papers, and tutorials had to be visited to confirm exactly what a filter was and what it should be doing. Through the research we have done, we were able to create several filters for our app. These filters had algorithms and/or formulas that needed to be implemented.
Using Figure 6 as our original image, the effects of the filters will be shown through the image manipulation.

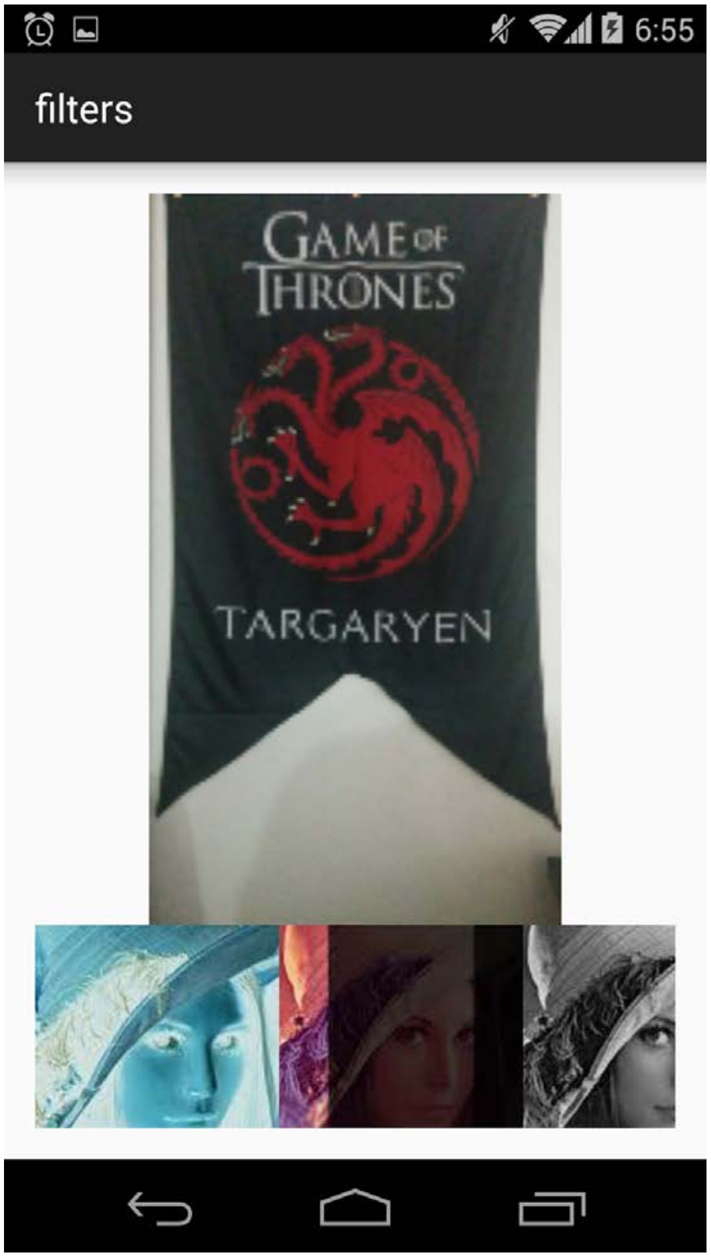

Figure 6: Original image as reference

Grayscale Filter

In this filter, the original coloured jpeg image is converted into a grayscale digital image which is also known as a black and white picture. Conversion of a coloured image to grayscale is not unique.

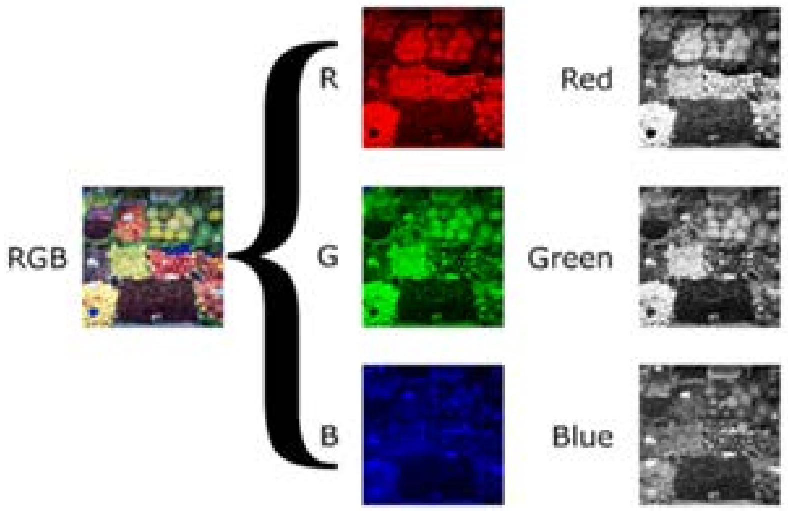

Figure 7: Composition of Red, Green, and Blue (RGB) from 3 Grayscale images

Basic algorithm to convert original to grayscale is

- Get the RGB value of the pixel.
- Find the average of RGB

  new_RGB =(R+G+B)/3
- Replace the R, G and B value of the pixel with average (new_RGB)

To implement the above conversion process we will create two variables x & y and use two for loops to traverse each pixels. From each pixel RGB values are extracted and set new RGB values for a pixel.

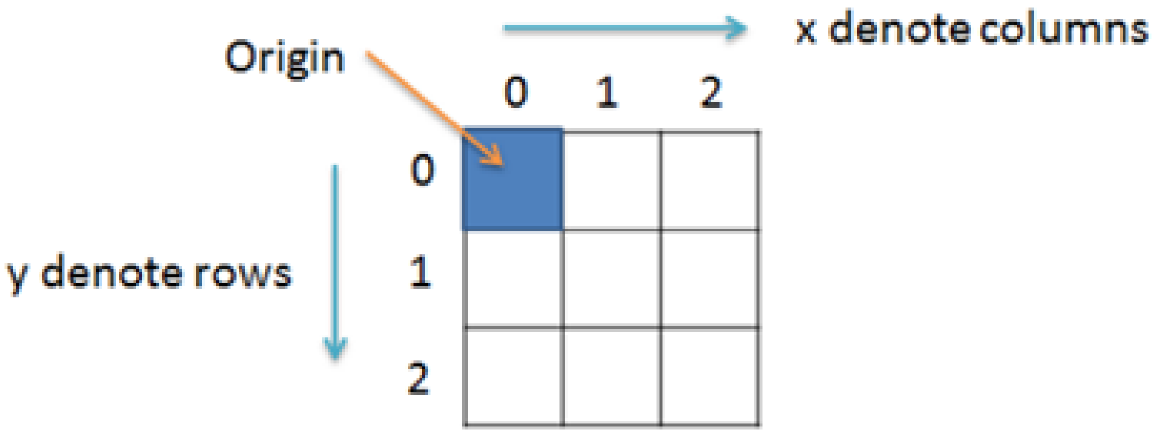

Figure 8: Grayscale code Algorithm to access RGB values for each pixel

This will result in Grayscale image as shown below in Figure 9.

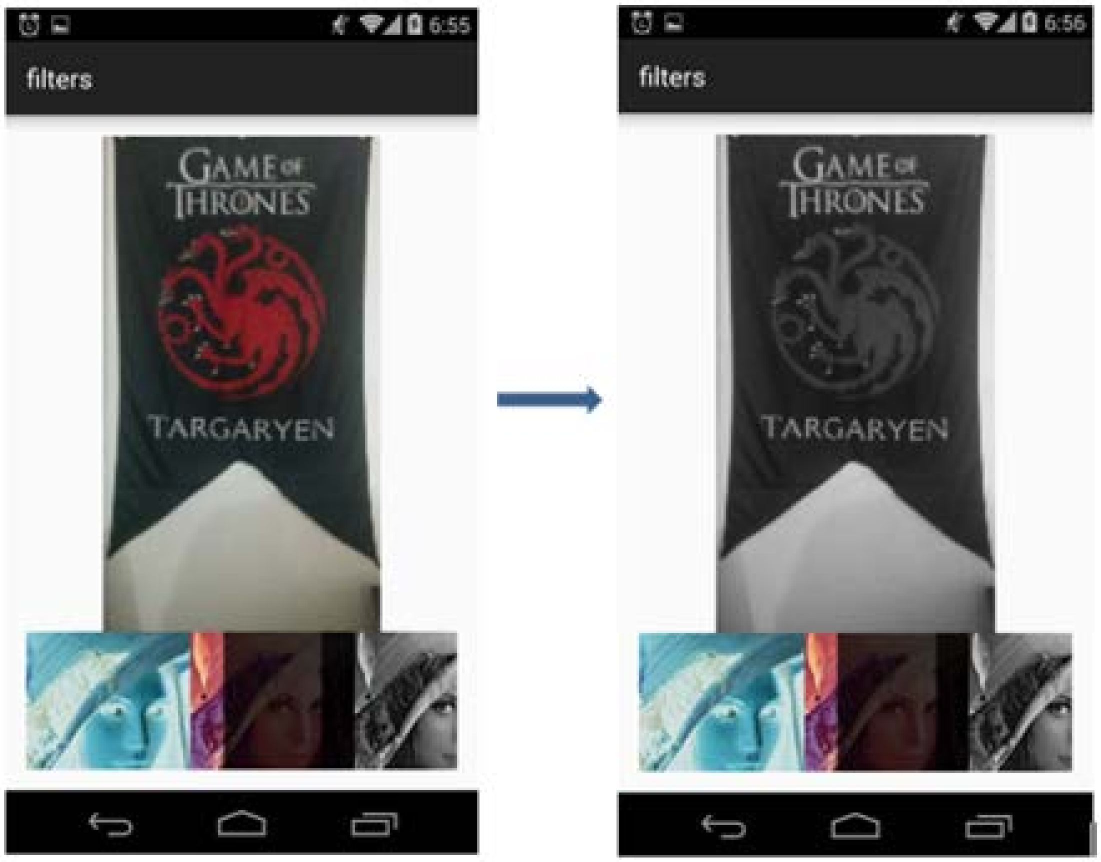

Figure 9: Screenshot of the Grayscale effect

Sepia Filter

Unlike black and white photography, which uses a grayscale, sepia toned photos appear on a brown scale. To convert original coloured image to Sepia image first convert it into grayscale image which follows the same algorithm as shown above. once picture is converted to grayscale then algorithm that is used to convert image to Sepia format.

- Get the RGB value of the pixel.
- Find the average of RGB
  new_RGB =(R+G+B)/3
- Replace the R, G and B value of the pixel with average (new_RGB)
- Sepia depth is increased by adding 40 into R values and adding 20 into G values.
- then check that R,G and B values are less than 255 if not then replace the value to 255.
- user will enter the Sepia Intensity value which is subtracted from the B value this is done to darken blue color to increase sepia effect.
- then check again that value of B is between [0,255] if it is outside the limit then assign the boundary value to B value.

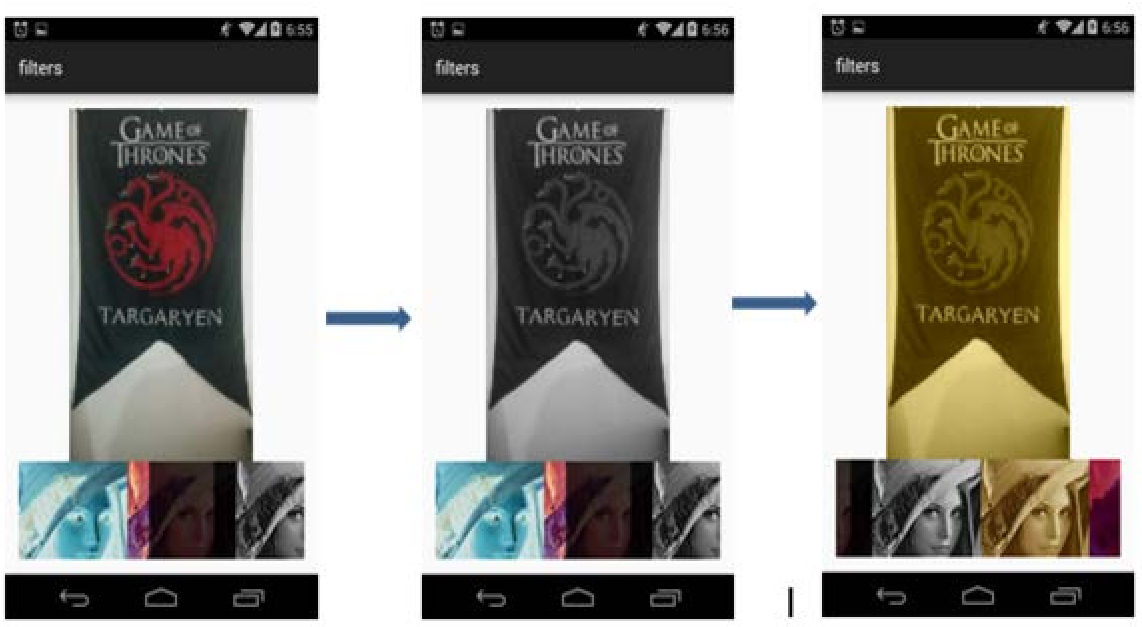

Figure 10: Screenshot of the Sepia effect

Gamma Correction

Gamma is what translates between our eye's light sensitivity and that of the camera [3]. This filter allows the user to control the colors of Red or Green or Blue individually. Figure 11 shows the original image being manipulated in RGB.

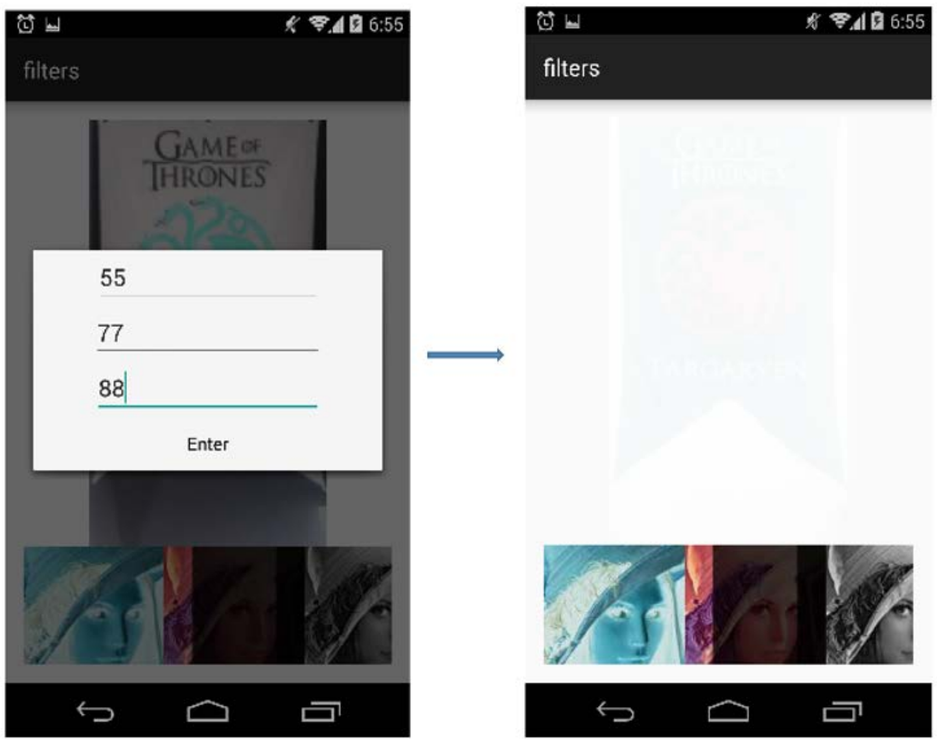

Figure 11: Screenshot of the Gamma Correction effect

Basic algorithm for Gamma Correction:

- First get the RGB pixels of the image, already mentioned above in the Grayscale filter (see Figure 8).
- Set the RGB to the formula below, the P denotes the pixels, the gamma is the y symbol which is inputted by the user [3]

$$P' = 255\ x(\frac{P}{255})^{(1/\gamma)}$$

- Then the 3 input values will control either Red, Green, or Blue.
- Lastly, set an output image with the new pixels.

Changing Brightness

The concept of changing brightness is just the decreasing and increasing of the RGB channels together. Figure 12 shows the brightness of the original image being lowered.

Algorithm for Changing Brightness:

- Get the RGB pixels of the image (shown in Grayscale filter)
- Then given an input from the user, control Red, Green, and Blue (RGB) at the same time.
- Lower given values for RGB means a darker image and higher values for RGB is brighter.

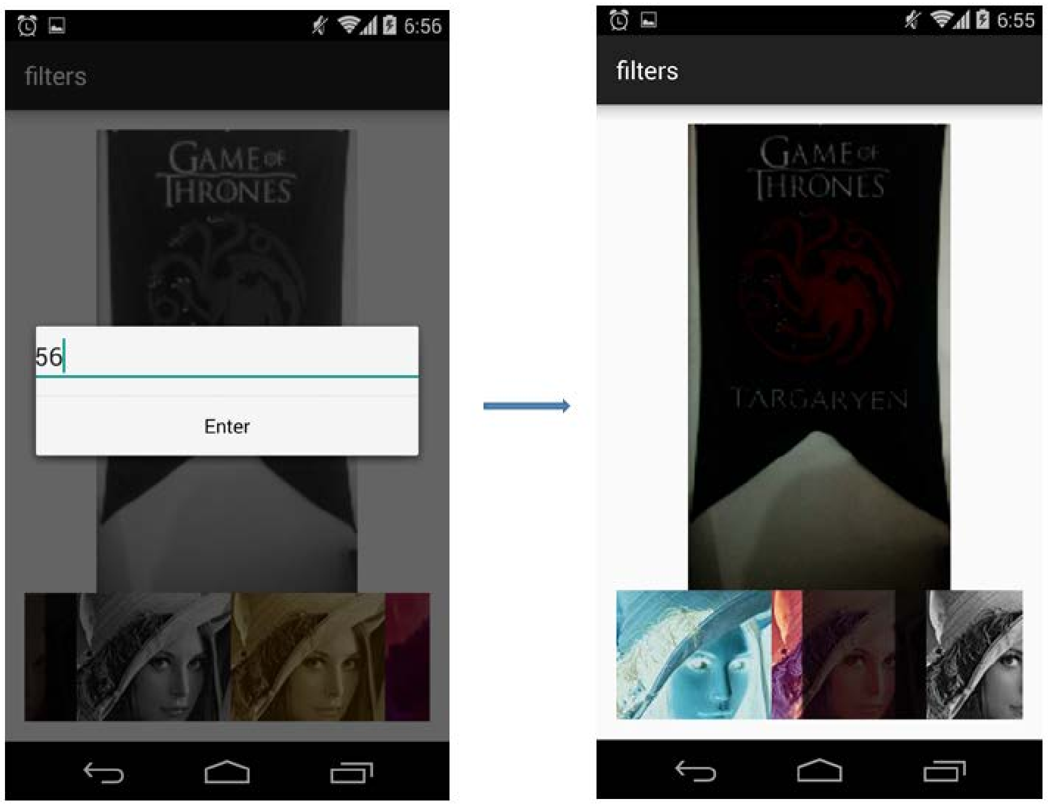

Figure 12: Screenshot of the Changing Brightness effect

Erosion
The erosion filter's effect is to erode away the boundaries of the image. It shrinks the foreground pixels of the image [6]. This shrinking effect is seen in Figure 13.

Algorithm for Erosion:

- Get the foreground pixels of the image (see Grayscale) and then superimpose a structuring element. This separates the image into foreground and background values.
- Using binary as an example, setting foreground values to 1, look at its surrounding neighbours from eight sides and if the neighbour is a 0 then it becomes a 0, too, it becomes a background value.

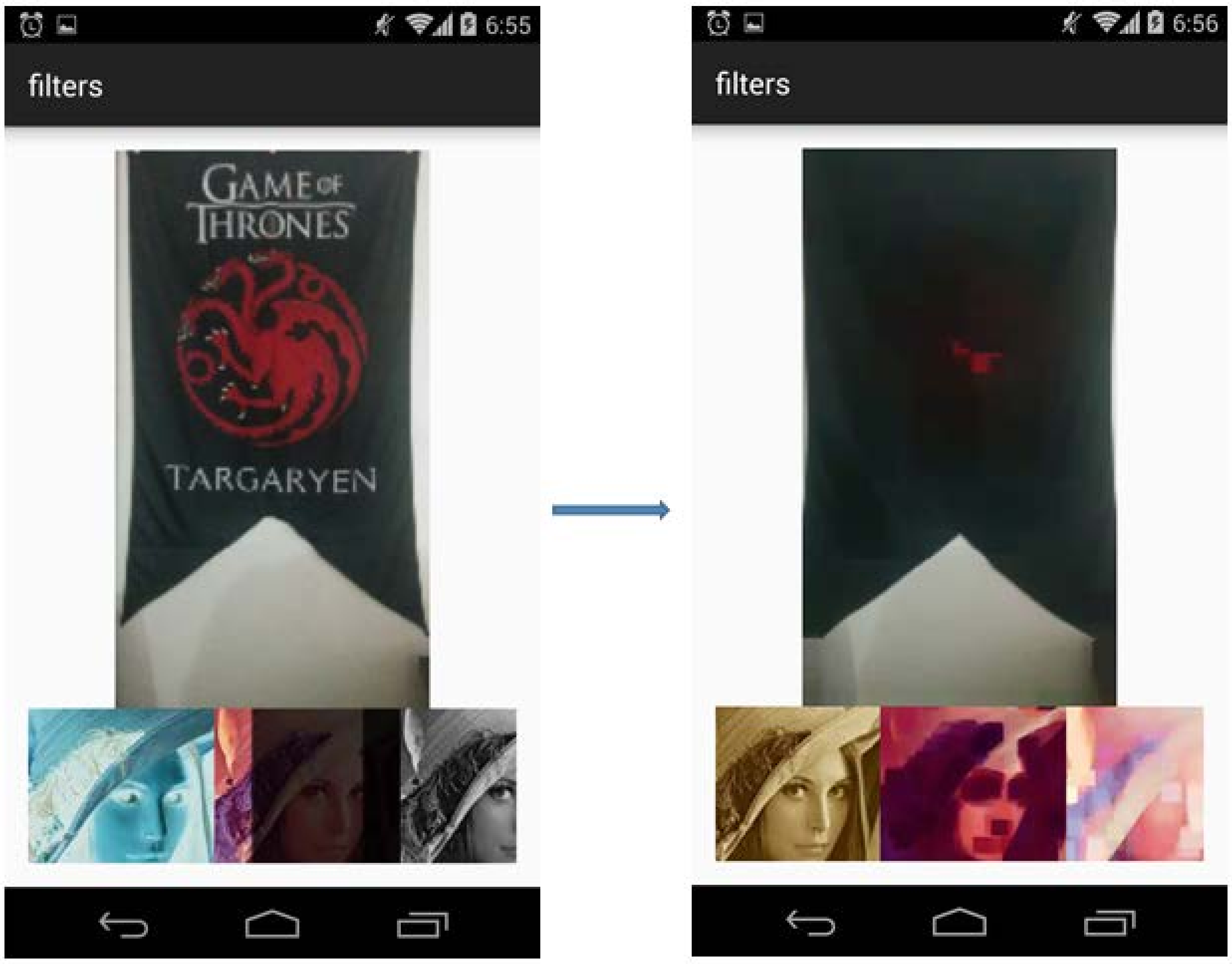

Figure 13: Screenshot of the Corrosion effect

Dilation

The dilation filter is the opposite of the erosion filter. It expands the foreground pixels of the image. This enlarges the boundaries of and allows the pixels to enlarge as seen in Figure 14.

Algorithm for Dilation is the similar to erosion but the opposite:

- Get the foreground pixels of the image (see Grayscale) and then superimpose a structuring element. This separates the image into foreground and background values.
- Using binary as an example, setting foreground values to 1, look at its surrounding neighbours from eight sides and if the neighbour is a 0 then those neighbours also become a 1. The background values become foreground values.

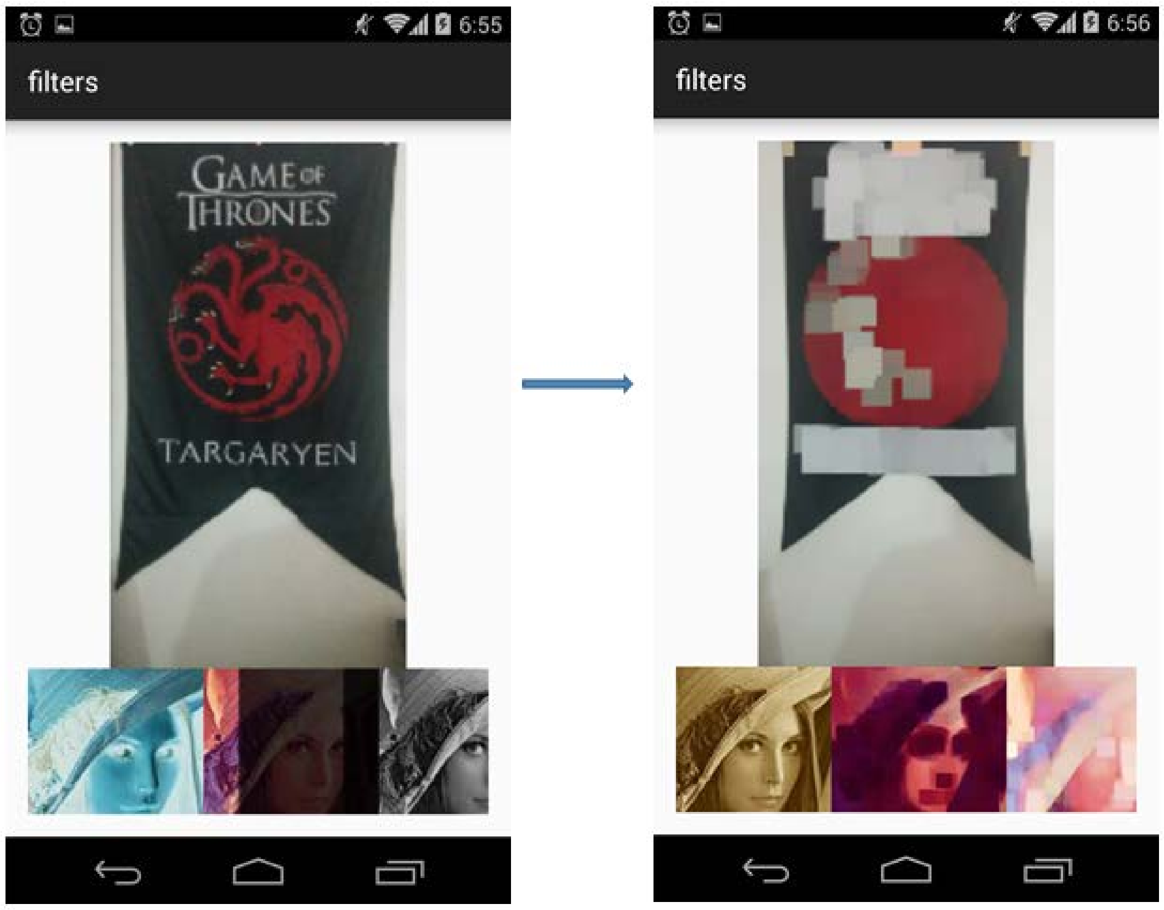

Figure 14: Screenshot of the Dilation effect

Contribution
Anmol - framework of the creation of the application, integration of the codes, report
Lestley - coding of the filters, presentation and demonstration slides, report
Aman - coding of the filters, report
Apassara - presentation and demonstration slides, report

Conclusion
Image editing apps are very popular and we wanted to do something that we could evolve from a school project. If we could add more advanced filters into our application we definitely release Philter into Google Play. We have learned a lot about filters. One important we learned was that many filters relied on the manipulation of the RGB and alpha channels. While others, like the erosion and dilation relied on a structuring element to manipulate the image.

Reference

[1] android-arsenal.com
[2] jhlabs.com
[3] http://developer.bostjan-cigan.com/java-gamma-correction-algorithm/ - gamma correction
[4] http://www.porterhousesy1.co.uk/ - Icon
[5] http://blog.ostermiller.org/dilate-and-erode - dilation and erosion
[6] http://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm - erosion
[7] http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm - dilation
[8]http://www.dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-grayscale-image-in-java - Grayscale